EPTCS 426

Proceedings of the

22nd International Conference on

Quantum Physics and Logic

Varna, Bulgaria, 14 July 2025 - 18 July 2025

Edited by: Alejandro Díaz-Caro, Ognyan Oreshkov and Ana Belén Sainz

Published: 20th August 2025
DOI: 10.4204/EPTCS.426
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Preface

Alejandro Díaz-Caro
Université de Lorraine, CNRS, Inria, LORIA
France
Universidad Nacional de Quilmes
Argentina
alejandro.diaz-caro@inria.fr

Ognyan Oreshkov
Centre for Quantum Information and Communication
Université libre de Bruxelles
Belgium
ognyan.oreshkov@ulb.be

Ana Belén Sainz
International Centre for Theory of Quantum Technologies
University of Gdańsk
Poland
ana.sainz@ug.edu.pl

This volume contains the proceedings of the 22nd International Conference on Quantum Physics and Logic (QPL 2025). The conference was held from 14 to 18 July 2025 in Varna, Bulgaria.

Quantum Physics and Logic is a series of conferences that brings together academic and industry researchers working on the mathematical foundations of quantum computation, quantum physics, and related areas. The main focus is on the use of algebraic and categorical structures, formal languages, type systems, semantic methods, and other mathematical and computer science techniques applicable to the study of physical systems, physical processes, and their composition. Work applying quantum-inspired techniques and structures to other fields (such as linguistics, artificial intelligence, and causality) is also welcome.

The QPL 2025 conference solicited three kinds of submissions: proceedings submissions, non-proceedings talk submissions, and poster submissions.

Proceedings submissions were papers required to provide sufficient evidence of results of genuine interest. Authors of accepted proceedings submissions were given the opportunity to present their work as a talk at the conference, and these papers were included in the proceedings of QPL 2025. No other types of submissions were considered for inclusion in the proceedings. Non-proceedings talk submissions consisted of a three-page summary, together with a link to a separate published paper or preprint. Authors of accepted non-proceedings talk submissions were invited to present their work as a talk during the conference. Poster submissions consisted of a three-page abstract of (possibly partial) results or work in progress, and authors of accepted poster submissions were invited to present their work during the poster session.

These proceedings contain 12 contributed papers selected for publication by the Program Committee. Papers submitted to QPL undergo a review process managed by members of the Program Committee (PC). The vast majority of submissions received at least three reviews. The selection of accepted papers was done through the EasyChair conference management system, following consideration of the submitted reviews and, where necessary, discussion among the PC members. The review process was single-blind: the identity of the authors was revealed to the reviewers, but not vice versa. PC members could invite external experts to serve as subreviewers and participate in discussions of the submissions they reviewed.

A total of 129 submissions (excluding withdrawals and retractions) were considered for review by the PC. QPL 2025 had 54 accepted submissions in the non-proceedings talk track and 12 accepted submissions in the proceedings track. Some of the talks were presented during parallel sessions, and a

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. ii–iii, doi:10.4204/EPTCS.426.0

selection of talks was presented during plenary sessions. The program also included a poster session with 29 posters out of all the 57 submissions accepted as posters, and one special session dedicated to quantum in the context of education. There was also an industry session where industrial sponsors of QPL 2025 were given the opportunity to present their companies. The industry session consisted of 2 talks—one by Quantinuum and one by NeverLocal, both of them Platinum sponsors.

The QPL 2025 conference featured an award for Best Student Paper. Papers eligible for the award were those in which all the authors were students at the time of submission. The PC decided to award the Best Student Paper award for QPL 2025 to Vivien Vandaele (Eviden Quantum Lab and Université de Lorraine, CNRS, Inria, LORIA) for his paper "Lower T-count with faster algorithms".

The official website of the conference is `https://qpl2025.github.io` and it contains a lot of relevant information about QPL 2025.

The Program Committee consisted of 56 members: Alastair Abbott, Pablo Arrighi, Miriam Backens, Rui Soares Barbosa, Ämin Baumeler, Jessica Bavaresco, Alessandro Bisio, Robert Booth, Cyril Branciard, Titouan Carette, Ulysse Chabaud, Giulio Chiribella, Bob Coecke, Marcelo Terra Cunha, Alejandro Díaz-Caro, Ross Duncan, Pierre-Emmanuel Emeriau, Flaminia Giacomini, Stefano Gogioso, Matty Hoban, Emmanuel Jeandel, Anna Jenčová, Robin Kaarsgaard, Martti Karvonen, Kohei Kishida, Aleks Kissinger, Ravi Kunjwal, Robin Lorenz, Glaucia Murta, Nuriya Nurgalieva, Ognyan Oreshkov, Anna Pearson, Simon Perdrix, Nicola Pinzani, Robert Rand, Neil Ross, Mehrnoosh Sadrzadeh, Ana Belén Sainz, Nitica Sakharwade, Carlo Maria Scandolo, John Selby, Peter Selinger, Sonja Smets, Pawel Sobocinski, Isar Stubbe, Benoît Valiron, Augustin Vanrietvelde, Vilasini Venkatesh, Renaud Vilmart, Juliana Kaizer Vizzotto, Quanlong Wang, Julian Wechs, Mirjam Weilenmann, John van de Wetering, Alexander Wilce, and Margherita Zorzi.

The Organising Committee consisted of six members: James Hefford, Timothée Hoffreumon, Ognyan Oreshkov (Chair), Eleftherios-Ermis Tselentis, Zixiuan Liu, and Vladimir Zamdzhiev. The conference also had a dedicated Safety and Inclusion Team overseeing the Code of Conduct, which consisted of Ravi Kunjwal, Ana Belén Sainz, and Eleftherios-Ermis Tselentis.

The QPL Steering Committee consisted of Bob Coecke, Ana Belén Sainz, and Peter Selinger.

We wish to thank all the members of the PC for their work in selecting the program of QPL 2025. We also thank all external subreviewers for their help and the authors for their submissions to QPL 2025. We are grateful to the EPTCS team for their help in preparing the proceedings of the conference. Finally, we thank the QPL Steering Committee for their support and all the people who have contributed to the success of QPL 2025.

QPL 2025 received financial support from Quantinuum and NeverLocal, and administrative support from the Université libre de Bruxelles.

July 2025,

Alejandro Díaz-Caro
Ognyan Oreshkov
Ana Belén Sainz

Proto-Quipper with Reversing and Control

Peng Fu
University of South Carolina, USA

Kohei Kishida
University of Illinois Urbana-Champaign, USA

Neil J. Ross

Peter Selinger
Dalhousie University, Canada

The quantum programming language Quipper supports circuit operations such as reversing and controlling certain quantum circuits. Additionally, Quipper provides a function called *with-computed*, which can be used to program circuits of the form $g; f; g^\dagger$. The latter is a common pattern in quantum circuit design. One benefit of using with-computed, as opposed to constructing the circuit $g; f; g^\dagger$ directly from g, f, and $g^\dagger$, is that it facilitates an important optimization. Namely, if the resulting circuit is later controlled, only f needs to be controlled; the circuits g and $g^\dagger$ need not even be controllable.

In this paper, we formalize a semantics for reversible and controllable circuits, using a dagger symmetric monoidal category **R** to interpret reversible circuits, and a new notion we call a *controllable category* **N**, which encompasses the control and with-computed operations in Quipper. We extend the language Proto-Quipper with reversing, control and the with-computed operation. Since not all circuits are reversible and/or controllable, we use a type system with modalities to track reversibility and controllability. This generalizes the modality of Fu-Kishida-Ross-Selinger 2023. We give an abstract categorical semantics, and show that the type system and operational semantics are sound with respect to this semantics.

1 Introduction

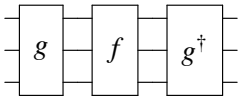
The goal of this paper is to devise a strongly typed, functional programming language that can formally treat the quantum operations of reversing and control. Many quantum algorithms take essential advantage of reversing, control, and their associated operation "with-computed". At the same time, some operations, such as state preparations and measurements, can make circuits irreversible or uncontrollable. It is therefore desirable to equip quantum programming languages with a type system that can prevent the user from trying to reverse the irreversible or to control the uncontrollable. In this paper, we incorporate reversing, control, and with-computed into the language *Proto-Quipper*.

Proto-Quipper is a family of programming languages that aims to provide a formal syntax and semantics for various features of the embedded quantum programming language Quipper [12, 13]. Like Quipper, Proto-Quipper is a quantum circuit description language. When a Proto-Quipper program is run, it outputs a quantum circuit; this generated circuit can later be executed on a quantum computer. We refer to these two runtimes as "circuit generation time" and "circuit execution time", respectively. There are various errors that programmers can introduce by attempting the impossible—e.g., to duplicate linear data, to apply circuit operations to a program that is not a circuit, etc.—but since Proto-Quipper is strongly typed, it can catch such errors at compile time, rather than at circuit generation time or circuit execution time. Previous research on Proto-Quipper includes providing a syntax and semantics for circuit boxing [19, 20], supporting recursion [17], combining linear types and dependent types [6, 10], and incorporating dynamic lifting [3, 7, 8, 16]. This paper defines a new variant Proto-Quipper-C, which extends the type system of Proto-Quipper with reversibility and controllability.

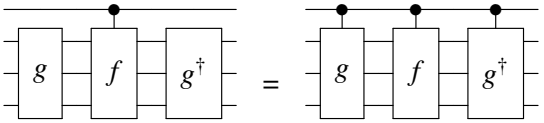
A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 1–22, doi:10.4204/EPTCS.426.1

1.1 Reversing, control, and with-computed

In Quipper, the `controlled` function inputs a circuit and returns the controlled version of the circuit. The `reverse` function inputs a circuit and returns its adjoint. The program (`with_computed` g f) first performs a circuit g, then a circuit f, and finally the adjoint of g.



The *with-computed* circuit pattern is very common in quantum computing. For example, Bennett's method for constructing reversible classical circuits [1] uses this pattern to initialize ancillary qubits and uncompute them; and the quantum Fourier transform (QFT) addition [4] conjugates a series of controlled rotation gates by the QFT. This pattern is used so often that Quipper implements the following optimization: when controlling a quantum circuit $g; f; g^\dagger$ generated by the with-computed function, it suffices to control the circuit f.



In fact, the circuits g and $g^\dagger$ need not even be controllable. Thus the left hand side of the above circuit identity is more general than the right hand side. It is also more efficient, since it uses a smaller number of controlled gates. Note that in quantum computing, a controlled gate may require substantially more resources than the non-controlled version. For example, a common way of measuring resources is the *T-count*, i.e., the number of T-gates required to implement a gate. While a T-gate itself has T-count 1, a controlled T-gate has a T-count of at least 5.

1.2 Modalities for reversing and control

In Quipper, controlling an uncontrollable circuit or reversing a non-reversible circuit will give rise to runtime errors. We want to incorporate reversing and control in Proto-Quipper in such a way that erroneously controlling or reversing a circuit is detected as a typing error at compile time. We achieve this by introducing a notion of modality $\alpha \in \{0,1,2\}$. Here, the modality 2 is associated with circuits that are both controllable and reversible, for example a Hadamard gate. The modality 1 is associated with circuits that are reversible but not controllable, for example an initialization gate. (Recall that "reversing" means taking the adjoint of a circuit, not necessarily the inverse. For a more detailed discussion of this circuit model, see [13, Sec. 4.2].) The modality 0 is associated with circuits that are neither controllable nor reversible, for example a measurement gate. Note that because all controllable quantum gates are unitary and therefore reversible, we do not include a modality for circuits that are controllable but not reversible.

We can use modalities to annotate the type of a circuit. E.g., the values of the type $\mathbf{Circ}_\alpha(S,U)$ are circuits with input type S, output type U and a modality annotation α. One important feature of modalities is that they are composable. Suppose we want to compose a circuit $\mathbf{Circ}_\alpha(S,U)$ with a circuit $\mathbf{Circ}_\beta(U,S')$. The resulting circuit will have type $\mathbf{Circ}_{\alpha\wedge\beta}(S,S')$, where $\alpha\wedge\beta = \min(\alpha,\beta)$. This means that as long as we know the modalities for the basic gate sets, we can devise a type system to track the modalities of the circuits constructed from basic gates. Moreover, the reversing, control, and with-computed operations can be given the following types:

$$\begin{aligned}
\mathsf{control}_S &: \mathbf{Circ}_2(U,U) \to \mathbf{Circ}_2(S\otimes U, S\otimes U). \\
\mathsf{reverse} &: \mathbf{Circ}_\alpha(U,S) \to \mathbf{Circ}_\alpha(S,U), && \text{where } \alpha > 0. \\
\mathsf{withComputed} &: \mathbf{Circ}_\alpha(U,S) \times \mathbf{Circ}_2(S,S) \to \mathbf{Circ}_2(U,U), && \text{where } \alpha > 0.
\end{aligned}$$

Note that withComputed requires its first argument to be at least reversible, and the resulting circuit is again controllable.

Our modal type system makes it possible to detect errors, like controlling an uncontrollable circuit, at compile time. In a practical implementation (such as the one available from [5]), it is moreover possible for the type checker to automatically infer modalities, so that the programmer does not need to explicitly specify them in the source code. See Appendix A for some sample code from our prototype implementation.

1.3 Related work

The reverse, control and with-computed operations originally appeared in Quipper [13], but were lacking a formal semantics. In our work on Proto-Quipper [7, 8], we gave type systems with modalities for a feature called dynamic lifting, but not for reversing and control.

Many practical quantum programming languages, such as Qiskit and Cirq also include the concepts of reversibility and controllability. However, since these languages are imperative, errors of reversibility or controllability are treated as runtime exceptions [11, 15]. The functional quantum programming language QWire [18] supports circuit reversal, but not the control and with-computed operations. The language Silq [2] has type annotations **qfree** and **mfree**, which denote classical functions and functions that do not use measurement, respectively. Silq's core language supports reversing, and the control operation is supported by an if-then-else construct. The main difference between Silq and Proto-Quipper is that Silq is not a circuit description language and does not have a notion of boxed circuits.

We are only aware of one language besides Quipper that has a with-computed operation, namely ProjectQ [14, 23]. However, ProjectQ is not a strongly-typed language.

1.4 Contributions

In this paper, we formalize a notion of controllable category. We extend the Proto-Quipper type system of [8] with modalities for reversing and control and define an operational semantics. We provide safety properties that guarantee that a well-typed program is free of run-time errors. We also give an axiomatization of an abstract categorical model of Proto-Quipper with reversing, control, and the with-computed operation, and show that the operational semantics is sound for it.

2 Controlling a permutation circuit

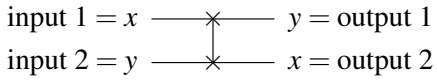
In Proto-Quipper, there are two ways to represent the swap operation. One way is by permuting the logical order of the circuit outputs without using any gates, as in the following diagram.

$$\begin{aligned}
\text{input } 1 = x \quad &\text{————} \quad x = \text{output } 2 \\
\text{input } 2 = y \quad &\text{————} \quad y = \text{output } 1
\end{aligned} \tag{1}$$

The above circuit is generated by the following Proto-Quipper program.

```
f : !(Qubit * Qubit -> Qubit * Qubit)
f input = let (x, y) = input in (y, x)
```

The other way is by using an explicit swap gate, as in the following diagram.



(2)

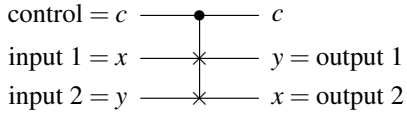
The corresponding Proto-Quipper program is the following.

```
g : !(Qubit * Qubit -> Qubit * Qubit)
g input = let (x, y) = input in Swap x y
```

These circuits are semantically equivalent: each of them sends (x,y) to (y,x). However, care must be taken when controlling these circuits, since controlling them naively may produce the following non-equivalent circuits.

(a)

(b)



(3)

While the second circuit is correct, the first one is not. Indeed, the first circuit will send input 1 to output 2 independently of the state of the control qubit. This is not the correct behavior, since the swapping should only take place when the control qubit is in the state $|1\rangle$. When the control qubit is in the state $|0\rangle$, the functions f and g should both behave like the identity function on `Qubit * Qubit`.

Therefore, the programming language must be aware not only of the circuit, but also of the implicit permutation of any qubits performed in the language. If such an operation is controlled, explicit swap gates must sometimes be inserted. Proto-Quipper-C does this correctly, whereas the original Quipper implementation did not. In Proto-Quipper-C, controlling circuit (1) and circuit (2) both result in (3)(b), not (3)(a).

3 A formal model of reversing, control and with-computed

Before we can define a programming language for building quantum circuits, we must specify what a quantum circuit is. However, there exist many different classes of circuits; for example, they differ by what data they can manipulate (only qubits, or also classical bits, and/or more exotic objects like qutrits), what the built-in gates are (for example, the Clifford+T gate set, Toffoli+Hadamard, rotations by arbitrary angles), whether or not qubit initialization and termination is supported, whether measurement is considered as a gate, and so on. Therefore, rather than tailoring our programming language to a specific class of circuits, we make both the language and its operational and denotational semantics *parametric* on a given class of circuits. This is analogous to making a classical programming language parametric on some signature of built-in operations. For us, quantum circuits are abstractly given as the morphisms of monoidal categories with certain properties, which we now specify.

We start from a given symmetric monoidal category $\mathbf{M}$. In practice, the objects of $\mathbf{M}$ are typically generated from wire types such as **Bit** and **Qubit**, and the morphisms are quantum circuits generated from a finite set of gates. In the following, we define what we mean by a reversible subcategory of $\mathbf{M}$. Recall that a dagger symmetric monoidal category is a symmetric monoidal category equipped with a contravariant, identity-on-objects, involutive functor such that for all morphism $f : A \to B$, we have $f^\dagger : B \to A$. Moreover, the dagger functor is required to be compatible with the symmetric monoidal structure [21].

Definition 3.1. By a *reversible subcategory* of $\mathbf{M}$, we mean a dagger symmetric monoidal category $\mathbf{R}$, together with an identity-on-objects, faithful, symmetric (strong) monoidal functor $I : \mathbf{R} \to \mathbf{M}$. We usually regard I as an inclusion functor, i.e., we regard $\mathbf{R}$ as a subcategory of $\mathbf{M}$.

A morphism f of $\mathbf{R}$ is called *unitary* if $f \circ f^\dagger = \mathrm{id}$ and $f^\dagger \circ f = \mathrm{id}$. Note that we do not require all morphisms of $\mathbf{R}$ to be unitary. Thus, our notion of reversible morphism means a morphism that has an adjoint, not necessarily an inverse. The dagger functor provides a semantics for the operation of reversing a quantum circuit.

In order to capture the notion of control and with-computed, we define a notion of controllable category $\mathbf{N}$. To motivate the following definition, we should first clarify some use cases. The most common example of control from quantum computing is the control usually represented by a black dot "•". This kind of control "fires" the gate if the control qubit is in state $|1\rangle$, and acts as the identity otherwise. There are also other ways of controlling a qubit; for example, the white dot "∘" fires if the control qubit is in state $|0\rangle$. A more general kind of control in quantum computing is as follows: Let V, W be Hilbert spaces, let K be a subspace of V, and let $G : W \to W$ be a gate. We define the K-controlled gate $\mathsf{ctrl}_K(G) : V \otimes W \to V \otimes W$ by $\mathsf{ctrl}_K(G)(v \otimes w) = v \otimes G(w)$ if $v \in K$ and $\mathsf{ctrl}_K(G)(v \otimes w) = v \otimes w$ if $v \in K^\perp$, extended to all other states by linearity. Note that this general notion of control not only encompasses the black and white dot, but also many other kinds; for example, it is possible to control a gate on the state $|+\rangle$, to control on multiple qubits and/or classical bits, in which case the subspace K may or may not be 1-dimensional. Even the trivial control is possible, namely when $K = V$; in this case the gate always "fires".

Working in an abstract monoidal category, we will not talk about subspaces and their orthogonal complements. Instead, we consider a monoid $\mathbf{C}$ of *controls*, regarded as a discrete monoidal category, together with a monoidal functor $F : \mathbf{C} \to \mathbf{N}$. We write $\underline{K}$ for $F(K)$. The idea is that each $K \in \mathbf{C}$ specifies a kind of control on the object $\underline{K} \in \mathbf{N}$.

Definition 3.2. Let $\mathbf{R}$ be a dagger symmetric monoidal category. By a *controllable category* of $\mathbf{R}$, we mean a dagger symmetric monoidal category $\mathbf{N}$ with the same objects as $\mathbf{R}$, and equipped with the following structure:

(**a**) For all $A, B \in \mathbf{N}$, a function $- \bullet - : \mathbf{R}(B,A) \times \mathbf{N}(A,A) \to \mathbf{N}(B,B)$ (also denoted by $\mathsf{withComputed}$) such that:

$$\mathrm{id} \bullet h = h$$

$$(g_1; g_2) \bullet h = g_1 \bullet (g_2 \bullet h).$$

$$(g_1 \otimes g_2) \bullet (h_1 \otimes h_2) = (g_1 \bullet h_1) \otimes (g_2 \bullet h_2)$$

$$(g \bullet h)^\dagger = g \bullet h^\dagger$$

(**b**) A discrete monoidal category $\mathbf{C}$ of *controls*, together with a monoidal functor mapping $K \in \mathbf{C}$ to $\underline{K} \in \mathbf{N}$.

(**c**) For all $A \in \mathbf{N}$ and $K \in \mathbf{C}$, a function $\mathsf{ctrl}_K : \mathbf{N}(A,A) \to \mathbf{N}(\underline{K} \otimes A, \underline{K} \otimes A)$ such that the following hold:

$$\mathsf{ctrl}_K(\mathrm{id}_A) = \mathrm{id}_{\underline{K} \otimes A}$$

$$\begin{array}{ccc} \underline{I} \otimes A & \xrightarrow{\mathsf{ctrl}_I(h)} & \underline{I} \otimes A \\ \downarrow \cong & & \downarrow \cong \\ A & \xrightarrow{h} & A, \end{array}$$

$$\begin{array}{ccc}
(\underline{A \otimes B}) \otimes C & \xrightarrow{\mathsf{ctrl}_{A\otimes B}(h)} & (\underline{A \otimes B}) \otimes C \\
\downarrow \cong & & \downarrow \cong \\
\underline{A} \otimes (\underline{B} \otimes C) & \xrightarrow{\mathsf{ctrl}_A(\mathsf{ctrl}_B(h))} & \underline{A} \otimes (\underline{B} \otimes C),
\end{array}$$

$$\begin{array}{ccc}
(\underline{A \otimes B}) \otimes C & \xrightarrow{\mathsf{ctrl}_{A\otimes B}(h)} & (\underline{A \otimes B}) \otimes C \\
\downarrow \cong & & \downarrow \cong \\
(\underline{B \otimes A}) \otimes C & \xrightarrow{\mathsf{ctrl}_{B\otimes A}(h)} & (\underline{B \otimes A}) \otimes C.
\end{array}$$

(**d**) An identity-on-objects, strict dagger symmetric monoidal functor $G : \mathbf{N} \to \mathbf{R}$ such that

$$G(g \bullet h) = g; G(h); g^\dagger$$

Remarks.

- The functor G gives an intended interpretation for the with-computed function. It captures the common quantum computation pattern $g; h; g^\dagger$.
- The functor G preserves the equalities of condition (a). For example, $G(\mathrm{id} \bullet h) = \mathrm{id}; G(h); \mathrm{id}^\dagger = G(h)$. Note that we do not require $g \bullet \mathrm{id}_A = \mathrm{id}_B$, because this would imply $g; g^\dagger = g; \mathrm{id}_A; g^\dagger = G(g \bullet \mathrm{id}_A) = G(\mathrm{id}_B) = \mathrm{id}_B$, which is not an assumption we make for **R**. For similar reasons, we do not require $g \bullet (h_1; h_2) = (g \bullet h_1); (g \bullet h_2)$.

To illustrate how the above notions are applicable to quantum computing, we can consider a "standard model" of these axioms, parameterized by a class of quantum circuits. Suppose the circuits are generated by some types of wires (for example, **Qubit** and **Bit**) and some set of gates. Suppose that some distinguished subset of the gates are reversible, and a further subset of these are controllable. Also suppose that the class of circuits is equipped with some primitive control gadgets, for example, a black-dot and a white-dot control on qubits, and that for each gate, all of its controlled versions are also gates. The category **M** consists of all circuits, arranged into a symmetric monoidal category (the symmetric monoidal structure permits the crossing of wires, but such crossings are not considered to be gates). Let **R** be the subcategory of circuits that are made from reversible gates. Finally, the structure of **N** is somewhat interesting. The morphisms of **N** are circuits made from gates of two colors, say red and green. Each red gate is reversible and each green gate is controllable (and in particular, each controllable gate comes in two different colors). All morphisms in **N** are reversible, and reversing does not change color. The objects of the monoidal category **C** are sequences of control gadgets, for example $\bullet \circ \circ$, and $\underline{\bullet \circ \circ} = \mathbf{Qubit} \otimes \mathbf{Qubit} \otimes \mathbf{Qubit}$. The control function adds controls to the green gates. The with-computed function takes any morphism g of **R** and any morphism f of **N**, and produces $g; f; g^\dagger$, where g and $g^\dagger$ have been colored red. The functor G is the forgetful functor that forgets the colors.

4 A type system for reversing, control and with-computed

In this section, we define the syntax and type system of Proto-Quipper-C, a version of Proto-Quipper with support for reversing, control and with-computed. We assume that we are given three fixed small categories **M**, **R**, and **N** as specified in the previous section. The type system and (in later sections) the operational semantics and categorical semantics are all parameterized by this choice.

Modalities	α, β	::=	$2 \mid 1 \mid 0$
Types	A, B	::=	$\mathbf{Unit} \mid W \mid \mathbf{Bool} \mid !_\alpha A \mid A \multimap_\alpha B$
			$\mid \mathbf{Circ}_\alpha(S,U) \mid A \otimes B$
Parameter Types	P, R	::=	$\mathbf{Unit} \mid \mathbf{Bool} \mid !_\alpha A \mid \mathbf{Circ}_\alpha(S,U) \mid P \otimes R$
Simple Types	S, U	::=	$\mathbf{Unit} \mid W \mid S \otimes U$
Terms	M, N	::=	$c \mid x \mid \ell \mid \mathsf{unit} \mid \lambda x.M \mid MN \mid (M,N) \mid \mathsf{let}\ (x,y) = N\ \mathsf{in}\ M$
			$\mid \mathsf{lift}\ M \mid \mathsf{force} M$
			$\mid \mathsf{circ}(S,C,U) \mid \mathsf{apply}(M,N) \mid \mathsf{box}_U M$
			$\mid \mathsf{reverse} M \mid \mathsf{control}_K M \mid \mathsf{withComputed} M N$
Simple Terms	a, b	::=	$\ell \mid \mathsf{unit} \mid (a,b)$
Contexts	Γ	::=	$\cdot \mid x : A, \Gamma \mid \ell : W, \Gamma$
Parameter contexts	Φ	::=	$\cdot \mid x : P, \Phi.$
Label Contexts	Σ	::=	$\cdot \mid \ell : W, \Sigma$
Values	V	::=	$c \mid x \mid \ell \mid \mathsf{unit} \mid \lambda x.M \mid (V, V') \mid \mathsf{lift}\ M \mid \mathsf{circ}(S,C,U)$
Circuits	C, D	$\in$	$\mathbf{N}([\![S]\!],[\![U]\!]) \mid \mathbf{R}([\![S]\!],[\![U]\!]) \mid \mathbf{M}([\![S]\!],[\![U]\!])$

Figure 1: The syntax of Proto-Quipper-C

4.1 Syntax

The syntax of Proto-Quipper-C is shown in Figure 1. It is similar to the one specified in [8], except for the highlighted terms and types.

Simple types. The letter W ranges over a set of *wire types*, which in typical applications will be types that represent a single wire in a circuit, such as **Qubit** and **Bit**. Each wire type represents a particular object of the category **M**, and tensors of wire types are called *simple types*. We write $[\![S]\!]$ to denote the object corresponding to S in the category **M**, **N**, or **R** (note that they all have the same objects).

Labels. We also assume that ℓ ranges over a set of *labels*, which are distinct from variables and are used as pointers to places in a circuit where a gate can be attached. Unlike variables, labels cannot be substituted, and they can only have wire types. A *simple term* is essentially a tuple of labels. We call a typing context that contains only labels a *label context* (denoted by Σ). If $\Sigma = \ell_1 : W_1, \ldots, \ell_n : W_n$, we write $[\![\Sigma]\!]$ for the object corresponding to Σ in **M**, **N**, or **R**, i.e., $[\![W_1]\!] \otimes \ldots \otimes [\![W_n]\!]$.

Parameter types. A certain subset of the types are called *parameter types*. These are the types whose values can be freely duplicated and discarded, whereas values of simple types are resources. Also, the values of parameter types are computed at circuit generation time, whereas the values of simple types are merely tuples of labels, which are placeholders for values that will be computed at circuit execution time. A typing context that contains only parameter types is a *parameter context* (denoted by Φ). We have included **Bool** as a typical example of a parameter type, but in a real language, one would of course have many more such types (such as **Nat**, **Int**, sum types, etc.) For space reasons, we omit a comprehensive treatment of sum types, including booleans.

Terms. As usual, x ranges over a set of *variables*. The symbol c is used for constant symbols of the language, such as True and False for the type **Bool**, as well as other appropriate constants. The terms of the lambda calculus are fairly standard. Lift and force are from linear lambda calculus, and are used to construct and deconstruct a term of type $!_\alpha A$. We will discuss the rest of the terms later.

Modalities. The symbols α and β range over modalities, which we take to be numbers in the lattice $2 > 1 > 0$. Here, $\alpha = 2$ indicates that a circuit is reversible and controllable; $\alpha = 1$ indicates that a circuit

$$\frac{}{\Phi, x : A \vdash_2 x : A}\ \mathit{var} \qquad \frac{}{\Phi, \ell : W \vdash_2 \ell : W}\ \mathit{label} \qquad \frac{}{\Phi \vdash_2 \mathsf{unit} : \mathbf{Unit}}\ \mathit{unit}$$

$$\frac{\Gamma, x : A \vdash_\alpha M : B}{\Gamma \vdash_2 \lambda x.M : A \multimap_\alpha B}\ \mathit{lambda} \qquad \frac{\Phi, \Gamma_1 \vdash_{\alpha_1} M : A \multimap_\beta B \quad \Phi, \Gamma_2 \vdash_{\alpha_2} N : A}{\Phi, \Gamma_1, \Gamma_2 \vdash_{\alpha_1 \wedge \alpha_2 \wedge \beta} MN : B}\ \mathit{app}$$

$$\frac{\Phi \vdash_\alpha M : A}{\Phi \vdash_2 \mathsf{lift} M : !_\alpha A}\ \mathit{lift} \qquad \frac{\Phi, \Gamma_1 \vdash_{\alpha_1} M : A \quad \Phi, \Gamma_2 \vdash_{\alpha_2} N : B}{\Phi, \Gamma_1, \Gamma_2 \vdash_{\alpha_1 \wedge \alpha_2} (M, N) : A \otimes B}\ \mathit{pair}$$

$$\frac{\Gamma \vdash_\beta M : !_\alpha A}{\Gamma \vdash_{\alpha \wedge \beta} \mathsf{force}\, M : A}\ \mathit{force} \qquad \frac{\Phi, \Gamma_1, x : A, y : B \vdash_{\alpha_1} M : C \quad \Phi, \Gamma_2 \vdash_{\alpha_2} N : A \otimes B}{\Phi, \Gamma_1, \Gamma_2 \vdash_{\alpha_1 \wedge \alpha_2} \mathsf{let}\ (x, y) = N\ \mathsf{in}\ M : C}\ \mathit{let}$$

$$\frac{C \in \mathbf{D}^\alpha([\![S]\!], [\![U]\!])}{\Phi \vdash_2 \mathsf{circ}(S, C, U) : \mathbf{Circ}_\alpha(S, U)}\ \mathit{circ} \qquad \frac{\Phi, \Gamma_1 \vdash_\alpha M : \mathbf{Circ}_\gamma(S, U) \quad \Phi, \Gamma_2 \vdash_\beta N : S}{\Phi, \Gamma_1, \Gamma_2 \vdash_{\alpha \wedge \beta \wedge \gamma} \mathsf{apply}(M, N) : U}\ \mathit{apply}$$

$$\frac{\Gamma \vdash_\alpha M : !_\beta (S \multimap_\gamma U)}{\Gamma \vdash_\alpha \mathsf{box}_S\, M : \mathbf{Circ}_{\beta \wedge \gamma}(S, U)}\ \mathit{box} \qquad \frac{\Gamma \vdash_\alpha M : \mathbf{Circ}_2(S, S)}{\Gamma \vdash_\alpha \mathsf{control}_K M : \mathbf{Circ}_2(\underline{K} \otimes S, \underline{K} \otimes S)}\ \mathit{ctrl}$$

$$\frac{\Gamma \vdash_\alpha M : \mathbf{Circ}_\gamma(S, U) \quad \gamma > 0}{\Gamma \vdash_\alpha \mathsf{reverse} M : \mathbf{Circ}_\gamma(U, S)}\ \mathit{rev} \qquad \frac{\begin{array}{c}\Phi, \Gamma_1 \vdash_\alpha M : \mathbf{Circ}_\gamma(U, S) \quad \gamma > 0 \\ \Phi, \Gamma_2 \vdash_\beta N : \mathbf{Circ}_2(S, S)\end{array}}{\Phi, \Gamma_1, \Gamma_2 \vdash_{\alpha \wedge \beta} \mathsf{withComputed}\, MN : \mathbf{Circ}_2(U, U)}\ \mathit{wc}$$

Figure 2: The typing rules

is reversible but not necessarily controllable, and $\alpha = 0$ indicates a general circuit that may be neither reversible nor controllable. The modality in the type $!_\alpha A$ means that when its value is forced, it may append a gate that has modality α to the then-current circuit. Similarly, the modality in the type $A \multimap_\alpha B$ indicates that when its value is applied to an argument, it may append a gate that has modality α.

Circuits. Since Proto-Quipper-C is a circuit description language, we must have some terms in the language that represent circuits. We write such terms as $\mathsf{circ}(S, C, U)$, where $C : [\![S]\!] \to [\![U]\!]$ is a morphism in the appropriate circuit category **M**, **R**, or **N**. Note that in this context, category theory is used not as a denotational semantics (that will be done in Section 6), but as part of the *syntax* of the language. This is effectively the same as adding a constant symbol for every possible circuit. We write $\mathbf{Circ}_\alpha(S, U)$ for the type of quantum circuits with inputs S and outputs U (which are simple types), subject to the modality α as described above. The terms apply and box are used for constructing and deconstructing circuits; they will be explained in more detail in the section on the operational semantics. We include the terms $\mathsf{reverse} M$, $\mathsf{control}_K M$, and $\mathsf{withComputed}\, MN$ for reversing, control and the with-computed operation. The subscript K ranges over the objects of the category **C** which is part of the given structure of the control category **N**. We moreover assume that $\underline{K}$ is a simple type (rather than just an arbitrary object).

In a practical implementation, programmers will not usually write values of the form $\mathsf{circ}(S, C, U)$ directly. Instead, basic quantum gates such as the Hadamard gate will usually be predefined in a library, and users will write programs to combine these into larger circuits.

4.2 Typing rules

The typing rules are shown in Figure 2. Typing judgments are of the form $\Gamma \vdash_\alpha M : A$. Here, the modality α asserts that during the evaluation of the term M, gates of modality α may be appended to the current

circuit. Recall that we write $\alpha \wedge \beta$ to denote the greatest lower bound, or $\min(\alpha, \beta)$. To facilitate the statement of the typing rules, we treat the contexts as unordered lists.

Except for the modalities, the typing rules are similar to the ones in [8]. We make the following observations about the modalities. Since a value cannot append any gates, it does not change the current circuit state. Therefore values always have modality 2, i.e., $\Gamma \vdash_2 V : A$. The *lambda* and *lift* rules store the modality α of M in the respective types $A \multimap_\alpha B$ and $!_\alpha A$. Similarly, in the typing rule *box*, the modalities β, γ jointly determine the modality of the circuit type. Conversely, in the rules *app*, *force*, and *apply*, the modality in the types $A \multimap_\alpha B$, $!_\alpha A$, and $\mathbf{Circ}_\gamma(S,U)$ affects the modality of the current term. In the rule *circ*, we write $\mathbf{D}^\alpha$ to mean $\mathbf{M}$ if $\alpha = 0$, and $\mathbf{R}$ if $\alpha = 1$, and $\mathbf{N}$ if $\alpha = 2$. The *rev* rule requires the circuit to be reversible, i.e., γ must be at least 1. Similarly, the rule *ctrl* requires a controllable circuit, i.e., modality 2. It also requires the circuit to have matching input and output types, i.e., $\mathbf{Circ}_2(S,S)$ rather than $\mathbf{Circ}_2(S,U)$. Finally, the typing rule for with-computed requires the term N to be a controllable circuit, whereas the term M only needs to be reversible. The resulting term $\mathsf{withComputed}\, M\, N$ is again a controllable circuit.

5 Operational semantics

In this section, we give an operational semantics of our language. We follow the usual paradigm of keeping type checking separate from evaluation, i.e., after type checking succeeds, the evaluation is done on untyped terms. Accordingly, in Section 5.1, we provide evaluation rules for untyped terms (which could potentially lead to run-time errors). In Section 5.2, we prove that well-typed terms do not cause runtime errors.

5.1 Evaluation rules

In this section, we will define a big-step, call-by-value operational semantics for Proto-Quipper-C. Like the syntax and the type system, the operational semantics is parameterized by the triple of categories $\mathbf{M}$, $\mathbf{R}$, and $\mathbf{N}$. The operational semantics is defined in Figure 3. It is defined on configurations that are triples (C, Σ, M), where M is a term, Σ is a label context, and $C : X \to [\![\Sigma]\!]$ is a morphism in $\mathbf{M}$, $\mathbf{R}$ or $\mathbf{N}$. We call the pair (C, Σ) the *circuit state* of the configuration. The evaluation of a closed program M begins with the empty circuit state, i.e., $(\mathrm{id}_I, \emptyset)$.

The *app*, *force*, *let* and *pair* rules are standard. They do not directly modify the underlying circuit state.

The *box* and *apply* rules use some notations that warrant explaining. If Σ is a label context, a a simple term, and S a simple type, we write $\Sigma \vdash a : S$ instead of $\Sigma \vdash_2 a : S$. In this case, Σ and S determine each other uniquely given a. We write $\mathrm{ungen}(a, \Sigma)$ for the unique S such that $\Sigma \vdash a : S$. Conversely, we write $\mathrm{gen}(S) = (a, \Sigma)$ to indicate the generation of a fresh simple term a and a corresponding label context Σ such that $\Sigma \vdash a : S$. There is an obvious interpretation $[\![a]\!] : [\![\Sigma]\!] \to [\![S]\!]$ as an isomorphism in $\mathbf{M}$, $\mathbf{R}$, or $\mathbf{N}$.

In the *box* rule, the codomain of D is $[\![\Sigma''']\!]$ and by the definition of ungen, we have $\Sigma''' \vdash b : U$. Then $[\![S]\!] \xrightarrow{D} [\![\Sigma''']\!] \xrightarrow{[\![b]\!]} [\![U]\!]$ is a morphism in $\mathbf{M}$, $\mathbf{R}$ or $\mathbf{N}$.

In the *rev* and *ctrl* rules, the morphism D is reversed/controlled using the reverse/control function from the category it belongs to. If D is not a morphism in $\mathbf{R}$ or $\mathbf{N}$, respectively, then it will cause a runtime error. The type preservation property (Theorem 5.2) will ensure that there are no such runtime errors. In the *wc* rule, we again abuse the notation; if $D_1 \in \mathbf{N}$, we apply the functor $G : \mathbf{N} \to \mathbf{R}$ before the "$\bullet$" operation. It is a runtime error if $D_1 \in \mathbf{M}$ or $D_2 \notin \mathbf{N}$.

$$\frac{(C_1,\Sigma_1,M)\Downarrow(C_2,\Sigma_2,\lambda x.M') \quad (C_2,\Sigma_2,N)\Downarrow(C_3,\Sigma_3,V) \quad (C_3,\Sigma_3,[V/x]M')\Downarrow(C_4,\Sigma_4,V')}{(C_1,\Sigma_1,MN)\Downarrow(C_4,\Sigma_4,V')}\ app$$

$$\frac{(C,\Sigma,M)\Downarrow(C',\Sigma',\mathsf{lift}\ M') \quad (C',\Sigma',M')\Downarrow(C'',\Sigma'',V)}{(C,\Sigma,\mathsf{force}\ M)\Downarrow(C'',\Sigma'',V)}\ force$$

$$\frac{(C,\Sigma,N)\Downarrow(C',\Sigma',(V_1,V_2)) \quad (C',\Sigma',[V_1/x,V_2/y]M)\Downarrow(C'',\Sigma'',V)}{(C,\Sigma,\mathsf{let}\ (x,y)=N\ \mathsf{in}\ M)\Downarrow(C'',\Sigma'',V)}\ let$$

$$\frac{(C,\Sigma,M)\Downarrow(C',\Sigma',V_1) \quad (C',\Sigma',N)\Downarrow(C'',\Sigma'',V_2)}{(C,\Sigma,(M,N))\Downarrow(C'',\Sigma'',(V_1,V_2))}\ pair$$

$$\frac{(C,\Sigma,M)\Downarrow(C',\Sigma',\mathsf{lift}\ M') \quad \mathrm{gen}(S)=(a,\Sigma'') \quad (\llbracket a\rrbracket^\dagger,\Sigma'',M'\ a)\Downarrow(D,\Sigma''',b) \quad \mathrm{ungen}(b,\Sigma''')=U}{(C,\Sigma,\mathsf{box}_S\ M)\Downarrow(C',\Sigma',\mathsf{circ}(S,\llbracket b\rrbracket\circ D,U))}\ box$$

$$\frac{(C_1,\Sigma_1,M)\Downarrow(C_2,\Sigma_2,\mathsf{circ}(S,D,U)) \quad (C_2,\Sigma_2,N)\Downarrow(C_3,\Sigma_3,a) \quad \mathrm{gen}(U)=(b,\Sigma) \quad (C',\Sigma')=\mathrm{append}(C_3,\Sigma_3,a,D,b,\Sigma)}{(C_1,\Sigma_1,\mathsf{apply}(M,N))\Downarrow(C',\Sigma',b)}\ apply$$

$$\frac{(C,\Sigma,M)\Downarrow(C',\Sigma',\mathsf{circ}(S,D,U))}{(C,\Sigma,\mathsf{reverse}\,M)\Downarrow(C',\Sigma',\mathsf{circ}(U,D^\dagger,S))}\ rev$$

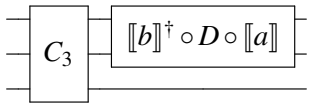
$$\frac{(C,\Sigma,M)\Downarrow(C',\Sigma',\mathsf{circ}(S,D,U))}{(C,\Sigma,\mathsf{control}_K M)\Downarrow(C',\Sigma',\mathsf{circ}(\underline{K}\otimes S,\mathsf{ctrl}_K D,\underline{K}\otimes S))}\ ctrl$$

$$\frac{(C',\Sigma',M)\Downarrow(C'',\Sigma'',\mathsf{circ}(S,D_1,U)) \quad (C,\Sigma,N)\Downarrow(C',\Sigma',\mathsf{circ}(U,D_2,U))}{(C,\Sigma,\mathsf{withComputed}\,MN)\Downarrow(C'',\Sigma'',\mathsf{circ}(S,D_1\bullet D_2,S))}\ wc$$

Figure 3: The operational semantics

In the *apply* rule, by the definition of configurations, we have $C_3 : X \to \llbracket\Sigma_3\rrbracket$, for some object X. We also have $D : \llbracket S\rrbracket \to \llbracket U\rrbracket$. Moreover, $\mathrm{gen}(U) = (b,\Sigma)$ implies that $\Sigma \vdash b : U$ and $\llbracket b\rrbracket : \llbracket\Sigma\rrbracket \to \llbracket U\rrbracket$. Let Σ_3' be the unique label context such that $\Sigma_3' \vdash a : S$. We assert that Σ_3 can be written in the form $\Sigma_3 = \Sigma_3', \Sigma_3''$ (it is a runtime error if the assertion fails). We have $\llbracket a\rrbracket : \llbracket\Sigma_3'\rrbracket \to \llbracket S\rrbracket$. Note that $(\llbracket b\rrbracket^\dagger \circ D \circ \llbracket a\rrbracket) : \llbracket\Sigma_3'\rrbracket \to \llbracket\Sigma\rrbracket$. Let $\Sigma' = \Sigma, \Sigma_3''$, and let $C' : X \to \llbracket\Sigma\rrbracket$ be the following composition:

$$C' = X \xrightarrow{C_3} \llbracket\Sigma_3\rrbracket \xrightarrow{\cong} \llbracket\Sigma_3'\rrbracket \otimes \llbracket\Sigma_3''\rrbracket \xrightarrow{(\llbracket b\rrbracket^\dagger\circ D\circ\llbracket a\rrbracket)\otimes\mathrm{id}} \llbracket\Sigma\rrbracket \otimes \llbracket\Sigma_3''\rrbracket \xrightarrow{\cong} \llbracket\Sigma'\rrbracket \quad =$$



We write $(C',\Sigma') = \mathrm{append}(C_3,\Sigma_3,a,D,b,\Sigma)$ for this operation. With a slight abuse of notation, this composition is always possible, even when the morphisms belong to different categories. For example, if $D \in \mathbf{R}$ and $C_3 \in \mathbf{M}$, we can apply the functor $I : \mathbf{R} \to \mathbf{M}$ to D before the composition.

Example. Since the evaluation rules of Figure 3 are a bit complex, we give an example illustrating their use. Consider the configuration

$$(\mathrm{id}_I,\ \emptyset,\ \mathsf{box}_S\,\mathsf{lift}(\lambda x.\mathbf{let}(a_1,a_2)=x\ \mathbf{in}\ \mathsf{apply}(\mathsf{circ}(S,\mathsf{CNOT},S),(a_2,a_1)))),$$

where $S = \mathbf{Qubit}\otimes\mathbf{Qubit}$ and $\mathsf{CNOT} : \mathbf{Qubit}\otimes\mathbf{Qubit} \to \mathbf{Qubit}\otimes\mathbf{Qubit}$ is a morphism in $\mathbf{N}$. Using the *box* rule, we first call $\mathrm{gen}(S)$, which returns a pair of fresh labels (ℓ_1,ℓ_2) and the label context $\Sigma = \ell_1 : \mathbf{Qubit}, \ell_2 : \mathbf{Qubit}$. Next, using the rules *app* and *let*, we evaluate

$$(\llbracket(\ell_1,\ell_2)\rrbracket^\dagger,\ \Sigma,\ (\lambda x.\mathbf{let}(a_1,a_2)=x\ \mathbf{in}\ \mathsf{apply}(\mathsf{circ}(S,\mathsf{CNOT},S),(a_2,a_1))))(\ell_1,\ell_2))$$

to

$$((\llbracket(\ell_1,\ell_2)\rrbracket^\dagger,\ \Sigma,\ \mathsf{apply}(\mathsf{circ}(S,\mathsf{CNOT},S),(\ell_2,\ell_1))).$$

Then by the *apply* rule, it is evaluated to

$$(\llbracket(\ell_3,\ell_4)\rrbracket^\dagger \circ \mathsf{CNOT} \circ \llbracket(\ell_2,\ell_1)\rrbracket \circ \llbracket(\ell_1,\ell_2)\rrbracket^\dagger,\ \Sigma',\ (\ell_3,\ell_4)),$$

where $\mathrm{gen}(S)$ returns fresh labels (ℓ_3,ℓ_4) and the label context $\Sigma' = \ell_3 : \mathbf{Qubit}, \ell_4 : \mathbf{Qubit}$. Therefore, continuing with the *box* rule, we obtain

$$(\mathrm{id}_I,\ \emptyset,\ \mathsf{circ}(S, \llbracket(\ell_3,\ell_4)\rrbracket \circ \llbracket(\ell_3,\ell_4)\rrbracket^\dagger \circ \mathsf{CNOT} \circ \llbracket(\ell_2,\ell_1)\rrbracket \circ \llbracket(\ell_1,\ell_2)\rrbracket^\dagger, S).$$

Note that $\llbracket(\ell_3,\ell_4)\rrbracket \circ \llbracket(\ell_3,\ell_4)\rrbracket^\dagger = \mathrm{id}_S$ and $\llbracket(\ell_2,\ell_1)\rrbracket \circ \llbracket(\ell_1,\ell_2)\rrbracket^\dagger = \gamma : \mathbf{Qubit} \otimes \mathbf{Qubit} \to \mathbf{Qubit} \otimes \mathbf{Qubit}$, where the morphism γ comes from the symmetric monoidal structure on $\mathbf{N}$. Thus, the final configuration is

$$(\mathrm{id}_I,\ \emptyset,\ \mathsf{circ}(S, \mathsf{CNOT} \circ \gamma, S)).$$

Note that the effect of the morphism γ is to switch two wires without inserting an explicit swap gate (it is, for example, the same as the interpretation of the function `f` in Section 2). If we later control the circuit $\mathsf{circ}(S, \mathsf{CNOT} \circ \gamma, S)$, this morphism γ will be replaced by a controlled swap gate, just like circuit (b) in Section 2.

5.2 Type preservation

In order to formulate type preservation, we first define a notion of *well-typed configuration*.

Definition 5.1 (Well-typed configuration)**.** We define a well-typed configuration $S \vdash_\alpha (C,\Sigma,M) : A;\Sigma'$ to mean: There exist β, γ, Σ'' such that $C \in \mathbf{D}^\beta(\llbracket S\rrbracket, \llbracket\Sigma\rrbracket)$, $\Sigma = (\Sigma'',\Sigma')$, $\Sigma'' \vdash_\gamma M : A$ and $\alpha = \beta \wedge \gamma$.

Theorem 5.2 (Type preservation)**.** *Suppose* $S \vdash_\alpha (C_1,\Sigma_1,M) : A;\Sigma'$ *and* $(C_1,\Sigma_1,M) \Downarrow (C_2,\Sigma_2,V)$. *Then* $S \vdash_\alpha (C_2,\Sigma_2,V) : A;\Sigma'$.

Proof. The proof is by induction on the derivation of $(C_1,\Sigma_1,M) \Downarrow (C_2,\Sigma_2,V)$. □

6 A categorical semantics for reversing and control

6.1 The abstract model

As before, we assume that the categories $\mathbf{M}$, $\mathbf{R}$, and $\mathbf{N}$ are given. These three categories lack the necessary structures to interpret a programming language. Instead, we will interpret the typing rules in a category $\mathbf{A}$, whose structure we now describe.

Definition 6.1. A category $\mathbf{A}$ is a *model for Proto-Quipper-C* if it has the following structure.

- (**a**) $\mathbf{A}$ is symmetric monoidal closed. We write $\varepsilon : (A \multimap B) \otimes A \to B$ for the application map.
- (**b**) $\mathbf{A}$ has coproducts. Note that the tensor distributes over coproducts, because $- \otimes A$ is a left adjoint.
- (**c**) There is an adjunction $p \dashv \flat : \mathbf{Set} \to \mathbf{A}$ where p is a strong monoidal functor.
- (**d**) $\mathbf{A}$ is equipped with idempotent commutative strong monads T_0 and T_1 such that $T_0T_1 \cong T_0 \cong T_1T_0$. More specifically, we require the natural maps $\eta_{T_0B} : T_0B \to T_1T_0B$ and $T_0\eta_B : T_0B \to T_0T_1B$ to be isomorphisms. By idempotent monad we mean that $\mu_B : T^2B \to TB$ is an isomorphism. We write $s : TA \otimes B \to T(A \otimes B)$ for the strength.

(**e**) There are full and faithful embeddings $\mathbf{N} \hookrightarrow \mathbf{A}$, $\mathbf{R} \hookrightarrow Kl_{T_1}(\mathbf{A})$, and $\mathbf{M} \hookrightarrow Kl_{T_0}(\mathbf{A})$. These embedding functors are strong monoidal. Moreover, the following diagram commutes for all $S, U \in \mathbf{M}$.

$$\begin{array}{ccc} \mathbf{N}(S,U) & \xrightarrow{\cong} & \mathbf{A}(S,U) \\ \downarrow G & & \downarrow E \\ \mathbf{R}(S,U) & \xrightarrow{\cong} & Kl_{T_1}(\mathbf{A})(S,U) \\ \downarrow I & & \downarrow L \\ \mathbf{M}(S,U) & \xrightarrow{\cong} & Kl_{T_0}(\mathbf{A})(S,U), \end{array}$$

where E and L are the canonical identity-on-objects functors. Specifically, for any $f \in \mathbf{A}(A,B)$, we have $E(f) = \eta^{T_1} \circ f \in Kl_{T_1}(\mathbf{A})(A,B)$, and for any $f : A \to T_1 B \in Kl_{T_1}(\mathbf{A})$, we have $L(f) = A \xrightarrow{f} T_1 B \xrightarrow{\eta^{T_0}} T_0 T_1 B \xrightarrow{(T_0 \eta_B)^{-1}} T_0 B$.

All models of Proto-Quipper support conditions (a)-(c). Because $p : \mathbf{Set} \to \mathbf{A}$ is a left adjoint and is strong monoidal, we can deduce that $p(X) \cong p(\sum_{x \in X} 1) \cong \sum_{x \in X} p(1) \cong \sum_{x \in X} I$. Due to the adjunction $p \dashv \flat$, we also have $\mathbf{Set}(X, \flat B) \cong \mathbf{A}(p(X), B) \cong \mathbf{A}(\sum_{x \in X} I, B) \cong \prod_{x \in X} \mathbf{A}(I,B) \cong \mathbf{Set}(X, \mathbf{A}(I,B))$. Therefore by Yoneda's principle, we have $\flat(B) \cong \mathbf{A}(I,B)$.

For convenience, we define $T_2 = \mathrm{id}$ to be the identity monad on $\mathbf{A}$. Then in condition (d), the monads T_0, T_1, and T_2 represent (via their Kleisli categories) general circuits, reversible circuits, and controllable circuits, respectively. Recall that we write $\mathbf{D}^\alpha$ for the categories $\mathbf{M}$, $\mathbf{R}$, and $\mathbf{N}$ when $\alpha = 0$, 1, and 2, respectively. Condition (e) ensures that morphisms of simple types in the Kleisli category $Kl_{T_\alpha}(\mathbf{A})$ correspond to morphisms in $\mathbf{D}^\alpha$.

For any $S, U \in \mathrm{obj}(\mathbf{M})$, the maps unbox and box are defined by

$$\mathsf{unbox} : \mathbf{D}^\beta(S,U) \xrightarrow{\cong} \mathbf{A}(S, T_\beta U) \xrightarrow{\text{curry}} \mathbf{A}(I, S \multimap T_\beta U) \xrightarrow{\cong} \flat(S \multimap T_\beta U).$$
$$\mathsf{box} : \flat T_\alpha(S \multimap T_\beta U) \xrightarrow{\cong} \mathbf{Set}(1, \flat T_\alpha(S \multimap T_\beta U)) \xrightarrow{\cong} \mathbf{A}(I, T_\alpha(S \multimap T_\beta U)) \xrightarrow{k} \mathbf{A}(S, T_{\alpha \wedge \beta} U) \xrightarrow{\cong} \mathbf{D}^{\alpha \wedge \beta}(S,U).$$

Here, $k : \mathbf{A}(I, T_\alpha(S \multimap T_\beta U)) \to \mathbf{A}(S, T_{\alpha \wedge \beta} U)$ is given by

$$\mathbf{A}(I, T_\alpha(S \multimap T_\beta U)) \to \mathbf{A}(I, S \multimap T_\alpha T_\beta U) \to \mathbf{A}(I, S \multimap T_{\alpha \wedge \beta} U) \to \mathbf{A}(S, T_{\alpha \wedge \beta} U),$$

where the first map comes from the strength. Note that box is the inverse of unbox when $\alpha = 2$.

Given $\mathbf{M}, \mathbf{R}$ and $\mathbf{N}$, one can construct a concrete category $\mathbf{A}$ satisfying the require properties, using a generalization of our earlier work on biset enrichment [8]. For space reasons, we did not include the construction here, but it can be found in the longer (and slightly outdated) version of this paper at [9].

6.2 Interpretation

We will interpret the typing rules in the category $\mathbf{A}$. The following is the interpretation for types.

$$\begin{array}{lcl} [\![\mathbf{Qubit}]\!] & = & \mathbf{Qubit} \\ [\![A \otimes B]\!] & = & [\![A]\!] \otimes [\![B]\!] \\ [\![\mathbf{Circ}_\alpha(S,U)]\!] & = & p\mathbf{D}^\alpha([\![S]\!], [\![U]\!]) \\ [\![!_\alpha A]\!] & = & p\flat T_\alpha [\![A]\!] \\ [\![A \multimap_\alpha B]\!] & = & [\![A]\!] \multimap T_\alpha [\![B]\!] \end{array}$$

We interpret a typing context Γ as the tensor of all of its objects (denoted by $[\![\Gamma]\!]$). Each valid typing judgment $\Gamma \vdash_\alpha M : A$ will be interpreted as a morphism $[\![M]\!] : [\![\Gamma]\!] \to T_\alpha [\![A]\!]$ in $\mathbf{A}$. The interpretation is defined by induction on the typing rules. See Appendix B for the details.

6.3 Soundness of operational semantics

We now prove that the operational semantics is sound with respect to the categorical model **A**. To do so, we first interpret a well-typed configuration as a morphism in **A**.

Definition 6.2. Suppose $S \vdash_\alpha (C, \Sigma, M) : A; \Sigma'$ is a well-typed configuration, with $C \in \mathbf{D}^\beta(\llbracket S \rrbracket, \llbracket \Sigma \rrbracket)$, $\Sigma = (\Sigma'', \Sigma')$, $\Sigma'' \vdash_\gamma M : A$ and $\alpha = \beta \wedge \gamma$. We define $\llbracket (C, \Sigma, M) \rrbracket : \llbracket S \rrbracket \to T_\alpha(\llbracket A \rrbracket \otimes \llbracket \Sigma' \rrbracket)$ as follows:

$$\llbracket S \rrbracket \xrightarrow{C} T_\beta(\llbracket \Sigma \rrbracket) \xrightarrow{\cong} T_\beta(\llbracket \Sigma'' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_\beta(\llbracket M \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_\beta(T_\gamma \llbracket A \rrbracket \otimes \llbracket \Sigma' \rrbracket) \to T_\alpha(\llbracket A \rrbracket \otimes \llbracket \Sigma' \rrbracket),$$

where the last step uses the strength and the natural isomorphism $T_\beta T_\gamma \cong T_\alpha$.

Theorem 6.3 (Soundness of the evaluation)**.** *If* $S \vdash_\alpha (C, \Sigma, M) : A; \Sigma'$ *and* $(C, \Sigma, M) \Downarrow (C', \Sigma', V)$*, then* $\llbracket (C, \Sigma, M) \rrbracket = \llbracket (C', \Sigma', V) \rrbracket$.

Proof. See Appendix C. □

We remark that our semantics also satisfies computational adequacy, but in our setting, it is a trivial consequence of soundness. This is because if V is a value of type $\mathbf{Circ}(S, U)$, then V is by definition a circuit, i.e., a morphism of the appropriate category **M**, **R**, or **N**. Its semantics is essentially itself. It immediately follows that $\llbracket V \rrbracket = \llbracket V' \rrbracket$ implies $V = V'$. Since our language is also terminating, adequacy for any closed term of type $\mathbf{Circ}(S, U)$ then follows from soundness.

7 Conclusion

In this paper, we showed how to extend Proto-Quipper with reversing, control, and the with-computed operation. Our language is parameterized by three categories **M**, **R**, and **N**, which correspond to general quantum circuits, reversible circuits, and controllable circuits, respectively. We defined a type system that uses modalities to distinguish the different types of circuits. We provided an operational and a denotational semantics for the language; the latter takes the form of an abstract categorical model in which our modalities are represented by monads. We proved that the operational semantics is sound with respect to the categorical model.

There are many possible directions for future work. For example, in this paper, we only considered the modalities of reversibility and controllability. But it seems that our construction of the type system and its categorical semantics would easily generalize from a three element set to an arbitrary poset of modalities. Introducing additional modalities might be useful for characterizing additional properties of gates. Although we have made a prototype implementation that includes a type inference algorithm, we did not formally study its properties. Another challenge for future work is to combine modalities with dependent types. Although our software implementation does support both modalities and dependent types, we have not considered a formal semantics for combining them.

Acknowledgements

We thank the referees for their thoughtful comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Air Force Office of Scientific Research under Award No. FA9550-21-1-0041.

References

[1] Charles H Bennett (1973): *Logical reversibility of computation*. *IBM Journal of Research and Development* 17(6), pp. 525–532, doi:10.1147/rd.176.0525.

[2] Benjamin Bichsel, Maximilian Baader, Timon Gehr & Martin Vechev (2020): *Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics*. In: *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, Association for Computing Machinery, New York, NY, USA, pp. 286–300, doi:10.1145/3385412.3386007.

[3] Andrea Colledan & Ugo Dal Lago (2022): *On dynamic lifting and effect typing in circuit description languages (extended version)*. Available from arXiv:2202.07636.

[4] Thomas G Draper (2000): *Addition on a quantum computer*. Available from arXiv:quant-ph/0008033.

[5] Peng Fu (2025): *Implementation of Proto-Quipper*. Available from https://gitlab.com/frank-peng-fu/dpq-remake.

[6] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2020): *A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper*. In: *Proceedings of the 12th International Conference on Reversible Computation, RC 2020, Oslo, Norway*, *Lecture Notes in Computer Science* 12227, Springer, pp. 153–168, doi:10.1007/978-3-030-52482-1_9. Also available from arXiv:2005.08396.

[7] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2023): *A biset-enriched categorical model for Proto-Quipper with dynamic lifting*. In: *Proceedings of the 19th International Conference on Quantum Physics and Logic, QPL 2022, Oxford*, *Electronic Proceedings in Theoretical Computer Science* 394, pp. 302–342, doi:10.4204/EPTCS.394.16.

[8] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2023): *Proto-Quipper with dynamic lifting*. In: *Proceedings of the 50th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2023, Boston*, *Proceedings of the ACM on Programming Languages* 7 (POPL), pp. 309–334, doi:10.1145/3571204. Also available from arXiv:2204.13041.

[9] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2024): *Proto-Quipper with reversing and control, version 1*. Available from arXiv:2410.22261v1.

[10] Peng Fu, Kohei Kishida & Peter Selinger (2020): *Linear dependent type theory for quantum programming languages*. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020, Saarbrücken, Germany*, pp. 440–453, doi:10.1145/3373718.3394765. Also available from arXiv:2004.13472.

[11] Google (Accessed October 9, 2024): *Cirq Software Reference*. https://quantumai.google/reference/python/cirq/ControlledOperation.

[12] Alexander Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *An introduction to quantum programming in Quipper*. In: *Proceedings of the 5th International Conference on Reversible Computation, RC 2013, Victoria, British Columbia*, *Lecture Notes in Computer Science* 7948, Springer, pp. 110–124, doi:10.1007/978-3-642-38986-3_10. Also available from arXiv:1304.5485.

[13] Alexander Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: a scalable quantum programming language*. In: *Proceedings of the 34th Annual ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2013, Seattle*, *ACM SIGPLAN Notices* 48(6), pp. 333–342, doi:10.1145/2499370.2462177. Also available from arXiv:1304.3390.

[14] Thomas Häner, Damian S Steiger, Krysta Svore & Matthias Troyer (2018): *A software methodology for compiling quantum programs*. *Quantum Science and Technology* 3(2), p. 020501, doi:10.1088/2058-9565/aaa5cc. Also available from arXiv:1604.01401.

[15] IBM (Accessed October 9, 2024): *IBM Quantum Documentation, QuantumCircuit class*. https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit#control.

[16] Dongho Lee, Valentin Perrelle, Benoît Valiron & Zhaowei Xu (2021): *Concrete categorical model of a quantum circuit description language with measurement*. In Mikolaj Bojanczyk & Chandra Chekuri, editors:

41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, *LIPIcs* 213, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 51:1–51:20, doi:10.4230/LIPIcs.FSTTCS.2021.51. Also available from arXiv:2110.02691.

[17] Bert Lindenhovius, Michael Mislove & Vladimir Zamdzhiev (2018): *Enriching a linear/non-linear lambda calculus: A programming language for string diagrams*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, pp. 659–668, doi:10.1145/3209108.3209196. Also available from arXiv:1804.09822.

[18] Jennifer Paykin, Robert Rand & Steve Zdancewic (2017): *QWIRE: a core language for quantum circuits*. In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, *ACM SIGPLAN Notices* 52, ACM, pp. 846–858, doi:10.1145/3009837.3009894.

[19] Francisco Rios & Peter Selinger (2018): *A categorical model for a quantum circuit description language. Extended abstract*. In: *Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen*, *Electronic Proceedings in Theoretical Computer Science* 266, pp. 164–178, doi:10.4204/EPTCS.266.11.

[20] Neil J. Ross (2015): *Algebraic and logical methods in quantum computation*. Ph.D. thesis, Dalhousie University, Department of Mathematics and Statistics. Available from arXiv:1510.02198.

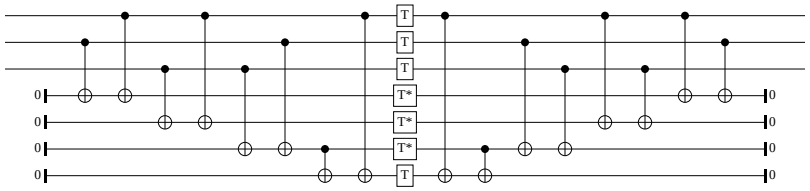
[21] Peter Selinger (2007): *Dagger Compact Closed Categories and Completely Positive Maps*. In: *Proceedings of the 3rd International Workshop on Quantum Programming Languages, QPL 2005, Chicago*, *Electronic Notes in Theoretical Computer Science* 170, Elsevier, pp. 139–163, doi:10.1016/j.entcs.2006.12.018.

[22] Peter Selinger (2013): *Quantum Circuits of T-Depth One*. *Physical Review A* 87, p. 042302 (4 pages), doi:10.1103/PhysRevA.87.042302. Also available from arXiv:1210.0974.

[23] Damian S Steiger, Thomas Häner & Matthias Troyer (2018): *ProjectQ: an open source software framework for quantum computing*. *Quantum* 2, p. 49, doi:10.22331/q-2018-01-31-49. Also available from arXiv:1612.08091.

A Controlling a CCZ gate

Here, we give a basic example of using control and with-computed in the prototype implementation of Proto-Quipper from [5]. It is well-known that a CCZ gate can be implemented by 7 T-gates with T-depth one [22]. See the following circuit.



We can define the above circuit in Proto-Quipper, using its `withComputed` operator.

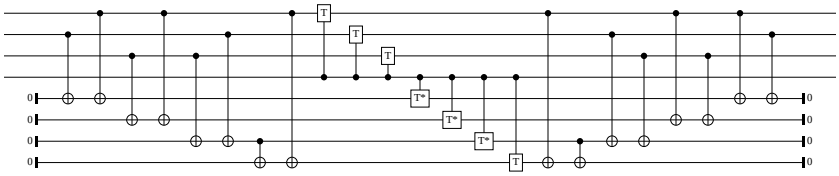
```
my_ccz : Circ(Qubit * Qubit * Qubit, Qubit * Qubit * Qubit)
my_ccz = withComputed box_cnot_circuit box_parallel_T
```

The function `box_parallel_T` generates the 7 parallel T gates in the middle of the circuit and the function `box_cnot_circuit` generates the initialization gates and the cnot circuits before the parallel T gates. Their definitions are available in `examples/CCZ.dpq` in [5].

We can use Proto-Quipper's `control` operator to add an an extra control to the CCZ circuit.

```
ctrl_ccz : Circ(Qubit * (Qubit * Qubit * Qubit), Qubit * (Qubit * Qubit * Qubit))
ctrl_ccz = control my_ccz
```

This generates the following circuit. We can see that only the T-gates in the middle are controlled.



If instead, we program the CCZ circuit without using the `withComputed` operation, we will get an error when trying to control it. In the program below, `my_ccz'` is defined by manual composition, where `cnot_circuit_rev` is the reverse version of `cnot_circuit`.

```
cnot_circuit_rev :
  !(Qubit * Qubit * Qubit * Qubit * Qubit * Qubit * Qubit
    -> Qubit * Qubit * Qubit)
cnot_circuit_rev = unbox (reverse (boxCirc cnot_circuit))

my_ccz' : Circ(Qubit * Qubit * Qubit, Qubit * Qubit * Qubit)
my_ccz' =
  boxCirc $ \input -> cnot_circuit_rev (parallel_T (cnot_circuit input))

-- The following gives rise to a typing error.
ctrl_my_ccz' : Circ(Qubit * (Qubit * Qubit * Qubit), Qubit * (Qubit * Qubit * Qubit))
ctrl_my_ccz' = control my_ccz'
```

Controlling the circuit `my_ccz'` gives rise to a typing error, because according to the semantics of `control`, we would have to control all the gates in the circuit `my_ccz'`, which includes the non-controllable initialization gates. So in order to control the CCZ circuit that has initialization gates, we must construct the circuit using the `withComputed` operation.

B The categorical interpretation of the typing rules

Here, we only give some important cases. The remaining typing rules can be interpreted similarly.

- Case
$$\frac{\Gamma \vdash_\alpha M : !_{\alpha_1}(S \multimap_{\alpha_2} U)}{\Gamma \vdash_\alpha \mathsf{box}_S M : \mathbf{Circ}_{\alpha_1 \wedge \alpha_2}(S,U)}$$
We define $[\![\mathsf{box}_S M]\!]$ to be
$$[\![\Gamma]\!] \xrightarrow{[\![M]\!]} T_\alpha p \flat T_{\alpha_1}([\![S]\!] \multimap T_{\alpha_2}[\![U]\!]) \xrightarrow{T_\alpha p \mathsf{box}} T_\alpha p \mathbf{D}^{\alpha_1 \wedge \alpha_2}([\![S]\!],[\![U]\!]).$$

- Case
$$\frac{C \in \mathbf{D}^\alpha([\![S]\!],[\![U]\!])}{\Phi \vdash_2 \mathsf{circ}(S,C,U) : \mathbf{Circ}_\alpha(S,U)}$$
Since $C : [\![S]\!] \to [\![U]\!]$ is a morphism in $\mathbf{D}^\alpha$, we define $[\![\Phi \vdash_2 \mathsf{circ}(S,C,U) : \mathbf{Circ}_\alpha(S,U)]\!]$ as the following composition, where $1_C : 1 \to \mathbf{D}^\alpha([\![S]\!],[\![U]\!])$ such that $1_C(*) = C$ and $\mathsf{discard} = p!_X, !_X \in \mathbf{Set}(X,1)$.
$$[\![\Phi]\!] \xrightarrow{\mathsf{discard}} p1 \xrightarrow{p1_C} p\mathbf{D}^\alpha([\![S]\!],[\![U]\!]).$$

- Case
$$\frac{\Gamma \vdash_\alpha M : \mathbf{Circ}_2(U,U)}{\Gamma \vdash_\alpha \mathsf{control}_K M : \mathbf{Circ}_2(\underline{K} \otimes U, \underline{K} \otimes U)}$$
By the induction hypothesis, we have $[\![M]\!] : [\![\Gamma]\!] \to T_\alpha p\mathbf{N}([\![S]\!],[\![S]\!])$. Using the function $\mathsf{ctrl}_K : \mathbf{N}([\![U]\!],[\![U]\!]) \to \mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!], [\![\underline{K}]\!] \otimes [\![U]\!])$, we define $[\![\mathsf{control}_K M]\!]$ to be the following.
$$[\![\Gamma]\!] \xrightarrow{[\![M]\!]} T_\alpha p\mathbf{N}([\![U]\!],[\![U]\!]) \xrightarrow{T_\alpha p(\mathsf{ctrl}_K)} T_\alpha p\mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!], [\![\underline{K}]\!] \otimes [\![U]\!])$$

- Case
$$\frac{\begin{array}{l}\Gamma_1 \vdash_\alpha M : \mathbf{Circ}_\gamma(U,S) \quad \gamma > 0 \\ \Gamma_2 \vdash_\beta N : \mathbf{Circ}_2(S,S)\end{array}}{\Gamma_1, \Gamma_2 \vdash_{\alpha \wedge \beta} \mathsf{withComputed}\, MN : \mathbf{Circ}_2(U,U)}$$
Suppose $\alpha = \beta = \gamma = 2$ and $\Gamma_1 \vdash_2 M : \mathbf{Circ}_2(U,S)$. By the induction hypothesis, we have
$$pG_{U,S} \circ [\![M]\!] : [\![\Gamma_1]\!] \to p\mathbf{N}([\![U]\!],[\![S]\!]) \to p\mathbf{R}([\![U]\!],[\![S]\!]),$$
$$[\![N]\!] : [\![\Gamma_2]\!] \to p\mathbf{N}([\![S]\!],[\![S]\!]).$$
Using the function $\mathsf{withComputed} : \mathbf{R}([\![U]\!],[\![S]\!]) \times \mathbf{N}([\![S]\!],[\![S]\!]) \to \mathbf{N}([\![U]\!],[\![U]\!])$, we define

$[\![\mathsf{withComputed}\, MN]\!]$ to be the following.

$$[\![\Gamma_1]\!] \otimes [\![\Gamma_2]\!] \xrightarrow{(pG_{U,S} \circ [\![M]\!]) \otimes [\![N]\!]} p\mathbf{R}([\![U]\!],[\![S]\!]) \otimes p\mathbf{N}([\![S]\!],[\![S]\!])$$

$$\xrightarrow{p(\mathsf{withComputed})} p\mathbf{N}([\![U]\!],[\![U]\!])$$

Suppose $\alpha = \beta = 2$, $\gamma = 1$ and $\Gamma_1 \vdash_2 M : \mathbf{Circ}_1(U,S)$. By the induction hypothesis, we have

$$[\![M]\!] : [\![\Gamma_1]\!] \to p\mathbf{R}([\![U]\!],[\![S]\!]),$$

$$[\![N]\!] : [\![\Gamma_2]\!] \to p\mathbf{N}([\![S]\!],[\![S]\!]).$$

We define $[\![\mathsf{withComputed}\, MN]\!]$ to be the following.

$$[\![\Gamma_1]\!] \otimes [\![\Gamma_2]\!] \xrightarrow{[\![M]\!] \otimes [\![N]\!]} p\mathbf{R}([\![U]\!],[\![S]\!]) \otimes p\mathbf{N}([\![S]\!],[\![S]\!]) \xrightarrow{p(\mathsf{withComputed})} p\mathbf{N}([\![U]\!],[\![U]\!])$$

The remaining cases (e.g. when $\alpha, \beta \neq 2$) are similar.

C Proof of soundness

Theorem C.1 (Soundness of the evaluation)**.** *If* $S \vdash_\alpha (C,\Sigma,M) : A;\Sigma'$ *and* $(C,\Sigma,M) \Downarrow (C',\Sigma',V)$*, then* $[\![(C,\Sigma,M)]\!] = [\![(C',\Sigma',V)]\!]$.

The proof is by induction on $(C,\Sigma,M) \Downarrow (C',\Sigma',V)$. We only give the proof for the rules *box*, *apply*, *ctrl*, and *wc*, as the remaining rules are routine.

- Case

$$\frac{\begin{array}{c}(C,\Sigma,M) \Downarrow (C',\Sigma',\mathsf{lift}\ M') \\ \mathrm{gen}(S) = (a,\Sigma'') \\ ([\![a]\!]^\dagger,\Sigma'',M'\ a) \Downarrow (D,\Sigma''',b) \\ \mathrm{ungen}(b,\Sigma''') = U\end{array}}{(C,\Sigma,\mathsf{box}_S M) \Downarrow (C',\Sigma',\mathsf{circ}(S,[\![b]\!] \circ D,U))}\ box$$

Suppose $S' \vdash_2 (C,\Sigma,\mathsf{box}_S M) : \mathbf{Circ}_0(S,U);\Sigma'_2$. This implies that $C \in \mathbf{N}([\![S]\!],[\![\Sigma'_1]\!] \otimes [\![\Sigma'_2]\!])$ and $\Sigma'_1 \vdash_2 \mathsf{box}_S M : \mathbf{Circ}_0(S,U)$. Thus $\Sigma'_1 \vdash_2 M : !_0(S \multimap_2 U)$, $C' \in \mathbf{N}([\![S]\!],[\![\Sigma'_2]\!])$, and $\vdash_0 M' : S \multimap_2 U$. By the induction hypothesis, $[\![(C,\Sigma,M)]\!] = [\![(C',\Sigma',\mathsf{lift}\, M')]\!]$ and $[\![([\![a]\!]^\dagger,\Sigma'',M'a)]\!] = [\![(D,\Sigma''',b)]\!]$. Thus we have

$$[\![S']\!] \xrightarrow{C} [\![\Sigma'_1]\!] \otimes [\![\Sigma'_2]\!] \xrightarrow{[\![M]\!] \otimes [\![\Sigma'_2]\!]} p\flat T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![\Sigma'_2]\!]$$

$$= [\![S']\!] \xrightarrow{C'} I \otimes [\![\Sigma'_2]\!] \xrightarrow{p\delta [\![M']\!] \otimes [\![\Sigma'_2]\!]} p\flat T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![\Sigma'_2]\!]$$

and

$$I \otimes [\![S]\!] \xrightarrow{I \otimes [\![a]\!]^\dagger} I \otimes [\![\Sigma'']\!] \xrightarrow{[\![M']\!] \otimes [\![a]\!]} T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!]$$
$$\xrightarrow{s} T_0([\![S]\!] \multimap [\![U]\!] \otimes [\![S]\!]) \xrightarrow{T_0 \varepsilon} T_0 [\![U]\!]$$

$=$

$$I \otimes [\![S]\!] \xrightarrow{[\![M']\!]\otimes[\![S]\!]} T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \xrightarrow{s} T_0([\![S]\!] \multimap [\![U]\!] \otimes [\![S]\!]) \xrightarrow{T_0\varepsilon} T_0[\![U]\!]$$

$\overset{(*)}{=}$

$$[\![S]\!] \xrightarrow{D} T_0[\![\Sigma'']\!] \xrightarrow{T_0[\![b]\!]} T_0[\![U]\!].$$

Moreover, $[\![(C,\Sigma,\mathsf{box}_S M)]\!] =$

$$[\![S']\!] \xrightarrow{C} [\![\Sigma_1']\!] \otimes [\![\Sigma_2']\!] \xrightarrow{[\![M]\!]\otimes[\![\Sigma_2']\!]} p\flat T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![\Sigma_2']\!] \xrightarrow{p\mathsf{box}} p\mathbf{M}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2']\!]$$

$=$

$$[\![S']\!] \xrightarrow{C'} I \otimes [\![\Sigma_2']\!] \xrightarrow{p\delta[\![M']\!]\otimes[\![\Sigma_2']\!]} p\flat T_0([\![S]\!] \multimap [\![U]\!]) \otimes [\![\Sigma_2']\!] \xrightarrow{p\mathsf{box}} p\mathbf{M}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2']\!].$$

So we just need to show

$$I \xrightarrow{p\delta[\![M']\!]} p\flat T_0([\![S]\!] \multimap [\![U]\!]) \xrightarrow{p\mathsf{box}} p\mathbf{M}([\![S]\!],[\![U]\!]) = I \xrightarrow{p1_{[\![b]\!]\circ D}} p\mathbf{M}([\![S]\!],[\![U]\!]).$$

In other words, we need to show $[\![b]\!] \circ D = \mathsf{box}(\delta[\![M']\!])$. It suffices to show they are equal in $\mathbf{A}$. This is true by definition of box and $(*)$.

- Case

$$\frac{\begin{array}{c}(C_1,\Sigma_1,M) \Downarrow (C_2,\Sigma_2,\mathsf{circ}(S,D,U)) \\ (C_2,\Sigma_2,N) \Downarrow (C_3,\Sigma_3,a) \\ \mathrm{gen}(U) = (b,\Sigma_4) \\ (C',\Sigma_5) = \mathrm{append}(C_3,\Sigma_3,a,D,b,\Sigma_4)\end{array}}{(C_1,\Sigma_1,\mathsf{apply}(M,N)) \Downarrow (C',\Sigma_5,b)}\ \textit{apply}$$

Suppose $S' \vdash_0 (C_1,\Sigma_1,\mathsf{apply}(M,N)) : U;\Sigma'$, where $\Sigma_1 = (\Sigma'',\Sigma')$, $C_1 \in \mathbf{R}([\![S]\!],[\![\Sigma'']\!] \otimes [\![\Sigma']\!])$, $\Sigma'' \vdash_0 \mathsf{apply}(M,N) : U$, $\Sigma_1'' \vdash_0 M : \mathbf{Circ}_1(S,U)$ and $\Sigma_2'' \vdash_1 N : S$ and $\Sigma'' = (\Sigma_1'',\Sigma_2'')$, $\Sigma_2 = (\Sigma_2'',\Sigma')$, $\Sigma_3 = (\Sigma_2''',\Sigma')$. By the induction hypothesis, we know that $[\![(C_1,\Sigma_1,M)]\!] = [\![(C_2,\Sigma_2,\mathsf{circ}(S,D,U))]\!]$ and $[\![(C_2,\Sigma_2,N)]\!] = [\![(C_3,\Sigma_3,a)]\!]$. Thus

$$\begin{aligned}[\![S']\!] &\xrightarrow{C_1} T_1([\![\Sigma_1'']\!] \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_1([\![M]\!]\otimes[\![\Sigma_2'']\!]\otimes[\![\Sigma']\!])} T_1(T_0 p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_1 s} T_1 T_0(p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!]) \\ &\xrightarrow{\cong} T_0(p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!])\end{aligned}$$

$=$

$$[\![S']\!] \xrightarrow{C_2} T_0([\![\Sigma_2'']\!] \otimes [\![\Sigma']\!]) \xrightarrow{T_0(p1_D\otimes[\![\Sigma_2'']\!]\otimes[\![\Sigma']\!])} T_0(p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!])$$

and

$$\begin{aligned}\llbracket S' \rrbracket &\xrightarrow{C_2} T_0(\llbracket \Sigma_2'' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\llbracket N \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(T_1 \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0 s} T_0 T_1(\llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{\cong} T_0(\llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket)\end{aligned}$$

=

$$\llbracket S' \rrbracket \xrightarrow{C_3} T_0(\llbracket \Sigma_2''' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\llbracket a \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket).$$

We want to show that $\llbracket (C_1, \Sigma_1, \mathsf{apply}(M,N)) \rrbracket = \llbracket (C', \Sigma_5, b) \rrbracket$, where $(C', \Sigma_5) = \mathrm{append}(C_3, \Sigma_3, a, D, b, \Sigma_4)$ is the following morphism in $\mathbf{M}$. So $\Sigma_5 = (\Sigma_4, \Sigma')$.

$$\begin{aligned}\llbracket S' \rrbracket &\xrightarrow{C_3} \llbracket \Sigma_2''' \rrbracket \otimes \llbracket \Sigma' \rrbracket \xrightarrow{\llbracket a \rrbracket \otimes \llbracket \Sigma' \rrbracket} \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket \\ &\xrightarrow{G(D) \otimes \llbracket \Sigma' \rrbracket} \llbracket U \rrbracket \otimes \llbracket \Sigma' \rrbracket \xrightarrow{\llbracket b \rrbracket^\dagger \otimes \llbracket \Sigma' \rrbracket} \llbracket \Sigma_4 \rrbracket \otimes \llbracket \Sigma' \rrbracket.\end{aligned}$$

Since $\mathbf{M} \hookrightarrow Kl_{T_0}(\mathbf{A})$ and $\mathbf{N} \hookrightarrow \mathbf{A}$, the corresponding morphism in $\mathbf{A}$ is

$$\begin{aligned}\llbracket S' \rrbracket &\xrightarrow{C_3} T_0(\llbracket \Sigma_2''' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\llbracket a \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(D \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket U \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\llbracket b \rrbracket^\dagger \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket \Sigma_4 \rrbracket \otimes \llbracket \Sigma' \rrbracket).\end{aligned}$$

We have $RHS =$

$$\llbracket S' \rrbracket \xrightarrow{C_3} T_0(\llbracket \Sigma_2''' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\llbracket a \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(D \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket U \rrbracket \otimes \llbracket \Sigma' \rrbracket)$$

and $LHS =$

$$\begin{aligned}\llbracket S' \rrbracket &\xrightarrow{C_1} T_1(\llbracket \Sigma_1'' \rrbracket \otimes \llbracket \Sigma_2'' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_1(\llbracket M \rrbracket \otimes \llbracket N \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_1(T_0 p\mathbf{N}(\llbracket S \rrbracket, \llbracket U \rrbracket) \otimes T_1 \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_1 s} T_1 T_0(p\mathbf{N}(\llbracket S \rrbracket, \llbracket U \rrbracket) \otimes T_1 \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0 s} T_0 T_1(p\mathbf{N}(\llbracket S \rrbracket, \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(\mathsf{unbox} \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(p\flat(\llbracket S \rrbracket \multimap \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(\mathsf{force} \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0((\llbracket S \rrbracket \multimap \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\varepsilon \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket U \rrbracket \otimes \llbracket \Sigma' \rrbracket)\end{aligned}$$

=

$$\begin{aligned}\llbracket S' \rrbracket &\xrightarrow{C_2} T_0(\llbracket \Sigma_2'' \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(p 1_D \otimes \llbracket N \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(p\mathbf{N}(\llbracket S \rrbracket, \llbracket U \rrbracket) \otimes T_1 \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0 s} T_0 T_1(p\mathbf{N}(\llbracket S \rrbracket, \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(\mathsf{unbox} \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0(p\flat(\llbracket S \rrbracket \multimap \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \\ &\xrightarrow{T_0(\mathsf{force} \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket)} T_0((\llbracket S \rrbracket \multimap \llbracket U \rrbracket) \otimes \llbracket S \rrbracket \otimes \llbracket \Sigma' \rrbracket) \xrightarrow{T_0(\varepsilon \otimes \llbracket \Sigma' \rrbracket)} T_0(\llbracket U \rrbracket \otimes \llbracket \Sigma' \rrbracket)\end{aligned}$$

$=$

$$[\![S']\!] \xrightarrow{C_3} T_0([\![\Sigma_2''']\!] \otimes [\![\Sigma']\!]) \xrightarrow{T_0(p1_D \otimes [\![a]\!] \otimes [\![\Sigma']\!])} T_0(p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![S]\!] \otimes [\![\Sigma']\!])$$
$$\xrightarrow{T_0(\mathsf{unbox} \otimes [\![S]\!] \otimes [\![\Sigma']\!])} T_0(p\flat([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \otimes [\![\Sigma']\!])$$
$$\xrightarrow{T_0(\mathsf{force} \otimes [\![S]\!] \otimes [\![\Sigma']\!])} T_0(([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \otimes [\![\Sigma']\!]) \xrightarrow{T_0(\varepsilon \otimes [\![\Sigma']\!])} T_0([\![U]\!] \otimes [\![\Sigma']\!]).$$

So we just need to show

$$[\![S]\!] \xrightarrow{D} [\![U]\!]$$

$=$

$$I \otimes [\![S]\!] \xrightarrow{1_D} p\mathbf{N}([\![S]\!],[\![U]\!]) \otimes [\![S]\!] \xrightarrow{p\,\mathsf{unbox} \otimes [\![S]\!]} p\flat([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!]$$
$$\xrightarrow{\mathsf{force} \otimes [\![S]\!]} ([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \xrightarrow{\varepsilon} [\![U]\!].$$

This is true because $LHS =$

$$I \otimes [\![S]\!] \xrightarrow{\mathrm{curry}(D) \otimes [\![S]\!]} ([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \xrightarrow{\varepsilon} [\![U]\!]$$

and $RHS =$

$$I \otimes [\![S]\!] \xrightarrow{p(\delta\,\mathrm{curry}(D)) \otimes [\![S]\!]} p\flat([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!]$$
$$\xrightarrow{\mathsf{force} \otimes [\![S]\!]} ([\![S]\!] \multimap [\![U]\!]) \otimes [\![S]\!] \xrightarrow{\varepsilon} [\![U]\!]$$

and we have $\mathsf{force} \circ p\delta = \mathrm{id}$.

- Case

$$\frac{(C,\Sigma,M) \Downarrow (C',\Sigma''',\mathsf{circ}(U,D,U))}{(C,\Sigma,\mathsf{control}_K M) \Downarrow (C',\Sigma''',\mathsf{circ}(\underline{K} \otimes U, \mathsf{ctrl}_K D, \underline{K} \otimes U))}\ ctrl$$

  – Suppose $C : [\![S]\!] \to [\![\Sigma'']\!] \otimes [\![\Sigma']\!] \in \mathbf{N}$, $\Sigma = (\Sigma'',\Sigma')$ and $\Sigma'' \vdash_2 M : \mathbf{Circ}_2(U,U)$. By the induction hypothesis and type preservation (Theorem 5.2), we have

$$[\![(C,\Sigma,M)]\!] = [\![(C',\Sigma',\mathsf{circ}(U,D,U))]\!]$$

  and $S \vdash_2 (C',\Sigma',\mathsf{circ}(U,D,U)) : \mathbf{Circ}_2(U,U) : \Sigma'$, where $\vdash_2 \mathsf{circ}(U,D,U) : \mathbf{Circ}_2(U,U)$ and $\Sigma''' = \Sigma'$. So

$$[\![S]\!] \xrightarrow{C} [\![\Sigma'']\!] \otimes [\![\Sigma']\!] \xrightarrow{[\![M]\!] \otimes [\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]$$
$$= [\![S]\!] \xrightarrow{C'} I \otimes [\![\Sigma']\!] \xrightarrow{p1_D \otimes [\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!].$$

  The above equality implies the following.

$$[\![S]\!] \xrightarrow{C} [\![\Sigma'']\!] \otimes [\![\Sigma']\!] \xrightarrow{[\![M]\!] \otimes [\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]$$
$$\xrightarrow{p(\mathsf{ctrl}_K) \otimes [\![\Sigma']\!]} p\mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!],[\![\underline{K}]\!] \otimes [\![U]\!]) \otimes [\![\Sigma']\!]$$
$$= [\![S]\!] \xrightarrow{C'} I \otimes [\![\Sigma']\!] \xrightarrow{p1_D \otimes [\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]$$
$$\xrightarrow{p(\mathsf{ctrl}_K) \otimes [\![\Sigma']\!]} p\mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!],[\![\underline{K}]\!] \otimes [\![U]\!]) \otimes [\![\Sigma']\!].$$

  Thus $[\![(C,\Sigma,\mathsf{control}_K M)]\!] = [\![(C',\Sigma',\mathsf{circ}(\underline{K} \otimes U, \mathsf{ctrl}_K(D), \underline{K} \otimes U))]\!]$.

– Now suppose $\Sigma = (\Sigma'', \Sigma')$, $C \in \mathbf{M}([\![S]\!], [\![\Sigma'']\!] \otimes [\![\Sigma']\!])$ and $\Sigma'' \vdash_1 M : \mathbf{Circ}_2(U,U)$. By the induction hypothesis, we have $[\![(C,\Sigma,M)]\!] = [\![(C',\Sigma''',\mathsf{circ}(U,D,U))]\!]$, i.e.,

$$\begin{aligned}[\![S]\!] &\xrightarrow{C} T_0([\![\Sigma'']\!] \otimes [\![\Sigma']\!]) \xrightarrow{T_0([\![M]\!]\otimes[\![\Sigma']\!])} T_0(T_1 p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_0 s} T_0(p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]) \\ = [\![S]\!] &\xrightarrow{C'} T_0(I \otimes [\![\Sigma']\!]) \xrightarrow{T_0(p1_D\otimes[\![\Sigma']\!])} T_0(p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]).\end{aligned}$$

Note that $\Sigma''' = \Sigma'$. The above equality implies the following.

$$\begin{aligned}[\![S]\!] &\xrightarrow{C} T_0([\![\Sigma'']\!] \otimes [\![\Sigma']\!]) \xrightarrow{T_0([\![M]\!]\otimes[\![\Sigma']\!])} T_0(T_1 p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_0 s} T_0(p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_0(p(\mathsf{ctrl}_K))\otimes[\![\Sigma']\!])} T_0(p\mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!],[\![\underline{K}]\!] \otimes [\![U]\!]) \otimes [\![\Sigma']\!]) \\ = [\![S]\!] &\xrightarrow{C'} T_0(I \otimes [\![\Sigma']\!]) \xrightarrow{T_0(p1_D\otimes[\![\Sigma']\!])} T_0(p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!]) \\ &\xrightarrow{T_0(p(\mathsf{ctrl}_K)\otimes[\![\Sigma']\!])} T_0(p\mathbf{N}([\![\underline{K}]\!] \otimes [\![U]\!],[\![\underline{K}]\!] \otimes [\![U]\!]) \otimes [\![\Sigma']\!]).\end{aligned}$$

By the naturality of the strength s, we have

$$[\![(C,\Sigma,\mathsf{control}_K M)]\!] = [\![(C',\Sigma''',\mathsf{circ}(\underline{K}\otimes U, \mathsf{ctrl}_K D, \underline{K}\otimes U))]\!].$$

• Case

$$\frac{(C,\Sigma,M) \Downarrow (C',\Sigma_1,\mathsf{circ}(U,D_1,S')) \quad (C',\Sigma_1,N) \Downarrow (C'',\Sigma_2,\mathsf{circ}(S',D_2,S'))}{(C,\Sigma,\mathsf{withComputed}\,M\,N) \Downarrow (C'',\Sigma_2,\mathsf{circ}(U,D_1 \bullet D_2,U))}\ wc$$

– Suppose $\Sigma = (\Sigma_1'', \Sigma_2'', \Sigma')$, $C \in \mathbf{N}([\![S]\!], [\![\Sigma_1'']\!] \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!])$, $\Sigma_1'' \vdash_2 N : \mathbf{Circ}_2(S',S')$ and $\Sigma_2'' \vdash_2 M : \mathbf{Circ}_1(U,S')$. By the induction hypothesis, we have

$$[\![(C,\Sigma,M)]\!] = [\![(C',\Sigma_1,\mathsf{circ}(U,D_1,S'))]\!]$$

and $[\![(C',\Sigma_1,N)]\!] = [\![(C'',\Sigma_2,\mathsf{circ}(S',D_2,S'))]\!]$. Thus we have

$$\begin{aligned}[\![S]\!] &\xrightarrow{C} [\![\Sigma_1'']\!] \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!] \xrightarrow{[\![M]\!]\otimes[\![\Sigma_2'']\!]\otimes[\![\Sigma']\!]} p\mathbf{R}([\![U]\!],[\![S']\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!] \\ = [\![S]\!] &\xrightarrow{C'} I \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!] \xrightarrow{p1_{D_1}\otimes[\![\Sigma_2'']\!]\otimes[\![\Sigma']\!]} p\mathbf{R}([\![U]\!],[\![S']\!]) \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!]\end{aligned}$$

and

$$\begin{aligned}[\![S]\!] &\xrightarrow{C'} [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!] \xrightarrow{[\![N]\!]\otimes[\![\Sigma']\!]} p\mathbf{N}([\![S']\!],[\![S']\!]) \otimes [\![\Sigma']\!] \\ = [\![S]\!] &\xrightarrow{C''} I \otimes [\![\Sigma']\!] \xrightarrow{p1_{D_2}\otimes[\![\Sigma']\!]} p\mathbf{N}([\![S']\!],[\![S']\!]) \otimes [\![\Sigma']\!].\end{aligned}$$

We want to show $[\![(C,\Sigma,\mathsf{withComputed}\,M\,N)]\!] = [\![(C'',\Sigma_2,\mathsf{circ}(U,D_1\bullet D_2,U))]\!]$, i.e.,

$$\begin{aligned}[\![S]\!] &\xrightarrow{C} [\![\Sigma_1'']\!] \otimes [\![\Sigma_2'']\!] \otimes [\![\Sigma']\!] \xrightarrow{[\![M]\!]\otimes[\![N]\!]\otimes[\![\Sigma']\!]} \\ p\mathbf{R}([\![U]\!],[\![S]\!]) \otimes p\mathbf{N}([\![S]\!],[\![S]\!]) \otimes [\![\Sigma']\!] &\xrightarrow{p(\mathsf{withComputed})\otimes[\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!] \\ = [\![S]\!] &\xrightarrow{C''} I \otimes [\![\Sigma']\!] \xrightarrow{p1_{D_1\bullet D_2}\otimes[\![\Sigma']\!]} p\mathbf{N}([\![U]\!],[\![U]\!]) \otimes [\![\Sigma']\!].\end{aligned}$$

This is the case by the induction hypotheses and $\mathsf{withComputed} \circ (1_{D_1} \otimes 1_{D_2}) = 1_{D_1\bullet D_2}$.

A Complete and Natural Rule Set for Multi-Qutrit Clifford Circuits

Sarah Meng Li [*] [†] [‡] [§]
sarah.li@uva.nl

Michele Mosca [†] [‡] [§]
michele.mosca@uwaterloo.ca

Neil J. Ross [¶]
neil.jr.ross@dal.ca

John van de Wetering [*]
john@vdwetering.name

Yuming Zhao [‖] [‡] [**]
yuming@math.ku.dk

We present a complete set of rewrite rules for n-qutrit Clifford circuits where n is any non-negative integer. This is the first completeness result for any fragment of quantum circuits in odd prime dimensions. We first generalize Selinger's normal form for n-qubit Clifford circuits to the qutrit setting. Then, we present a rewrite system by which any Clifford circuit can be reduced to this normal form. We then simplify the rewrite rules in this procedure to a small natural set of rules, giving a clean presentation of the group of qutrit Clifford unitaries in terms of generators and relations.

1 Introduction

Most research on quantum computing has focused on qu*bits*, two-dimensional quantum systems. In recent years, research on qu*dit* quantum computing, where the fundamental state space has a higher dimension than two, has witnessed a surge in interest and activity [10, 14, 15, 16, 26, 30, 32, 35]. Qudits offer richer algebraic structures that result in unique error correction capabilities [8, 33], lower overhead magic state distillation [9, 29], stronger quantum correlations [7, 13], and improvements to various quantum algorithms [6, 27]. On the physical side, most realizations of qubits naturally have higher energy states so they could readily encode qudits. This allows for greater information density, but comes at the cost of increased difficulty in system control [5, 17, 23, 30, 36, 37, 38].

In this paper, we focus on qu*trits*, the three-dimensional quantum systems. Qutrits strike a balance between enabling the benefits of working with higher-dimensional systems, while not being too complicated to prevent hardware implementations. In particular, we study the group of qutrit *Clifford unitaries*, the unitaries that normalize the qutrit analogue of the Pauli group. As for qubits, the qutrit Clifford group plays a crucial role in the stabilizer formalism for ternary quantum systems, serving as the backbone for designing fault-tolerant quantum gates and protocols [2, 3, 12, 15, 18].

Using a small set of generators for the qutrit Clifford group—the Hadamard, S and CZ gates—we find a simple set of relations that suffice to prove all equalities between Clifford circuits. That is, we prove that the rule set in Figure 1 is *complete*. This is the first completeness result for a non-trivial quantum circuit fragment for higher-dimensional qudits. Our proof strategy is generic and is intended to generalize to qudits of higher dimensions.

[*] Informatics Institute, University of Amsterdam, Amsterdam, Netherlands.
[†] Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Canada.
[‡] Institute for Quantum Computing, University of Waterloo, Waterloo, Canada.
[§] Perimeter Institute for Theoretical Physics, Waterloo, Canada.
[¶] Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada.
[‖] Department of Pure Mathematics, University of Waterloo, Waterloo, Canada.
[**] QMATH, Department of Mathematical Sciences, University of Copenhagen, Copenhagen, Denmark.

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 23–78, doi:10.4204/EPTCS.426.2

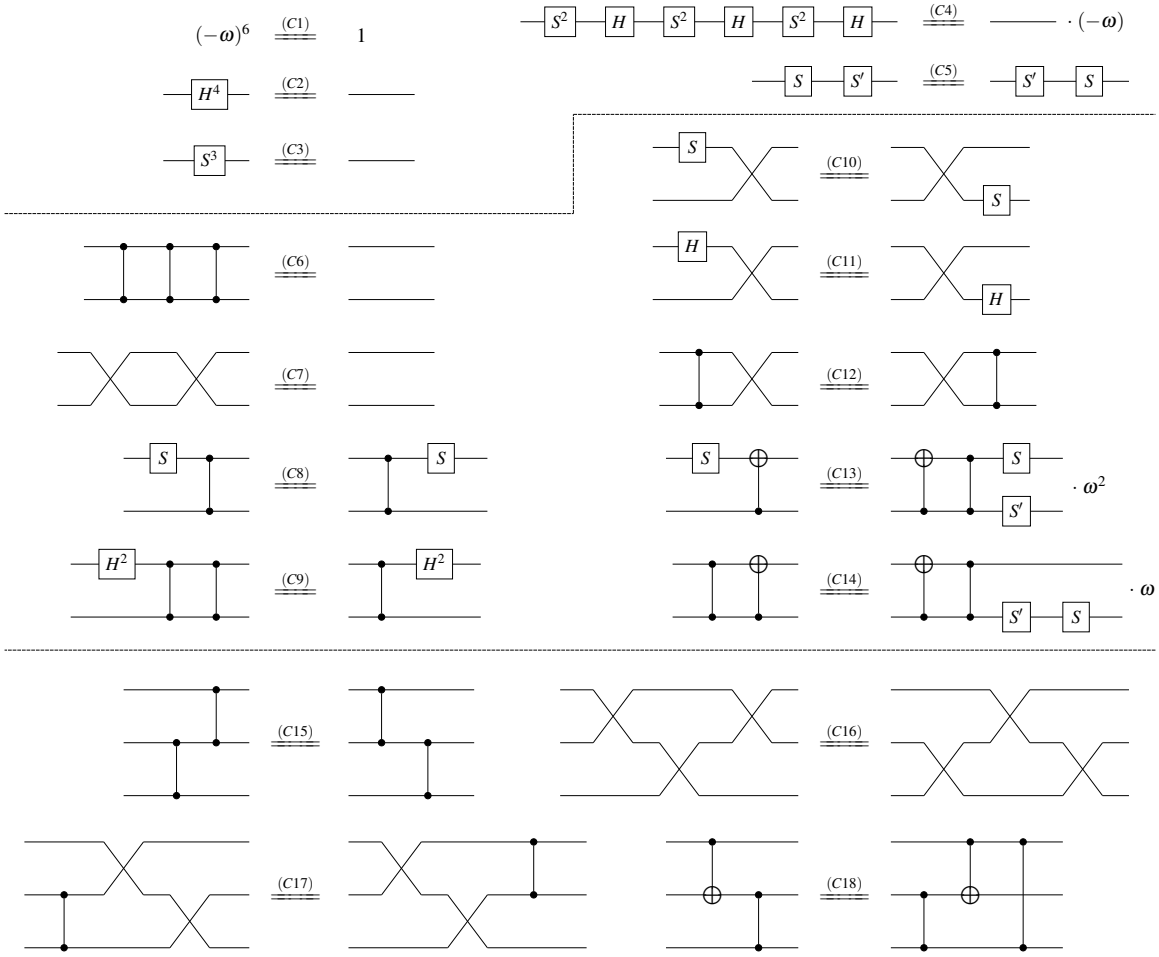


Figure 1: A complete set of rewrite rules for n-qutrit Clifford circuits. Note that ω, S', SWAP, CX, XC, and the CZ acting on the first and third qutrits are derived generators defined in Figure 5. Let g be a gate and m be a natural number. g^m denotes m copies of g gate. In addition to these circuit relations, we assume the standard spatial relations which allow us to commute gates acting on disjoint subsets of qutrits. In the remainder of this paper, we use 'relations' and '(rewrite) rules' interchangeably.

A complete graphical calculus for qutrit linear maps in the stabilizer fragment exists using the ZX-calculus [28, 34]. However, ZX-diagrams represent general linear maps, not just unitaries. Hence, this completeness result does not restrict to the unitary Clifford group in any straightforward way. In general, a completeness result through a graphical calculus for all linear maps of a certain type does not seem to easily imply a method of deriving a calculus for just the unitaries. Consider for instance that a complete ZX-calculus for universal qubit linear maps was found in 2017 [19, 25], but it was not until 2022 that a complete calculus of universal quantum circuits appeared and it used very different methods [11]. Moreover, there is no complete calculus for, for instance, Toffoli-Hadamard circuits, even though there is one for the corresponding set of linear maps [4].

One way to prove completeness for a fragment of quantum circuits (i.e., a collection of unitary operators) is to define a unique normal form for each unitary and to show that every circuit can be rewritten

to this normal form. This is how the completeness result for qubit Clifford circuits was established in [24, 31]. Our proof follows along similar lines, but with significant complications due to the increased system dimension and the additional degrees of freedom. At a high level, we can view the normal form proposed by [31] as encoding the *stabilizer tableau* of a Clifford unitary. Each part of the normal form determines the image of Pauli generators under the action of this Clifford unitary. To prove completeness, we need to show that when a Clifford circuit is in normal form and it is composed with a Clifford generator (i.e., a Hadamard, S, or CZ gate), the resulting circuit can be rewritten into a new normal form. This process, that we call *Clifford normalization*, is sketched in Figure 2. Then our problem is reduced to finding rewrite rules to push Clifford generators through each part of the normal form in a systematic manner.

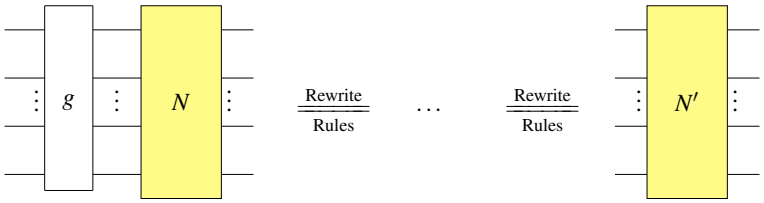


Figure 2: N is a Clifford circuit in normal form. g is either an H or S gate acting on some qutrit, or a CZ gate acting on two adjacent qutrits. After pushing g through each part of N, the normal form is updated to N'.

The problem in generalizing this to qutrits is that the choice of each normal form component, what we call a *normal box*, is not unique, but has some degrees of freedom. Different choices result in different 'dirty gates' appearing when we push a generator through a normal box. Figure 3 shows a concrete example. Making a wrong choice might produce dirty gates that cannot be properly pushed through the subsequent normal boxes. In the worst case, the Clifford normalization will not terminate. Another complication is that there are many more combinations of normal boxes and generators to check due to the increased dimension. All of this means that in the qubit case, the design space was small enough for a brute-force approach to succeed in finding a suitable way to construct the normal boxes, but with qutrits, the significantly larger design space makes such an approach much more challenging.

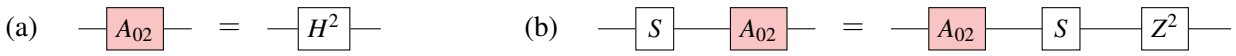


Figure 3: (a) shows a single-qutrit normal box called A_{02}, which in this particular case is equal to H^2. (b) shows a *box relation*, where an S gate is pushed through A_{02} and produces dirty gates S and Z^2. H^2 and Z^2 denote two copies of Hadamard and Pauli Z gates respectively.

Instead, we identify additional symmetries that each normal box must satisfy, which then guarantee that dirty gates have the desired form for most of the box relations. This restricts the design space and allows us to verify on a higher level that our normal boxes have the right properties for the Clifford normalization to terminate (see Appendix E for the details). With the correct definitions of the normal boxes in hand, we can then write down all the relations needed to push the generators through the normal boxes, what we call the *box relations*. In Appendix C, we present the 380 box relations, which together are complete for multi-qutrit Clifford circuits.

The second major component of this work is to reduce the 380 box relations to a much smaller, more

canonical set of rewrite rules. In the supplement [22], we prove that the 18 gate relations presented in Figure 1 imply these 380 box relations. In order to present these rules in the nicest possible fashion, we use some derived generators that are defined in Figure 5.

Overall, Figure 1 consists of natural and intuitive rules regarding the order of generators, commutations of generators, and properties of the SWAP gate. The only exception to this is (C4), which can be understood as describing an Euler decomposition of the Hadamard gate. Up to a global phase, it can be reframed with some rewriting to $H = S^2(HS^2H^{-1})(S')^2$, meaning H can be decomposed into a Z-, X- and then another Z-rotation. Note that the derived generator $S' = H^2SH^2$ in Figure 5 has no qubit counterpart, but it is a Z rotation that we can use together with S to define the Pauli Z gate, $Z = (S')^2S$. See also the discussion in Section 1.1 of [22]. The commutation of S and S' in (C5) is the only single-qutrit rule that has no qubit counterpart.

Generalizing from qutrits to qu*pits*, where each wire carries a p-dimensional quantum system for an odd prime p, we can construct a version of Figure 1 with sound equations for qupit Clifford operators. Although proving that such a generalized rule set is complete still requires substantial work, Figure 1 presents a natural framework for understanding gate interactions. This, in turn, offers a promising path toward extending the qutrit Clifford completeness to other odd prime dimensions.

The rest of the paper is organized as follows. In Section 2, we introduce the basic concepts and conventions that will be used throughout this paper. In Section 3, we define normal forms and prove that every multi-qutrit Clifford circuit admits a unique normal form. In Section 4, we describe how to rewrite any Clifford circuit to this normal form, based on which we find a complete set of rewrite rules for qutrit Clifford circuits. We end with some final remarks and open questions in Section 5.

2 Preliminaries

We are interested in *matrices*, which we sometimes call *operators*, or *gates*. We write quantum circuits from left to right, which is in the opposite order of writing the matrix multiplication.

Let $m \in \mathbb{N}$. We write I_m for the $m \times m$ identity matrix, dropping the subscript m when the dimension is clear from the context. An n-qutrit operator is a $3^n \times 3^n$ matrix. We say that an n-qutrit operator U is a *scalar*, if it is a scalar multiple of the identity operator, i.e., $U = \lambda I$, for some $\lambda \in \mathbb{C}$. For simplicity, we write $U = \lambda$ in such a case. In what follows, we use 'scalar' and 'phase' interchangeably.

2.1 Clifford Groups and Circuits

We fix $\omega = e^{2\pi i/3}$ and $-\omega = e^{-2\pi i/6}$ as our *primitive third* and *sixth roots of unity* respectively.

Definition 2.1. *The single-qutrit* Pauli X and Z gates *are defined as follows*

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{and} \qquad Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{bmatrix}.$$

For any $m \in \mathbb{N}$, we write $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ equipped with the standard addition modulo m.

Definition 2.2. *For $n \in \mathbb{N}$, the* n-qutrit Pauli group $\mathcal{P}_n$ *is defined as*

$$\mathcal{P}_n = \{\omega^c P_1 \otimes \cdots \otimes P_n;\ c \in \mathbb{Z}_3 \text{ and } P_j = X^{a_j}Z^{b_j} \text{ for some } a_j, b_j \in \mathbb{Z}_3\}.$$

We now define the *Clifford group*, which will be the focus of this paper. As Definition 2.3 states, the n-qutrit Clifford group is the normalizer of the Pauli group in the $3^n \times 3^n$ unitary group $\mathcal{U}(3^n)$.

Definition 2.3. *For $n \in \mathbb{N}$, the n-qutrit* Clifford group *$\mathcal{C}_n$ is defined as*

$$\mathcal{C}_n = \{U \in \mathcal{U}(3^n);\ UPU^\dagger \in \mathcal{P}_n \text{ for all } P \in \mathcal{P}_n\}.$$

Definition 2.4. *The single-qutrit H* and *S* gates *and the two-qutrit* CZ gate *are defined as follows*

$$H = \frac{1}{\omega^2 - \omega}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}, \quad S = \begin{bmatrix} \omega & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{and} \quad CZ = diag(1,1,1,1,\omega,\omega^2,1,\omega^2,\omega).$$

It can be verified by direct calculation that $H, S \in \mathcal{C}_1$ and $\mathrm{CZ} \in \mathcal{C}_2$. The H gate is also known as the *Hadamard gate*, while the CZ gate is also known as the *controlled-Z gate*. We chose the global phase in the H and S gates to make our results simpler to state.

Note that Definition 2.3 considers all possible global phases $e^{i\alpha}$ as being Clifford, where $\alpha \in \mathbb{R}$. This makes the group uncountably infinite, but H, S and CZ as matrices have entries in the ring $\mathbb{Z}[1/3, \omega]$ and hence can only represent a subset of these phases. In what follows, we restrict our attention to the subset of Clifford operators that can be represented as matrices over $\mathbb{Z}[1/3, \omega]$. As a result, Appendix B.1 shows that the global phases can only be powers of $-\omega$. In Proposition 3.9, we show that $-\omega$, H, S, and CZ generate $\mathcal{C}_n$.

It is clear from Definition 2.3, that $\mathcal{P}_n \subseteq \mathcal{C}_n$. In fact, Z and X can be expressed using H and S as $Z = SH^2S^2H^2$ and $X = HZH^3$. Note that when we write in text form, a juxtaposition of gates means we do the usual matrix multiplication. Since the matrices of Z, S, and CZ gates are diagonal, we sometimes refer to them as *diagonal gates*.

We can also denote Clifford operators using the diagrammatic language of quantum circuits. Figure 4 shows the circuit representation of the H, S, and CZ gates (as well as that of the identity).

Figure 4: Circuit notation for the one- and two-qutrit Clifford generators.

Next, we introduce *derived* generators that are summarized in Figure 5. They are useful for defining the Clifford normal forms in Section 3, designing the normal boxes in Appendix E, deriving box relations in Appendix C, and carrying out the relation reduction in the supplement [22]. One can think of them as circuit shorthands over the generators $-\omega$, H, S, and CZ gates. -1 (D1) and ω (D2) are used to simplify the expression of box relations. As opposed to the qubit H gate, the qutrit H gate is of order 4, $H^4 = 1$. Up to the global phase -1, H^2 acts as a 'negation', $H^2|x\rangle = |-x\rangle$. Here, the negation is taken modulo 3, so that $|0\rangle \mapsto |0\rangle$, $|1\rangle \mapsto |2\rangle$, and $|2\rangle \mapsto |1\rangle$. We then see that S' (D3) is also a diagonal gate, and as a matrix $S' = \mathrm{diag}(\omega, 1, \omega)$.

CX (D7) is short for a *controlled-NOT gate*. The control and target of a CX gate are on the top and bottom qutrit wires, respectively. Let $x, y \in \mathbb{Z}_3$. $\mathrm{CX}|x,y\rangle = |x, x+y\rangle$, where the addition is taken in $\mathbb{Z}_3$ and hence modulo 3. A XC gate (D8) has an opposite control-target arrangement.

We assume that the multi-qutrit (derived) generators only act on adjacent qutrits and we call them *local gates*. This is without loss of generality, as gates on non-adjacent qutrits can be equivalently expressed using SWAP gates. In Figure 5, we write a gate G acting on two non-adjacent qutrits as a circuit shorthand for a local gate surrounded by SWAP gates. (D9), (D10), and (D11) are called the *remote CZ, CX, and XC gates*.

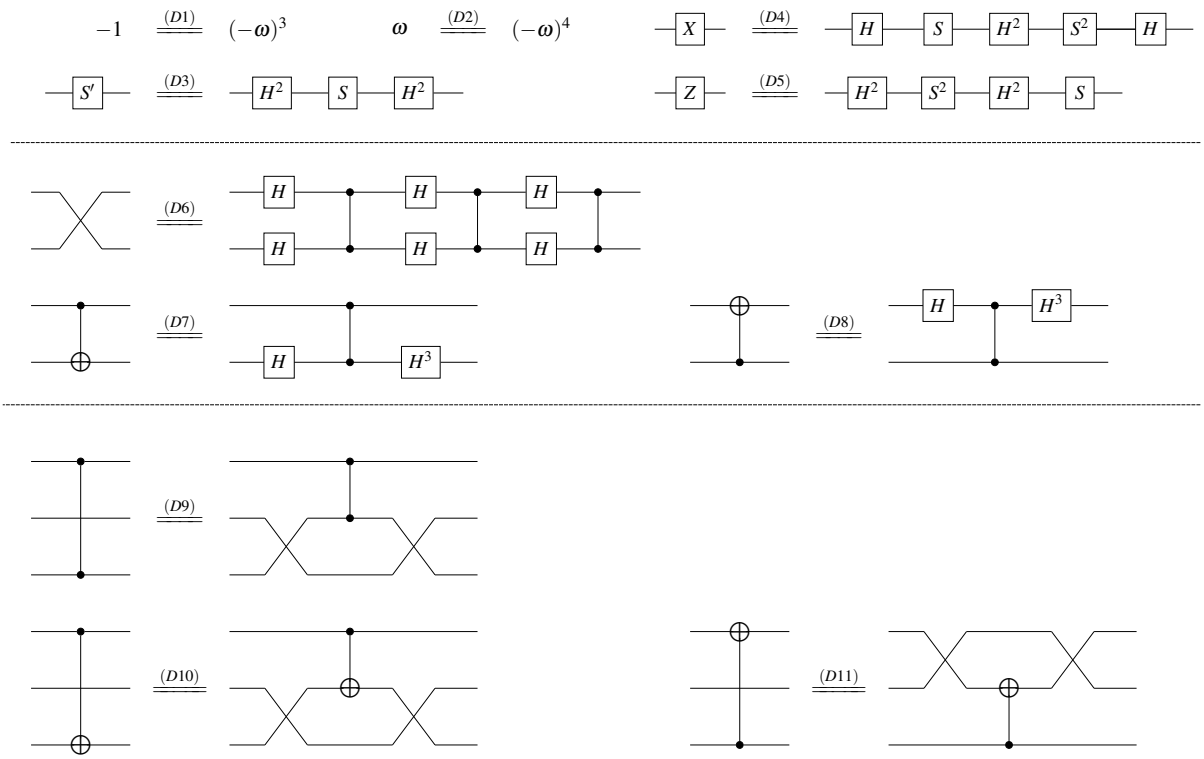


Figure 5: Circuit notation for the derived Clifford generators.

Finally, we introduce the notation conventions used in the rest of this paper. For a circuit $C \in \mathcal{C}_n$, its global phase $\lambda \in \mathbb{C}$ is placed at the end of the circuit, as shown in Figure 6.(a). Let $m \in \mathbb{N}$. In Figure 6.(b), we write C^m to denote the m-fold composition of C with itself.

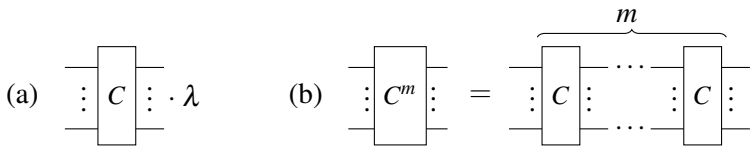


Figure 6: Notation conventions.

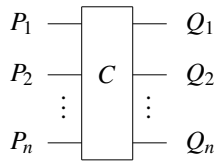
Remark 2.5. *Instead of viewing each $\mathcal{C}_n$ as a separate group, we could consider the collection of all these groups as a single* groupoid*: a category where every morphism is an isomorphism. In this category, the objects are $\mathbb{N}$, corresponding to the number of qutrits. This then has a monoidal structure given by $a \otimes b := a + b$ on objects $a, b \in \mathbb{N}$. In this groupoid, we get the standard coherence isomorphism $(f \otimes id) \circ (id \otimes g) = (id \otimes g) \circ (f \otimes id)$, allowing us to commute arbitrary gates acting on disjoint sets of qutrits. Throughout the paper, we will assume this property without further comment, as was done in [31].*

2.2 Clifford Conjugation on the Pauli Group

By definition, a Clifford operator $C \in \mathcal{C}_n$ maps any Pauli operator $P \in \mathcal{P}_n$ to another Pauli operator, $CPC^\dagger \in \mathcal{P}_n$. This conjugation operation is denoted as $C \bullet P = CPC^\dagger$, and we refer to it as *the Clifford C's*

action on the Pauli operator P. Let $Q = C \bullet P$. Then $CP = QC$, and we can interpret this as 'pushing P through C gives us Q'. It will be useful to introduce a circuit shorthand for this.

Definition 2.6. *Let $C \in \mathcal{C}_n$ and $P, Q \in \mathcal{P}_n$ with $Q = C \bullet P$. Write $P = P_1 \otimes \ldots \otimes P_n$ and $Q = Q_1 \otimes \ldots \otimes Q_n$ for $P_j, Q_j \in \mathcal{P}_1$. Then the conjugation relation $Q = C \bullet P$ is denoted as*



Following this convention, Appendix D summarizes the actions of H, S, X, Z, CZ, and SWAP gates on Pauli operators.

Lemma 2.7. *Let $C \in \mathcal{C}_n$, the action of C on $\mathcal{P}_n$ is a group automorphism of $\mathcal{P}_n$ and it fixes scalars.*

Proof. It is sufficient to show that for all $P, Q \in \mathcal{P}_n$, $C \bullet (PQ) = C(PQ)C^\dagger = (CPC^\dagger)(CQC^\dagger) = (C \bullet P)(C \bullet Q)$. For all $\lambda \in \mathbb{C}$, since $C \bullet \lambda = C\lambda C^\dagger = \lambda CC^\dagger = \lambda$, this automorphism fixes scalars. □

Proposition 2.8. *Let $C \in \mathcal{C}_n$. If $C \bullet P = P$ for all $P \in \mathcal{P}_n$, then C is a scalar (i.e. proportional to the identity).*

Proof. Any complex 3×3 matrix can be written in the form $\sum_{j=1}^{9} c_j P_j$, $c_j \in \mathbb{C}$, $P_j \in \{X^a Z^b;\ a, b \in \mathbb{Z}_3\}$. It follows that $\mathcal{P}_n$ spans the vector space formed by $3^n \times 3^n$ complex matrices. Since $C \bullet P = P$ for all $P \in \mathcal{P}_n$, $C \bullet M = CMC^\dagger = M$ for all operators M. Hence $CM = MC$ means C must commute with all matrices. Then C must be a scalar. □

Corollary 2.9. *Let $C_1, C_2 \in \mathcal{C}_n$. Suppose for all $P \in \mathcal{P}_n$, $C_1 \bullet P = C_2 \bullet P$. Then $C_1 = \lambda C_2$, $\lambda \in \mathbb{C}$.*

Proof. For all $P \in \mathcal{P}_n$, $C_1 P C_1^\dagger = C_2 P C_2^\dagger$. This implies that $C_2^\dagger C_1 P C_1^\dagger C_2 = P$. Take $U = C_2^\dagger C_1$, then $U^\dagger = (C_2^\dagger C_1)^\dagger = C_1^\dagger C_2$. Hence $UPU^\dagger = P$ for all $P \in \mathcal{P}_n$. By Proposition 2.8, $U = \lambda$, for some $\lambda \in \mathbb{C}$. It follows that $C_1 = \lambda C_2$. This completes the proof. □

It is well-known that a Clifford unitary C is fully determined by its action on Pauli operators. Since the Clifford conjugation is a group homomorphism, it suffices to know C acts on a set of *Pauli generators*. In Definition 2.10, we use $\mathcal{B}_n$ to denote a collection of Pauli operators that has a Pauli Z or X acting on a single qutrit, and identities everywhere else. $\mathcal{B}_n$ generates $\mathcal{P}_n$ and it is used in the remainder of this paper.

Definition 2.10. *Let $n \in \mathbb{N}$. $Z^{(j)}$ denotes the n-qutrit Pauli operator $P = P_1 \otimes \cdots \otimes P_n$ with $P_\ell = Z$ if $\ell = j$, and $P_\ell = I$ otherwise. We write $X^{(j)}$ for the analogous n-qutrit Pauli operator by substituting Z with X in the above definition. $\mathcal{B}_n = \{X^{(j)}, Z^{(j)};\ 1 \leq j \leq n\}$.*

Thus, for $1 \leq j \leq n$ and $P^{(j)}, Q^{(j)} \in \mathcal{P}_n$, knowing $P^{(j)} = C \bullet Z^{(j)}$ and $Q^{(j)} = C \bullet X^{(j)}$ fully determines C (up to a global phase). The collection of these $2n$ maps is often called the *stabilizer tableau* of C [1, 21]. This notion will become useful for designing the multi-qutrit Clifford normal form.

When $n = 1$, the stabilizer tableau is a 3×2 matrix over $\mathbb{Z}_3$. Figure 7 shows how to encode a single-qutrit Clifford operator $C = SHSH$ using a tableau. Let $P, Q \in \mathcal{P}_1$ be the images of Z and X in C respectively. They are encoded by the first and second columns of the tableau. The first row encodes the ω-exponent in the phase of P and Q. The second and third rows denote the participation of Z and X in P and Q, respectively. This notation can be generalized to encode an n-qutrit Clifford operator using a $(2n+1) \times 2n$ matrix over $\mathbb{Z}_3$.

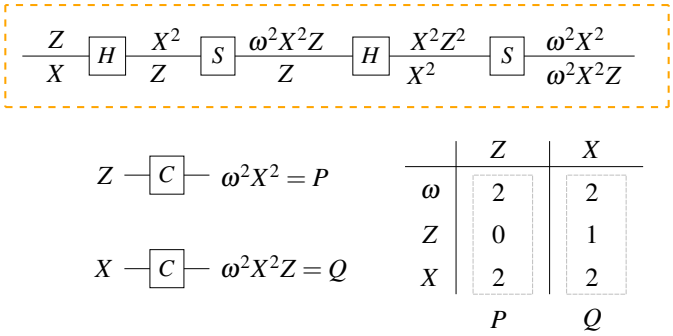


Figure 7: According to Appendix D, we can track the Pauli evolution in $C = SHSH$ and find the images of Z and X. Since Z is mapped to ω^2X^2 and X is mapped to ω^2X^2Z, we can construct a 3×2 stabilizer tableau for C.

Remark 2.11. *By Lemma 2.7, Clifford conjugation on Paulis is a group automorphism. Hence, $P^{(j)}$ and $Q^{(j)}$ inherit the commuting and non-commuting relations from $Z^{(j)}$ and $X^{(j)}$. That is, $P^{(j)}Q^{(j)} = \omega Q^{(j)}P^{(j)}$, and $P^{(j)}Q^{(\ell)} = Q^{(\ell)}P^{(j)}$ when $j \neq \ell$.*

3 A Normal Form for Multi-Qutrit Clifford Circuits

Similar to the qubit Clifford normal form defined in [24, 31], our normal form for a qutrit Clifford circuit $C \in \mathcal{C}_n$ is based on uniquely representing its stabilizer tableau, and hence uniquely representing this operator. First, we provide intuitions for designing such a normal form. Then, we lay out the technical details of constructing this normal form.

Intuitions It will be helpful to think of this normal form as synthesizing the *inverse* of C, and we will refer to this as the *inverse synthesis*. Suppose C^{-1} has a stabilizer tableau, for $1 \leq j \leq n$,

$$P^{(j)} = C^{-1} \bullet Z^{(j)}, \qquad Q^{(j)} = C^{-1} \bullet X^{(j)}. \tag{1}$$

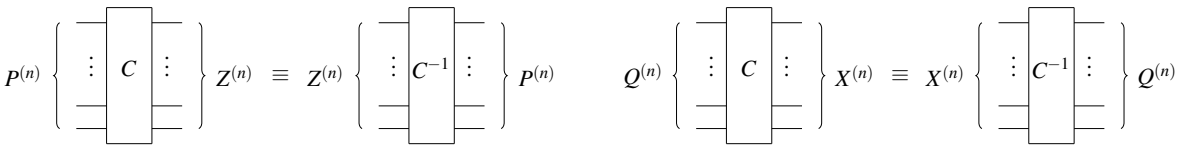
We want to find a Clifford circuit in normal form N such that

$$N \bullet P^{(j)} = Z^{(j)}, \qquad N \bullet Q^{(j)} = X^{(j)}. \tag{2}$$

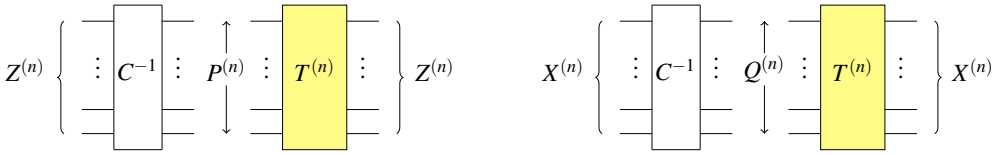
Putting (1) and (2) together, (3) shows that $(NC^{-1}) \bullet P = P$ for all $P \in \mathcal{P}_n$. By Proposition 2.8, $NC^{-1} = \lambda$, for some $\lambda \in \mathbb{C}$. Hence, $N = \lambda C$. Then we find the normal form of C, up to some global phase.

$$N \bullet P^{(j)} = NC^{-1} \bullet Z^{(j)} = Z^{(j)}, \qquad N \bullet Q^{(j)} = NC^{-1} \bullet X^{(j)} = X^{(j)}, \qquad 1 \leq j \leq n. \tag{3}$$

In a nutshell, we carry out the inverse synthesis by induction. Firstly, from (1), we have

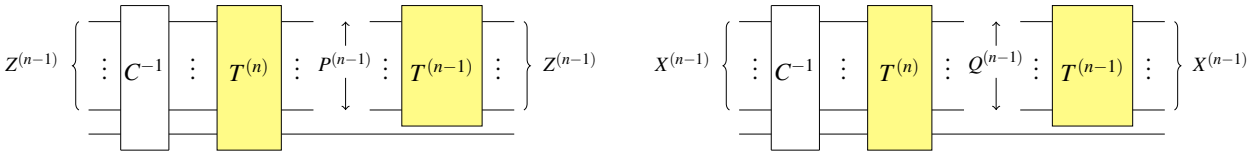


Next, find a circuit $T^{(n)}$ in normal form that sends $P^{(n)}$ to $Z^{(n)}$ and $Q^{(n)}$ to $X^{(n)}$. We then have

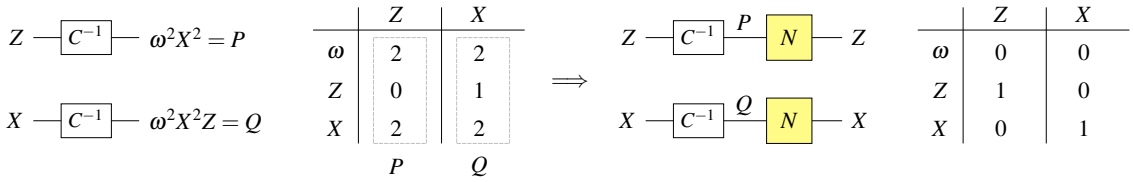


$$\left(T^{(n)}C^{-1}\right)\bullet Z^{(n)} = Z^{(n)} \qquad \left(T^{(n)}C^{-1}\right)\bullet X^{(n)} = X^{(n)}$$

This implies that $T^{(n)}C^{-1}$ acts as an identity on the last qutrit, so we can treat it as a Clifford operator acting on the first $n-1$ qutrits. For this 'smaller' circuit, repeat the same procedure and find a circuit $T^{(n-1)}$ in normal form such that $T^{(n-1)}T^{(n)}C^{-1}$ acts as an identity on the last two qutrits. By iteratively concatenating $T^{(j)}$, from $j=n$ to $j=1$, we obtain $N = T^{(1)}\cdots T^{(n-1)}T^{(n)}$.



One can think of the inverse synthesis as simplifying the stabilizer tableau by performing Gaussian elimination on its bottom $2n \times 2n$ matrix. As a simple example, when $n=1$ and $C = (SHSH)^{-1}$, the inverse synthesis reduces the stabilizer tableau of C^{-1} to an all-zero row followed by an identity matrix.



Technical Details Our goal is to map the Pauli generators to the desired outcomes by using some elementary building blocks, which we call *normal boxes*. To obtain a unique normal form, we need to identify some notion of uniqueness when designing these boxes. In total, there are six *types* of normal boxes. They are labelled as A, B, C, D, E, and F, as shown in Figure 8. Each type of normal box has different *variants*, and each variant is labelled by the subscript of the box.

Definition 3.1. *For $K \in \{A,B,C,D,E,F\}$, we call a normal box of* type *K a K* normal box. *We use K_k to denote a* variant *of the K normal box, where k can be any allowed indices in the first column of Figure 8.*

In the second column of Figure 8, we specify the required action of each normal box on some Paulis. Take B normal boxes as an example. The specification of other normal boxes can be understood analogously. For $a,b \in \mathbb{Z}_3$, B_{ab} maps $X^aZ^b \otimes Z$ to $Z \otimes I$. The uniqueness of these required actions determines the uniqueness of each B_{ab} box. However, this action does not specify a full stabilizer tableau, so it is not sufficient to determine which unitary we should use for each of these boxes. Different choices of implementation might result in different designs of the normal form. Making the 'wrong' choice might make it more difficult to establish the multi-qutrit Clifford completeness.

In the third column of Figure 8, we identify additional actions that some normal boxes should have on Paulis. This takes advantage of the underlying mechanics of our normal form and further restricts the design space. In Appendix E, we discuss in detail how these restrictions come into play.

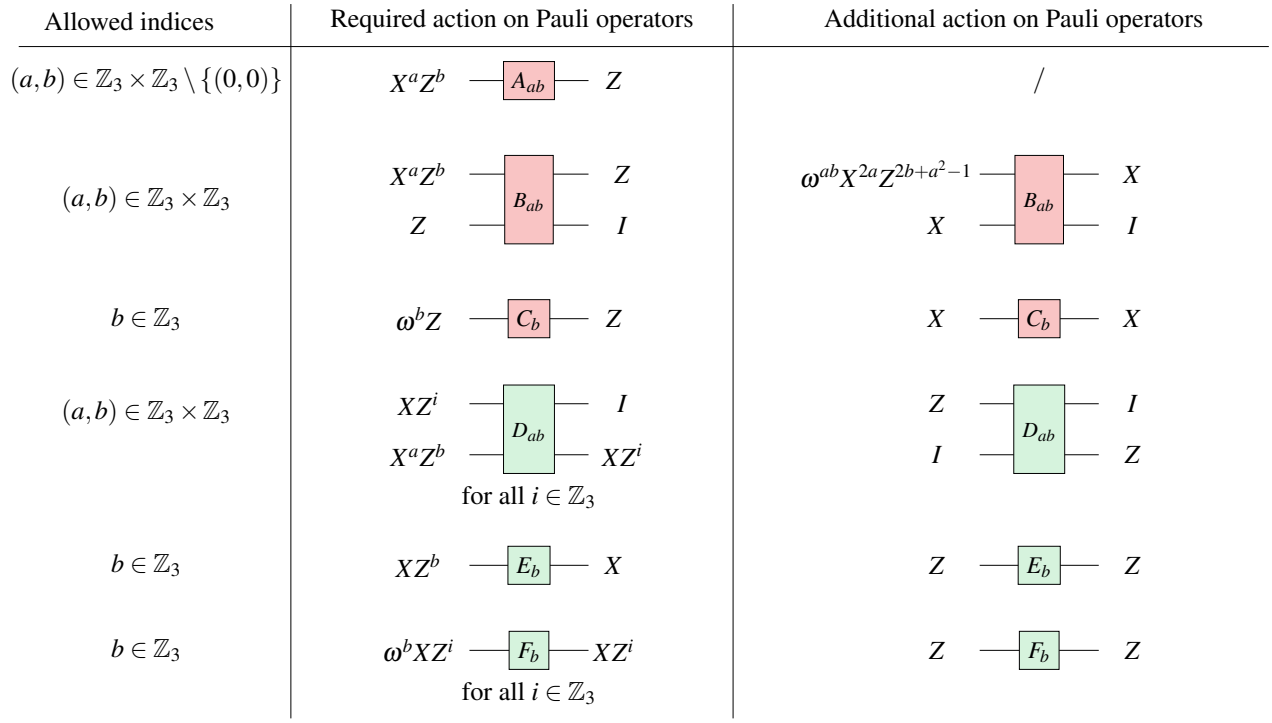


Figure 8: There are six types of normal boxes. The colour of each box (red or green) specifies whether it belongs to the Z-part or the X-part of the normal form.

Figure 9 demonstrates a concrete implementation of each normal box. Most of the constructions here are given parametrically, using the index of each box. This feature plays an important role in simplifying the search for box relations, as discussed in Section 4 and appendices C and E. Also note that the normal box construction uses some circuit shorthands defined in Figure 6. In Appendix A, Figures 15 and 16 illustrate the individual construction of each normal box using H, S, CZ, SWAP, and XC gates. Figure 17 presents the complete stabilizer tableaus for these boxes.

Using these normal boxes, we now define the normal form for n-qutrit Clifford circuits.

Definition 3.2. *For A, B, C, D, E, and F normal boxes defined in Figure 9:*

- *An n-qutrit Clifford circuit is* Z-normal *if it is of the form in Figure 10a.*
- *An n-qutrit Clifford circuit is* X-normal *if it is of the form in Figure 10b.*

Definition 3.3. *An n-qutrit Clifford circuit is* normal *if it is of the form in Figure 11.*

Remark 3.4. *When $n = 1$, a normal form is composed of only A, C, E, and F boxes. A generic construction of $N^{(1)}$ is given in Figure 47.*

Now that we have the normal form, we need to show that it is *universal*, meaning it can represent an arbitrary Clifford circuit, and *unique*, meaning that each Clifford unitary is uniquely represented by one specific instantiation of this normal form. For universality, by Corollary 2.9, it suffices to show that we can represent an arbitrary automorphism of $\mathcal{P}_n$ by a normal form.

Let ϕ be an automorphism of $\mathcal{P}_n$. We claim that one can construct a Z-normal circuit L and an X-normal circuit M such that $(ML)^{-1}$ acts as ϕ^{-1} on $Z^{(n)}$ and $X^{(n)}$. To obtain a normal form for ϕ, it suffices

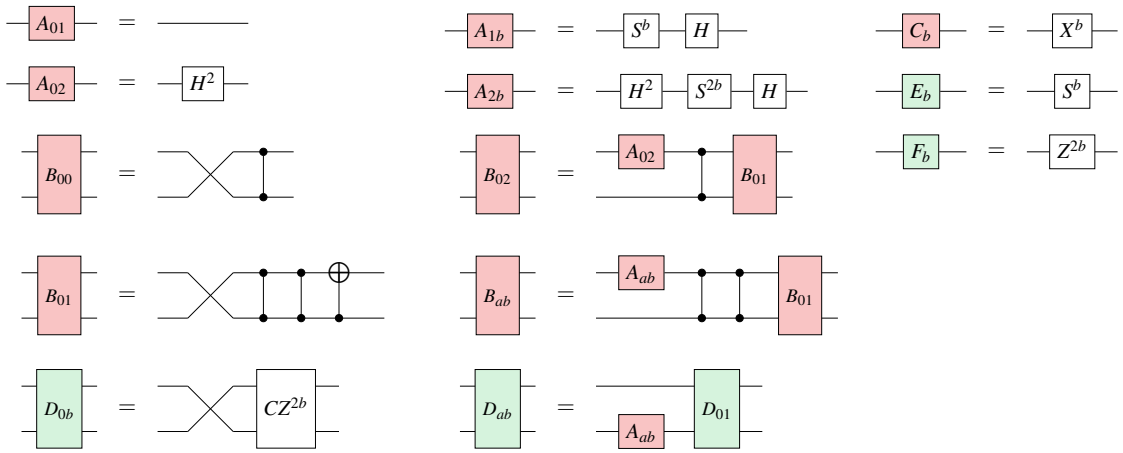
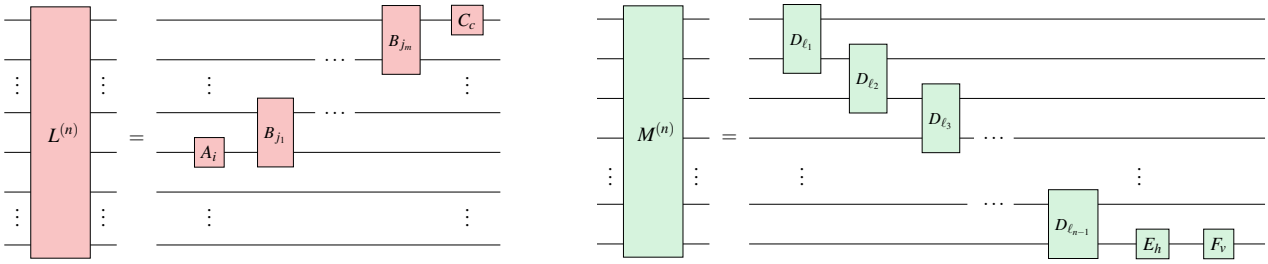


Figure 9: The concrete implementations of the Z and X normal boxes. Here $a, b \in \mathbb{Z}_3$ and $a \neq 0$.



(a) The Z-normal circuit.

(b) The X-normal circuit.

Figure 10: The Z- and X-normal circuits are used to build the normal form for an n-qutrit Clifford circuit.

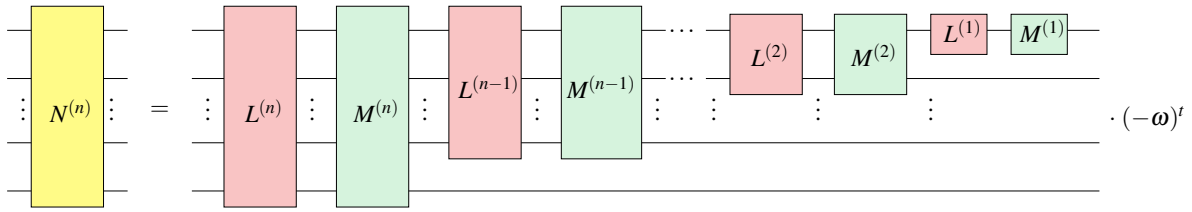


Figure 11: The structure of a normal form: we alternate layers of Z- and X-normal circuits, with each subsequent layer acting on one fewer qutrit. Here $t \in \mathbb{Z}_6$ defines the global phase. Note that we can view this normal form as being defined inductively, with the n-qutrit normal form adding a layer of Z- and X-normal circuits to the left of an existing $(n-1)$-qutrit normal form.

to construct L and M and then to recursively iterate this procedure with the updated automorphism $\phi' = \phi(ML)^{-1}$. The proofs of these statements are analogous to those for the qubit case in [24, 31], so we postpone them to Appendix B.2 (the only substantial difference is that instead of Paulis Z and X anticommuting with each other, they now 'ω-anticommute': $ZX = \omega XZ$).

Lemma 3.5. *For $n \geq 1$, let P be an n-qutrit Pauli operator and $P \notin \{I, \omega I, \omega^2 I\}$. Then there exists a unique Z-normal circuit L such that $L \bullet P = Z \otimes I \otimes \cdots \otimes I$.*

Lemma 3.6. *For $n \geq 1$, let Q be an n-qutrit Pauli operator and $(Z \otimes I \otimes \cdots \otimes I)Q = \omega Q(Z \otimes I \otimes \cdots \otimes I)$.*

Then there exists a unique X-normal circuit M such that $M \bullet Q = I \otimes \cdots \otimes I \otimes X$.

Lemma 3.7. *Every X-normal circuit M satisfies $M \bullet (Z \otimes I \otimes \cdots \otimes I \otimes I) = I \otimes I \otimes \cdots \otimes I \otimes Z$.*

Lemma 3.8. *For $n \geq 1$, let P, Q be n-qutrit Pauli operators such that $PQ = \omega QP$. Then there exist unique circuits M and L, where M is X-normal and L is Z-normal such that $(ML) \bullet P = I \otimes \cdots \otimes I \otimes Z$ and $(ML) \bullet Q = I \otimes \cdots \otimes I \otimes X$.*

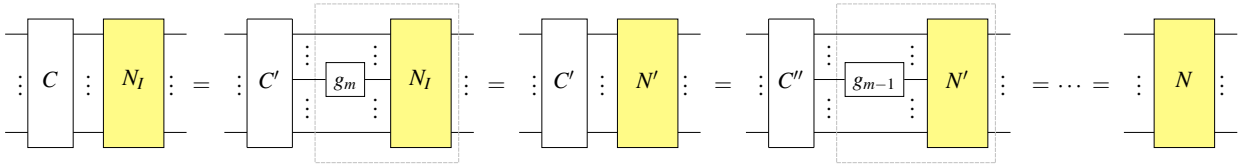
Proposition 3.9. *Let $\phi : \mathcal{P}_n \to \mathcal{P}_n$ be an automorphism of the Pauli group such that ϕ fixes scalars. Then there exists a Clifford circuit C in normal form such that for all $P \in \mathcal{P}_n$, $C \bullet P = \phi(P)$. Moreover, the normal form C is unique up to a scalar $(-\omega)^t$, $t \in \mathbb{Z}_6$.*

By the existence statement of Proposition 3.9, every automorphism of the Pauli group can be represented as a circuit over $-\omega$, H, S, and CZ. Hence, all of these automorphisms can be implemented by a Clifford circuit. Conversely, as discussed in Section 2, every Clifford operator is an automorphism of the Pauli group. Hence, Proposition 3.9 establishes that every Clifford operator admits a unique normal form. This also proves that $\mathcal{C}_n$ is generated by $-\omega$, H, S, and CZ. Moreover, since there is a bijection between Clifford operators and the normal forms, we can count the number of n-qutrit normal forms to compute the cardinality of $\mathcal{C}_n$. The proof of Corollary 3.10 can be found in Appendix B.3.

Corollary 3.10. $|\mathcal{C}_n| = 6 \cdot \prod_{k=1}^{n} 3(9^k - 1)9^k$.

4 A Complete Set of Relations for Multi-Qutrit Clifford Circuits

Here, we show how to derive a complete set of rewrite rules for multi-qutrit Clifford circuits by proving that any such circuit can be rewritten into its unique normal form. We call this the *Clifford normalization* process, and it proceeds as follows. Let $C \in \mathcal{C}_n$ be a circuit composed of gates $g_1, \ldots, g_m$. Append it on the right with the normal form of the identity circuit (N_I). Starting from the right-most gate g_m, iteratively 'push' each gate of C into the normal form, updating it at each step (e.g., from N_I to N'). This process continues until all gates have been absorbed into the updated normal form N. Below, we sketch a simple example, while Figure 45 in Appendix E provides a detailed illustration of this process.



Overview In (4) and (5), we present the normal form of the identity operator. Since $\mathcal{C}_n$ is generated by $-\omega$, H, S and CZ, it suffices to show how a normal form absorbs these gates and is updated to a new normal form. Locally, this reduces to demonstrating how to push a Clifford gate through each of the normal boxes specified in Figure 9. We call each such instance a *box relation*.

Figure 12.(a) gives an abstract illustration of *pushing through a normal box*, which results in an updated normal box M' and *residual dirty gates*, denoted as dir, appearing after M'. M' is uniquely determined by g and M. Lemmas E.2 and E.3 provide examples of finding M' and discuss this process in details. dir denotes a Clifford circuit, and it is determined by g, M, and M' collectively. Figure 12.(b) shows a concrete example of pushing an H gate through a B box.

In total, there are 380 box relations, which are summarized in Appendix C. These relations cover every possible case of pushing through a normal box, making them sufficient to normalize any Clifford

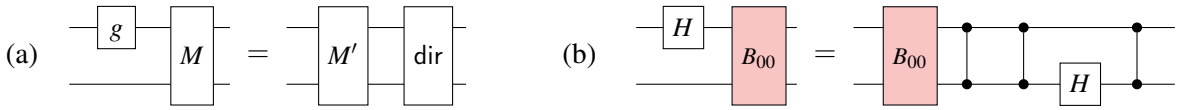


Figure 12: An abstract and concrete example of pushing a gate through a normal box M.

operator. Combined with Proposition 3.9, any two Clifford circuits that are equal as linear maps can be rewritten into the same normal form using these box relations. This shows that the box relations are *complete* for the multi-qutrit Clifford group. In [22], we further prove that these relations can be derived from the 18 gate relations listed in Figure 1. Together with the generating set $\{-\omega, H, S, \mathrm{CZ}\}$ and the derived generators in Figure 5, this provides a compact and canonical presentation of the Clifford group by generators and relations.

Technical details We need to show that the Clifford normalization terminates, and when this happens, there are no more gates left on the right-hand side of the normal form. Figure 13 provides a schematic summary of what a normal form could look like during the normalization process. It accounts for all possible cases when pushing through a normal box.

Definition 4.1. *We say that a circuit is in* clean normal form *if it is of the form in Figure 11. A circuit is in* dirty normal form *if it is of the form in Figure 13, which has the structure of a normal form, but with any number of additional Clifford gates allowed in all the designated spots, subject to the following rules.*

- *H gates can be present at any wire labelled* ①*;*
- *S gates can be present at any wire labelled* ①*,* ②*,* ③*, or* ④*;*
- *X gates can be present at any wire labelled* ②*;*
- *Z gates can be present at any wire labelled* ②*,* ③*,* ④*, or* ⑤*;*
- *CZ gates can be present at any pair of adjacent wires, provided that the top wire is labelled* ①*,* ②*, or* ③*, and the bottom wire is labelled* ①*.*

In the context of a dirty normal form, we will refer to the normal boxes as clean*, and all the other H, S, X, Z, and CZ gates as* dirty*.*

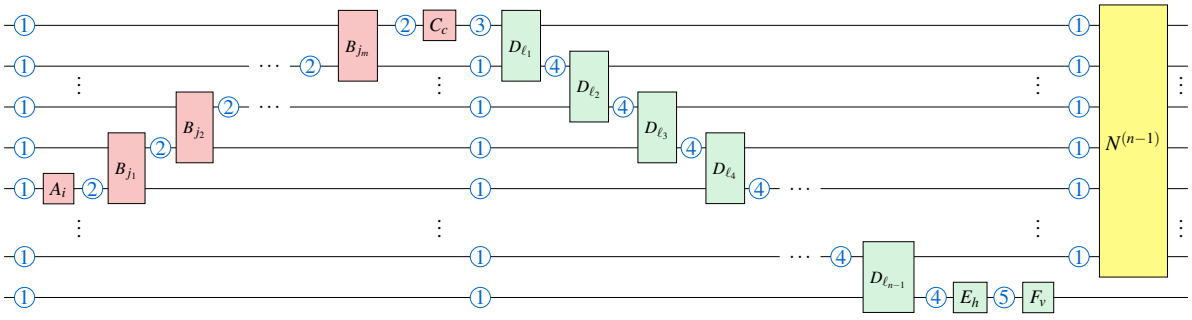


Figure 13: An example of an n-qutrit dirty normal form. Here, $N^{(n-1)}$ is in dirty normal form subject to the same constraints as $N^{(n)}$.

Note that at some locations, for instance those labelled by ②, a Pauli X or Z gate is allowed to appear in the dirty normal form. However, an H gate is not allowed at ②, even though it is used to construct Pauli gates (see Figure 5). The annotation in Figure 13 emphasizes that there can be no solitary H gates at the locations labelled by ②, ③, ④, or ⑤. It is, however, possible for an H gate to appear as 'a part of' other Clifford gates. The same convention applies to the S gate.

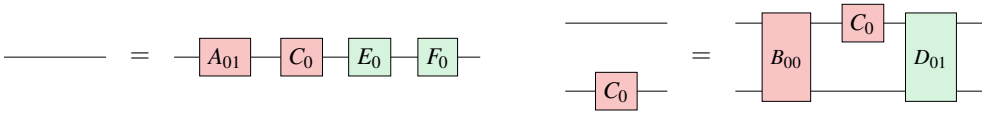
The placement of dirty gates is restricted by which gates can be properly absorbed by each normal box, as this ultimately determines whether Clifford normalization will terminate. Figures 52 to 54 summarize the allowed Clifford operators that may precede the B, C, D, E, and F boxes. Accordingly, the third column of Figure 8 identifies the additional symmetry each normal box must satisfy to ensure dirty gates always have the expected shape. By manually inspecting the box relations in Appendix C, we can confirm that all dirty gates follow the pattern described in Definition 4.1.

Lemma 4.2. *Any dirty normal form can be converted to its normal form by applying the box relations in Appendix C, in the left-to-right direction, a finite number of times.*

Proof. We adapt the proof of Lemma 5.4 in [24]. By Definition 4.1, dirty gates always appear before clean ones. Hence, if a dirty gate remains in the circuit, it must occur immediately before a clean normal box. The left-hand side of the rewrite rules in Appendix C contains all cases of a dirty gate occurring immediately before a clean normal box. Thus, so long as there are dirty gates in the circuit, one of the rules in Appendix C can be applied. Moreover, every application of a rule maps a dirty normal form to a dirty normal form (for this it is important to check that the residual dirty gates in our relations satisfy the constraints outlined in Definition 4.1).

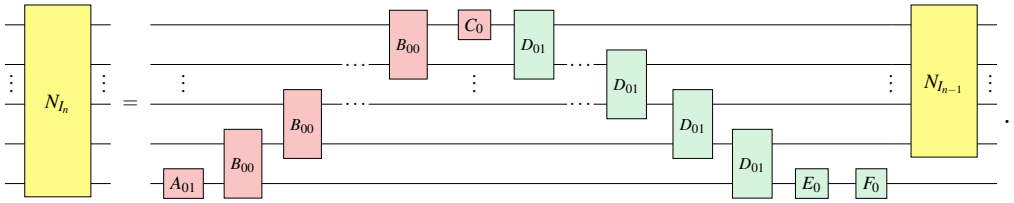
We now show that this process ends in finitely many rewrites. For this, we assign, to each dirty normal form, a sequence of nonnegative integer numbers. Assume that a dirty normal form has t clean gates (i.e. normal boxes), numbered $1,\ldots,t$ from left to right. Define $s = (s_1,\ldots,s_t)$ where s_i is the number of dirty gates that occur before the i-th clean gate. A left-to-right application of one of the box relations at the clean gate i decreases s_i and leaves $s_1,\ldots,s_{i-1}$ invariant so that s decreases in the lexicographic order. The length of the sequence of clean gates is not guaranteed to remain constant through the normalization process, but it is bounded above by the maximum possible number of clean gates. For a normal form on n qutrits, this bound is given by $\sum_{i=1}^{n} 2(i+1) = n^2+3n$. Therefore, this procedure terminates after finitely many rewrites. □

Proposition 4.3. *Consider a Clifford circuit $C \in \mathcal{C}_n$ expressed in terms of the generators $-\omega$, H, S, and CZ gates. Any such circuit can be converted to its normal form by using the box relations in Appendix C, together with the equations:*



(4)

Proof. For $1 \le k \le n$, let N_{I_k} be the normal form of the k-qutrit identity operator. First note that



(5)

Indeed, the identity circuit can be converted to this normal form by applying (4) a finite number times. Appending the given Clifford circuit C on the right with the normal form in (5), we obtain a dirty normal form, which can be converted to its normal form by Lemma 4.2. □

Together, Propositions 3.9 and 4.3 show that qutrit Clifford operators are presented by the generators $-\omega$, H, S, and CZ, the derived generators defined in Figure 5, the normal boxes given in Figure 9, as well as the 380 box relations listed in Appendix C. This presentation is highly redundant, so we provide a reduced set of relations as listed in Figure 1. This rule set remains complete for multi-qutrit Clifford circuits, but it offers a more intuitive framework for us to understand the qutrit Clifford gate interactions.

Theorem 4.4. *The* 18 *rewrite rules listed in Figure 1 are complete for qutrit Clifford circuits.*

Proof. It suffices to show that these 18 relations imply the 380 box relations. Proofs can be found in the supplement to this paper [22]. □

5 Conclusion

Generalizing from qubits to qudits, we present the first completeness result for the multi-qudit Clifford group. We consider the case when $d = 3$, and generalize the methods from [24, 31] to the qutrit setting. This is done in three steps. First, we define a normal form and show that every qutrit Clifford circuit can be uniquely written in this form. Then, we show how each Clifford circuit can be reduced to this normal form and find a set of 380 box relations that suffice for this process. Finally, we prove that these box relations can be reduced to the 18 gate relations in Figure 1. This complete set of rewrite rules offers a clear and intuitive framework for understanding the qutrit Clifford gate interactions.

Notably, our generalization of the normal form from the qubit setting is far from obvious. In [24, 31], the design space of the normal form is small enough that a working decomposition of the normal form could be found through brute force. In the qutrit case, however, the increased degrees of freedom make this approach significantly more tedious and labour-intensive. We therefore adapt the previous techniques to account for the subtleties of qutrit arithmetic and operations. This involves identifying additional symmetries that the normal form should satisfy, leading to a more canonical choice of normal form components.

Several open questions remain, the first of which concerns whether the reduced rule set presented in Figure 1 is minimal. We can show that all the single-qutrit Clifford relations except for (C5), the S-S' commutation rule, are necessary. Thus, (C1) to (C5) are nearly minimal. However, we are not aware of any arguments for the multi-qutrit Clifford relations. (C6) to (C18) appear quite natural, making it difficult to determine which, if any, could be eliminated. Alternatively, a more compact rule set might be achievable by using rotation operators as the generators for the qutrit Clifford group. Building on these considerations, we plan to generalize the framework developed in this paper to Clifford circuits in other odd prime dimensions. This will require more general techniques for constructing normal forms, supported by software tools for deriving, reducing, and verifying relations.

Acknowledgements The authors would like to thank Xiaoning Bian, Maris Ozols, and Peter Selinger for enlightening discussions, as well as the anonymous reviewers of the 22nd International Conference on Quantum Physics and Logic (QPL 2025) for their insightful comments on an earlier version of this paper. SML would like to thank Boldizsár Poór, Razin A. Shaikh, and Lia Yeh for their valuable feedback and support. The circuit diagrams in this paper and its supplement [22] were typeset using TikZiT [20].

SML and MM thank NTT Research for their financial support. This work was supported in part by Canada's NSERC. Research at IQC is supported in part by the Government of Canada through Innovation, Science and Economic Development Canada. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. YZ is supported by VILLUM FONDEN via QMATH Centre of Excellence grant number 10059 and Villum Young Investigator grant number 37532.

References

[1] Scott Aaronson & Daniel Gottesman (2004): *Improved Simulation of Stabilizer Circuits*. *Phys. Rev. A* 70, p. 052328, doi:10.1103/PhysRevA.70.052328. Available at https://link.aps.org/doi/10.1103/PhysRevA.70.052328.

[2] Paul Aigner, Maria Flors Mor-Ruiz & Wolfgang Dür (2025): *Qudit Noisy Stabilizer Formalism*. *arXiv preprint arXiv:2505.03889*. Available at https://arxiv.org/abs/2505.03889.

[3] Hussain Anwar, Earl T Campbell & Dan E Browne (2012): *Qutrit Magic State Distillation*. *New Journal of Physics* 14(6), p. 063006, doi:10.1088/1367-2630/14/6/063006. Available at https://dx.doi.org/10.1088/1367-2630/14/6/063006.

[4] Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering & Sal Wolffs (2023): *Completeness of the ZH-Calculus*. *Compositionality* 5, doi:10.32408/compositionality-5-5.

[5] M. S. Blok, V. V. Ramasesh, T. Schuster, K. O'Brien, J. M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao & I. Siddiqi (2021): *Quantum Information Scrambling on a Superconducting Qutrit Processor*. *Phys. Rev. X* 11, p. 021010, doi:10.1103/PhysRevX.11.021010.

[6] Alex Bocharov, Martin Roetteler & Krysta M. Svore (2017): *Factoring with Qutrits: Shor's Algorithm on Ternary and Metaplectic Quantum Architectures*. *Phys. Rev. A* 96, p. 012306, doi:10.1103/PhysRevA.96.012306. Available at https://link.aps.org/doi/10.1103/PhysRevA.96.012306.

[7] Nicolas Brunner, Stefano Pironio, Antonio Acin, Nicolas Gisin, André Allan Méthot & Valerio Scarani (2008): *Testing the Dimension of Hilbert Spaces*. *Phys. Rev. Lett.* 100, p. 210503, doi:10.1103/PhysRevLett.100.210503.

[8] Earl T. Campbell (2014): *Enhanced Fault-Tolerant Quantum Computing in d-Level Systems*. *Phys. Rev. Lett.* 113, p. 230501, doi:10.1103/PhysRevLett.113.230501. Available at https://link.aps.org/doi/10.1103/PhysRevLett.113.230501.

[9] Earl T. Campbell, Hussain Anwar & Dan E. Browne (2012): *Magic-State Distillation in All Prime Dimensions Using Quantum Reed-Muller Codes*. *Phys. Rev. X* 2, p. 041021, doi:10.1103/PhysRevX.2.041021. Available at https://link.aps.org/doi/10.1103/PhysRevX.2.041021.

[10] Yulin Chi, Jieshan Huang, Zhanchuan Zhang, Jun Mao, Zinan Zhou, Xiaojiong Chen, Chonghao Zhai, Jueming Bao, Tianxiang Dai, Huihong Yuan et al. (2022): *A Programmable Qudit-Based Quantum Processor*. *Nature communications* 13(1), p. 1166, doi:10.1038/s41467-022-28767-x.

[11] Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix & Benoît Valiron (2023): *A Complete Equational Theory for Quantum Circuits*. In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–13, doi:10.1109/LICS56636.2023.10175801.

[12] Hillary Dawkins & Mark Howard (2015): *Qutrit Magic State Distillation Tight in Some Directions*. *Phys. Rev. Lett.* 115, p. 030501, doi:10.1103/PhysRevLett.115.030501.

[13] Sébastien Designolle (2022): *Robust Genuine High-Dimensional Steering with Many Measurements*. *Phys. Rev. A* 105, p. 032430, doi:10.1103/PhysRevA.105.032430.

[14] Noah Goss, Alexis Morvan, Brian Marinelli, Bradley K Mitchell, Long B Nguyen, Ravi K Naik, Larry Chen, Christian Jünger, John Mark Kreikebaum, David I Santiago et al. (2022): *High-Fidelity Qutrit Entangling Gates for Superconducting Circuits*. *Nature communications* 13(1), p. 7481, doi:10.1038/s41467-022-34851-z.

[15] Daniel Gottesman (1999): *Fault-Tolerant Quantum Computation with Higher-Dimensional Systems*. In Colin P. Williams, editor: *Quantum Computing and Quantum Communications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 302–313, doi:10.1007/3-540-49208-9_27.

[16] Erik J Gustafson, Henry Lamm, Diyi Liu, Edison M Murairi & Shuchen Zhu (2025): *Synthesis of Single Qutrit Circuits from Clifford+R*. *arXiv preprint arXiv:2503.20203*. Available at https://arxiv.org/abs/2503.20203.

[17] Pavel Hrmo, Benjamin Wilhelm, Lukas Gerster, maarten van Mourik, Marcus Huber, Rainer Blatt, Philipp Schindler, Thomas Monz & Martin Ringbauer (2022): *Native Qudit Entanglement in a Trapped Ion Quantum Processor*. *Nature Communications* 14, doi:10.1038/s41467-023-37375-2.

[18] Akalank Jain & Shiroman Prakash (2020): *Qutrit and Ququint Magic States*. *Phys. Rev. A* 102, p. 042409, doi:10.1103/PhysRevA.102.042409.

[19] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, p. 569–578, doi:10.1145/3209108.3209139.

[20] Aleks Kissinger, Alexander Merry, Chris Heunen & K. Johan Paulsson: *TikZiT*. https://tikzit.github.io/index.html. Accessed: 2024-12-08.

[21] Aleks Kissinger & John van de Wetering (2024): *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint. Available at https://github.com/zxcalc/book?tab=readme-ov-file.

[22] Sarah Meng Li, Michele Mosca, Neil J. Ross, John van de Wetering & Yuming Zhao (2025): *A Complete and Natural Rule Set for Multi-Qutrit Clifford Circuits: Derived Relations and Supplemental Proofs*. *Available as an ancillary file from this paper's arXiv page.*

[23] Pei Jiang Low, Brendan M. White, Andrew A. Cox, Matthew L. Day & Crystal Senko (2020): *Practical Trapped-Ion Protocols for Universal Qudit-Based Quantum Computing*. *Phys. Rev. Res.* 2, p. 033128, doi:10.1103/PhysRevResearch.2.033128.

[24] Justin Makary, Neil J Ross & Peter Selinger (2021): *Generators and Relations for Real Stabilizer Operators*. *2021, Electronic Proceedings in Theoretical Computer Science, p. 14-36*, doi:10.4204/eptcs.343.2.

[25] Kang Feng Ng & Quanlong Wang (2017): *A Universal Completion of the ZX-Calculus*. *arXiv preprint arXiv:1706.09877*. Available at https://arxiv.org/abs/1706.09877.

[26] Anastasiia S Nikolaeva, Ilia V Zalivako, Alexander S Borisenko, Nikita V Semenin, Kristina P Galstyan, Andrey E Korolkov, Evgeniy O Kiktenko, Ksenia Yu Khabarova, Ilya A Semerikov, Aleksey K Fedorov et al. (2024): *Scalable Improvement of the Generalized Toffoli Gate Realization Using Trapped-Ion-Based Qutrits*. *arXiv preprint arXiv:2407.07758*. Available at https://arxiv.org/abs/2407.07758.

[27] Archimedes Pavlidis & Emmanuel Floratos (2021): *Quantum-Fourier-Transform-Based Quantum Arithmetic with Qudits*. *Phys. Rev. A* 103, p. 032417, doi:10.1103/PhysRevA.103.032417. Available at https://link.aps.org/doi/10.1103/PhysRevA.103.032417.

[28] Boldizsár Poór, Robert I Booth, Titouan Carette, John Van De Wetering & Lia Yeh (2023): *The Qupit Stabiliser ZX-Travaganza: Simplified Axioms, Normal Forms and Graph-Theoretic Simplification*. *2023, Electronic Proceedings in Theoretical Computer Science, p. 220-264*, doi:10.4204/EPTCS.384.13.

[29] Shiroman Prakash & Tanay Saha (2025): *Low Overhead Qutrit Magic State Distillation*. *Quantum* 9, p. 1768, doi:10.22331/q-2025-06-12-1768.

[30] Martin Ringbauer, Michael Meth, Lukas Postler, Roman Stricker, Rainer Blatt, Philipp Schindler & Thomas Monz (2021): *A Universal Qudit Quantum Processor with Trapped Ions*. Nature Physics 18, pp. 1053 – 1057, doi:10.1038/s41567-022-01658-0.

[31] Peter Selinger (2015): *Generators and Relations for N-Qubit Clifford Operators*. Logical Methods in Computer Science Volume 11, Issue 2:10, doi:10.2168/LMCS-11(2:10)2015. Available at https://lmcs.episciences.org/1570.

[32] Vinay Tripathi, Noah Goss, Arian Vezvaee, Long B. Nguyen, Irfan Siddiqi & Daniel A. Lidar (2025): *Qudit Dynamical Decoupling on a Superconducting Quantum Processor*. Phys. Rev. Lett. 134, p. 050601, doi:10.1103/PhysRevLett.134.050601. Available at https://link.aps.org/doi/10.1103/PhysRevLett.134.050601.

[33] Robert Frederik Uy & Dorian A. Gangloff (2024): *Qudit-Based Quantum Error-Correcting Codes from Irreducible Representations of $SU(d)$*. arXiv:2410.02407. Available at https://arxiv.org/abs/2410.02407.

[34] Quanlong Wang (2018): *Qutrit ZX-Zalculus is Complete for Stabilizer Quantum Mechanics*. 2018, Electronic Proceedings in Theoretical Computer Science, p. 58-70, doi:10.4204/eptcs.266.3.

[35] Yuchen Wang, Zixuan Hu, Barry C Sanders & Sabre Kais (2020): *Qudits and High-Dimensional Quantum Computing*. Frontiers in Physics Volume 8 - 2020, doi:10.3389/fphy.2020.589504.

[36] Karen Wintersperger, Florian Dommert, Thomas Ehmer, Andrey Hoursanov, Johannes Klepsch, Wolfgang Mauerer, Georg S. Reuber, Thomas Strohm, Ming-Yu Yin & Sebastian Luber (2023): *Neutral Atom Quantum Computing Hardware: Performance and End-User Perspective*. EPJ Quantum Technology 10, pp. 1–26, doi:10.1140/epjqt/s40507-023-00190-1.

[37] Biaoliang Ye, Zhen-Fei Zheng, Yu Zhang & Chui-Ping Yang (2018): *Circuit QED: Single-Step Realization of a Multiqubit Controlled Phase Gate with One Microwave Photonic Qubit Simultaneously Controlling n-1 Microwave Photonic Qubits*. Opt. Express 26(23), pp. 30689–30702, doi:10.1364/OE.26.030689.

[38] M. A. Yurtalan, J. Shi, M. Kononenko, A. Lupascu & S. Ashhab (2020): *Implementation of a Walsh-Hadamard Gate in a Superconducting Qutrit*. Phys. Rev. Lett. 125, p. 180504, doi:10.1103/PhysRevLett.125.180504.

A Complementary Definitions

Here we provide complementary definitions that are helpful for discussing our results.

By putting Definitions 3.2 and 3.3 together, Figure 14 shows the normal form of an n-qutrit Clifford circuit when expanding the Z- and X-normal circuits in Figure 10. Figures 15 and 16 describe the individual construction of each normal box in terms of H, S, CZ, and XC gates.

According to Section 2.2, we can describe the automorphism of each normal box by its actions on the Pauli generators. That is, consider a normal box M and a Pauli generator P, $MPM^\dagger = Q$, for some Pauli Q. For our purposes, it is also convenient to describe M's *inverse conjugation* on Pauli generators: $M^\dagger PM = Q'$, for some Pauli Q'.

For all normal boxes, Figure 9 provides a high-level overview of their constructions using parameterized definitions. Figures 15 and 16 provide a microscopic picture of their constructions. Based on these two perspectives, Figure 17 summarizes important behaviours of normal boxes. The action of A, C, E, and F boxes on $\mathcal{P}_1$ are described by their actions on Paulis X and Z, where we write the (pre)image of X above the qutrit wire, and that of Z below it. Based on the generic parameterized constructions of B and D boxes, it suffices to specify the automorphism of B_{00}, B_{01}, D_{00}, D_{01}, and D_{02}. Also note that $D_{02} = B_{00}$ and $D_{01} = B_{00}^{-1}$.

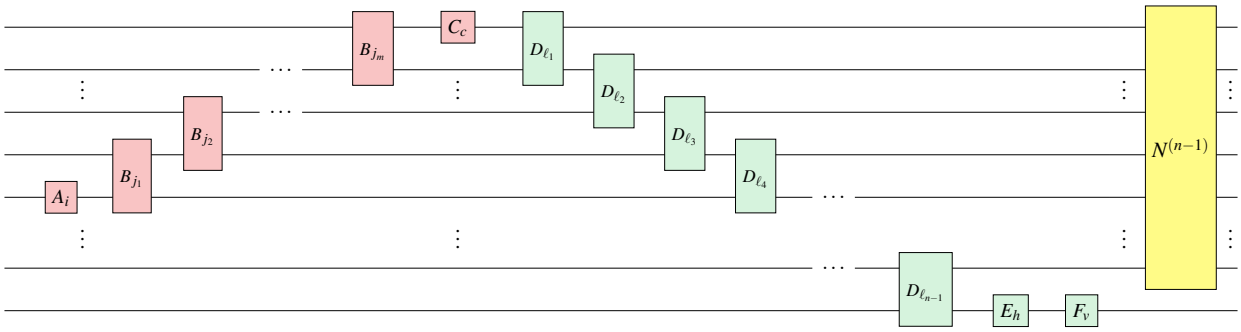


Figure 14: A close-up view of the normal form of an n-qutrit Clifford circuit. $N^{(n-1)}$ has the same structure as the circuit before it, but on one fewer qutrit.

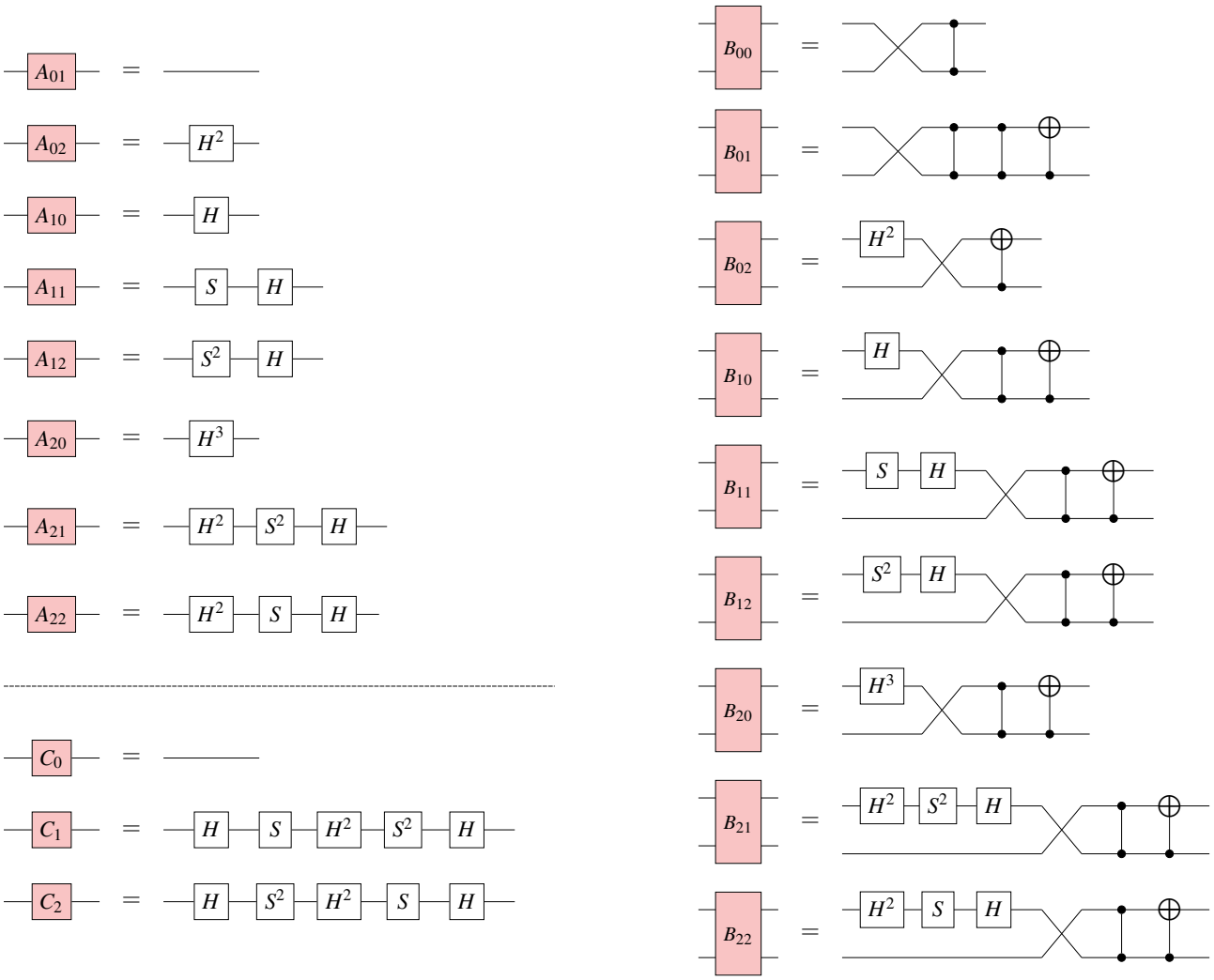


Figure 15: A microscopic picture of the Z normal boxes.

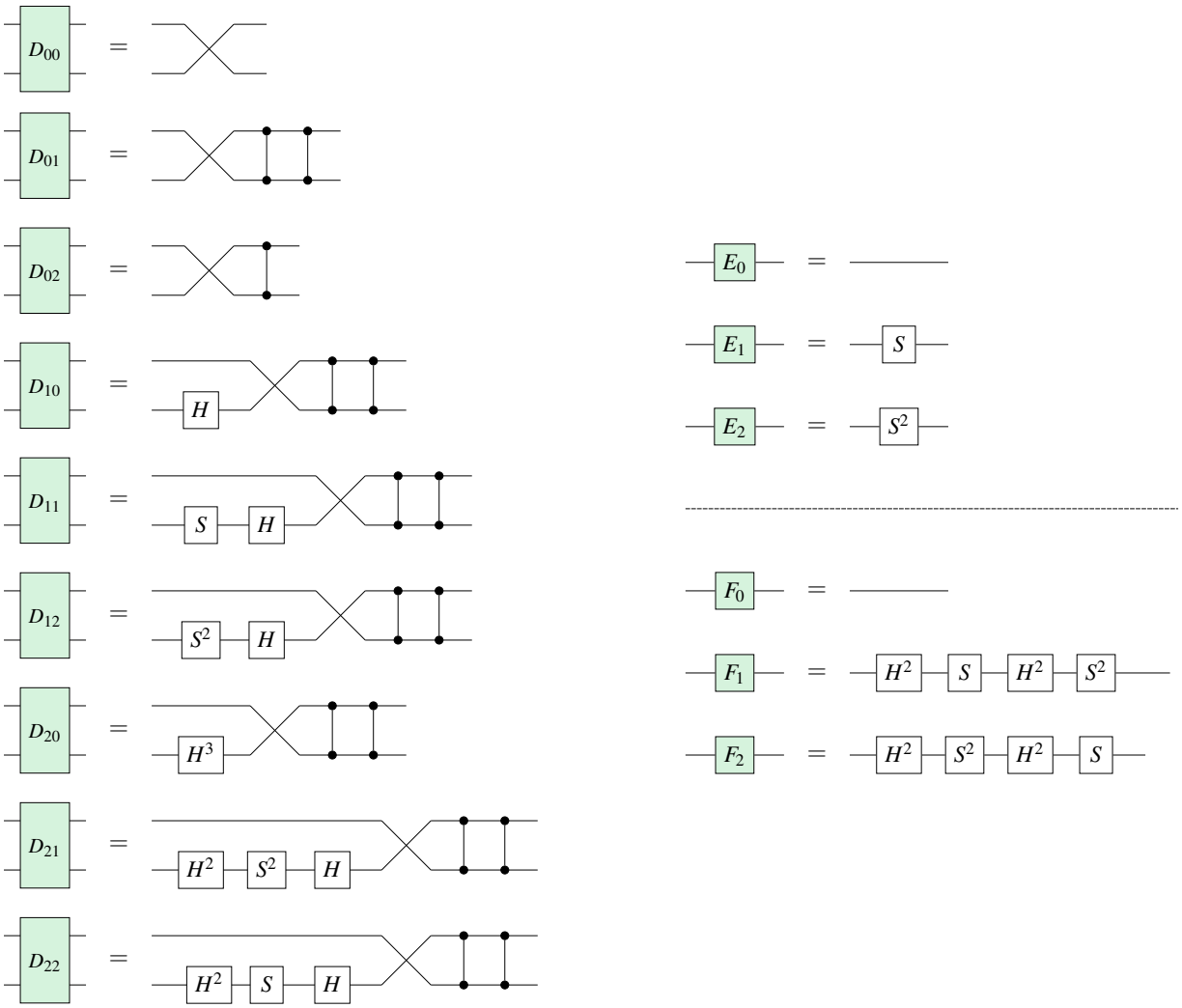


Figure 16: A microscopic picture of the X normal boxes.

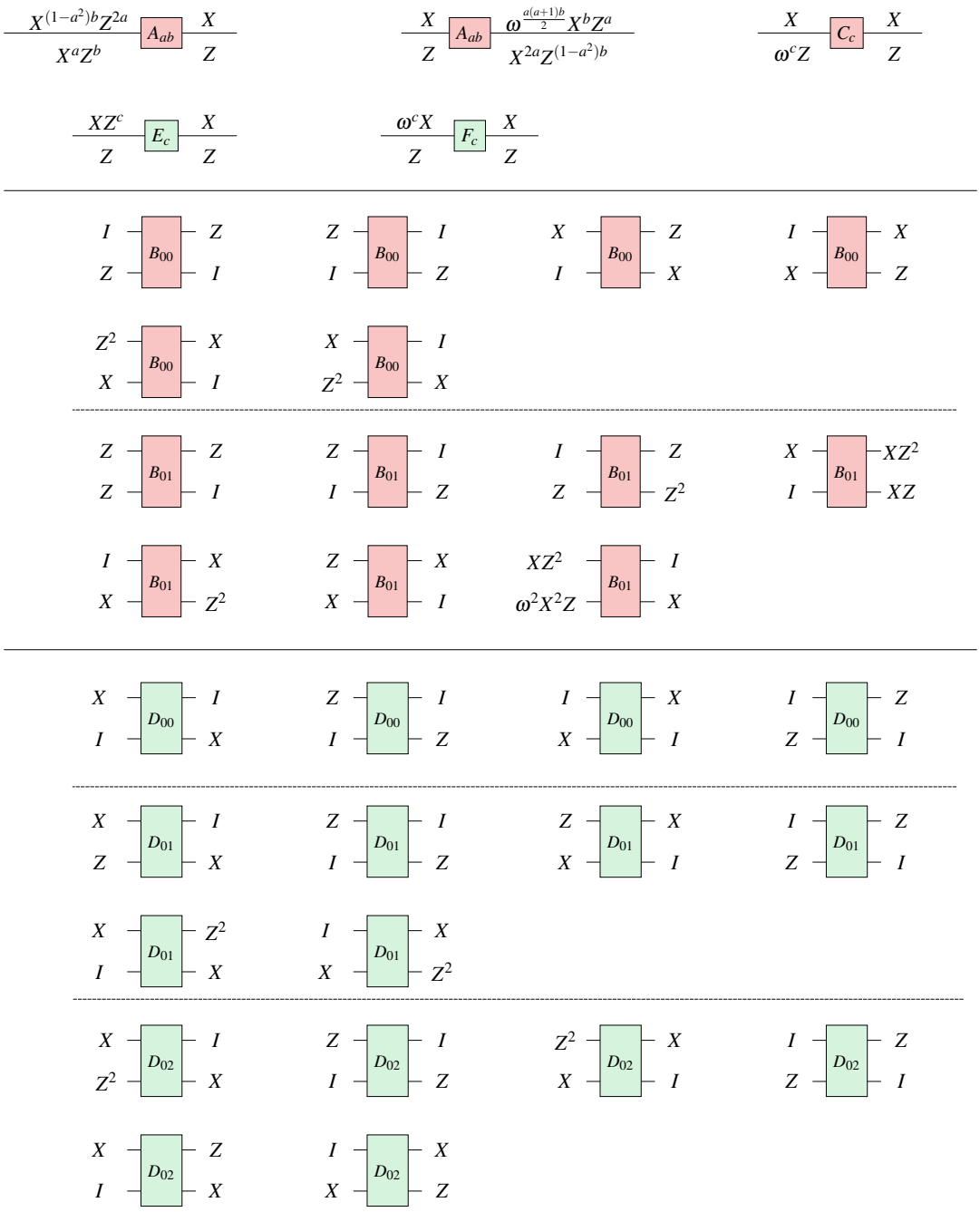


Figure 17: The actions of A, C, E, and F boxes on $\mathcal{P}_1$ are described by their actions on the single-qutrit Pauli generators X and Z. $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$ and $c \in \mathbb{Z}_3$. The actions of B_{00}, B_{01}, D_{00}, D_{01}, and D_{02} on $\mathcal{P}_2$ are described by their actions on the two-qutrit Pauli generators $X \otimes I$, $I \otimes X$, $Z \otimes I$, and $I \otimes Z$. For each Pauli generator, we display both their images and preimages within a normal box. These will be useful for finding all box relations.

B Complementary Proofs

Here, we provide detailed proofs of the results presented in Section 3. Let $\lambda \in \mathbb{C}$. $\lambda^\dagger$ denotes the complex conjugate of λ. When u, τ, and ν are elements of a ring R, we write $u \equiv_\nu \tau$ if u is congruent to τ modulo ν. That is, $u - \tau = r\nu$, for some $r \in R$. We write $u \not\equiv_\nu \tau$ otherwise.

B.1 Scalars of $\mathcal{C}_n$

We start by showing that all Clifford scalars are of the form $(-\omega)^t$, for $t \in \mathbb{Z}_6$. This becomes useful for finding the cardinality of $\mathcal{C}_n$, as shown in Corollary 3.10. To this end, we first show that the phases (complex numbers of unit norm) that belong to the ring $\mathbb{Z}[1/3, \omega]$ are exactly the complex numbers of the form $(-\omega)^t$, for $t \in \mathbb{Z}_6$. Since the complex numbers of this form all belong to $\mathbb{Z}[1/3, \omega]$, we only need to show that if an element of $\mathbb{Z}[1/3, \omega]$ has norm 1, then it is of the required form.

Lemma B.1. *Let $a, b \in \mathbb{Z}$. If $a \not\equiv_3 0$, $b \not\equiv_3 0$, and $a \not\equiv_3 b$, then $a^2 - ab + b^2 \equiv_9 3$.*

Proof. Since $a \not\equiv_3 0$, $b \not\equiv_3 0$, and $a \not\equiv_3 b$, we have either $a \equiv_3 1$ and $b \equiv_3 2$, or $a \equiv_3 2$ and $b \equiv_3 1$. Regardless, we then have

$$a + b \equiv_3 0 \quad \text{and} \quad ab \equiv_3 2.$$

The above implies that

$$(a+b)^2 \equiv_9 0 \quad \text{and} \quad 3ab \equiv_9 6.$$

Indeed, if $a + b = 3\ell$ for some integer ℓ, then $(a+b)^2 = 9\ell^2$; and similarly, if $ab = 2 + 3\ell$, then $3ab = 6 + 9\ell$. Hence, we get

$$a^2 - ab + b^2 = a^2 + 2ab + b^2 - 3ab = (a+b)^2 - 3ab \equiv_9 0 - 6 \equiv_9 3,$$

as desired. □

Proposition B.2. *Let $x \in \mathbb{Z}[1/3, \omega]$. If $||x|| = 1$, then $x = (-\omega)^t$, for $t \in \mathbb{Z}_6$.*

Proof. Let $x \in \mathbb{Z}[1/3, \omega]$. Then x can be written as $x = (a + \omega b)/3^k$, for some integers a, b, and k, with $k \geq 0$ and k minimal. Since $||x|| = 1$, we then have $x^\dagger x = 1$, and thus,

$$(a + \omega b)^\dagger (a + \omega b) = 3^{2k}. \tag{6}$$

Since $\omega^\dagger = \omega^2 = -1 - \omega$, we can rewrite (6) as

$$3^{2k} = (a + \omega b)^\dagger (a + \omega b) = a^2 + \omega ab + \omega^\dagger ab + b^2 = a^2 - ab + b^2. \tag{7}$$

If $k = 0$, then (7) becomes $a^2 - ab + b^2 = 1$, whose only solutions are $a = \pm 1$ and $b = 0$, $a = 0$ and $b = \pm 1$, and $a = \pm 1$ and $b = \mp 1$. These six solutions correspond exactly to $x = (-\omega)^t$, for $t \in \mathbb{Z}_6$.

To complete the proof, we now show that there are no solutions to (7) with $k \geq 1$. It suffices to show that $k \geq 1$ implies $a \not\equiv_3 0$, $b \not\equiv_3 0$, and $a \not\equiv_3 b$. Indeed, if this is the case, then Lemma B.1 shows that $a^2 - ab + b^2$ is not congruent to 0 modulo 9, which contradicts (7), since $3^{2k} \equiv_9 0$ when $k \geq 1$. If $k \geq 1$, then $a \not\equiv_3 0$ or $b \not\equiv_3 0$, since both a and b being divisible by 3 would contradict the minimality of k. But if $a \equiv_3 0$, then (7) implies that $b^2 \equiv_3 0$, so that $b \equiv_3 0$. Similarly, if $b \equiv_3 0$, then $a \equiv_3 0$. Therefore, it must be the case that both $a \not\equiv_3 0$ and $b \not\equiv_3 0$. Finally, since neither a nor b is congruent to 0 modulo 3, $a \equiv_3 b$ would imply

$$a^2 - ab + b^2 \equiv_3 1 - 1 + 1 \equiv_3 1,$$

which contradicts (7). Thus we must also have $a \not\equiv_3 b$. □

Corollary B.3. *Let $U,V \in \mathcal{C}_n$ and U,V act identically on $\mathcal{P}_n$. Then $V = (-\omega)^t U$, $t \in \mathbb{Z}_6$.*

Proof. For all $P \in \mathcal{P}_n$, since $UPU^\dagger = VPV^\dagger$, $(V^\dagger U)P(U^\dagger V) = (V^\dagger V)P(V^\dagger V) = P$. By Proposition 2.8, $V^\dagger U = \lambda$, for some $\lambda \in \mathbb{C}$. Recall that $\mathcal{C}_n = \langle H, S, \mathrm{CZ}\rangle$. By direct computation, H, S, and CZ are unitaries over $\mathbb{Z}[1/3,\omega]$. This implies that $U,V \in \mathcal{C}_n$ are unitaries over $\mathbb{Z}[1/3,\omega]$. Hence $\lambda \in \mathbb{Z}[1/3,\omega]$ and $\lambda\lambda^\dagger = 1$. By Proposition B.2, $\lambda \in \{(-\omega)^t;\ t \in \mathbb{Z}_6\}$. Based on the relation (C4) in Figure 1, $(HS^2)^3 = -\omega$. Hence, $\mathcal{C}_n$ must contain all powers of $-\omega$. This completes the proof. □

B.2 Proofs of Lemmas 3.5, 3.6, 3.7, 3.8, and Proposition 3.9

Following the conventions in Section 2.2, we use $C \bullet P$ to denote the conjugation of a Clifford $C \in \mathcal{C}_n$ on a Pauli $P \in \mathcal{P}_n$. Let $Q = C \bullet P$. We say that in C, Q is the *image* of P and P is the *preimage* of Q. Since C is unitary, we can also take the inverse of the circuit to get $C^\dagger \bullet Q = C^\dagger QC = P$.

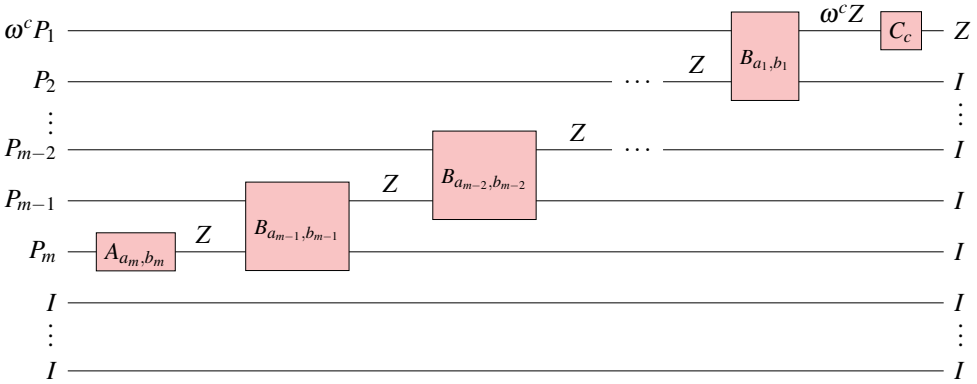
Definition B.4. *Let $P,Q \in \mathcal{P}_n$ and $PQ = \omega QP$. Then we say P ω-anticommutes with Q.*

Lemma 3.5. *For $n \geq 1$, let P be an n-qutrit Pauli operator and $P \notin \{I, \omega I, \omega^2 I\}$. Then there exists a unique Z-normal circuit L such that $L \bullet P = Z \otimes I \otimes \cdots \otimes I$.*

Proof. By Definition 2.2, an n-qutrit Pauli operator P has the form $P = \omega^c \bigotimes_{j=1}^n P_j$, $P_j = X^{a_j} Z^{b_j}$, for $a_j, b_j, c \in \mathbb{Z}_3$. Since $P \notin \{I, \omega I, \omega^2 I\}$, there exists $m \in \mathbb{N}$ and $1 \leq m \leq n$ such that P_m is not an identity. Let m be the largest such index, then

$$P = (\omega^c P_1) \otimes \cdots \otimes P_m \otimes I \otimes \cdots \otimes I = (\omega^c X^{a_1} Z^{b_1}) \otimes \cdots \otimes (X^{a_m} Z^{b_m}) \otimes I \otimes \cdots \otimes I. \tag{8}$$

Next, we show how to construct a unique Z-normal circuit L based on (8) and the unique normal box actions specified in the second column of Figure 8. Starting from qutrit wire m, A_{a_m,b_m} is uniquely determined by $P_m = X^{a_m} Z^{b_m}$. As a result, this A box maps $X^{a_m} Z^{b_m}$ to Z. Consequently, $B_{a_{m-1},b_{m-1}}$ is uniquely determined by $P_{m-1} \otimes Z = X^{a_{m-1}} Z^{b_{m-1}} \otimes Z$. Then, this B box maps $P_{m-1} \otimes Z$ to $Z \otimes I$. Continue this process by concatenating B normal boxes upward, until reaching the top two qutrit wires, where B_{a_1,b_1} is uniquely determined by $\omega^c P_1 \otimes Z = \omega^c X^{a_1} Z^{b_1} \otimes Z$. As a result, $\omega^c P_1 \otimes Z$ is mapped to $\omega^c Z \otimes I$. Finally, C_c is uniquely determined by $\omega^c Z$, and it maps $\omega^c Z$ to Z. This gives us a Z-normal circuit L that is displayed below. It is uniquely determined by P and $L \bullet P = Z \otimes I \otimes \cdots \otimes I$.



□

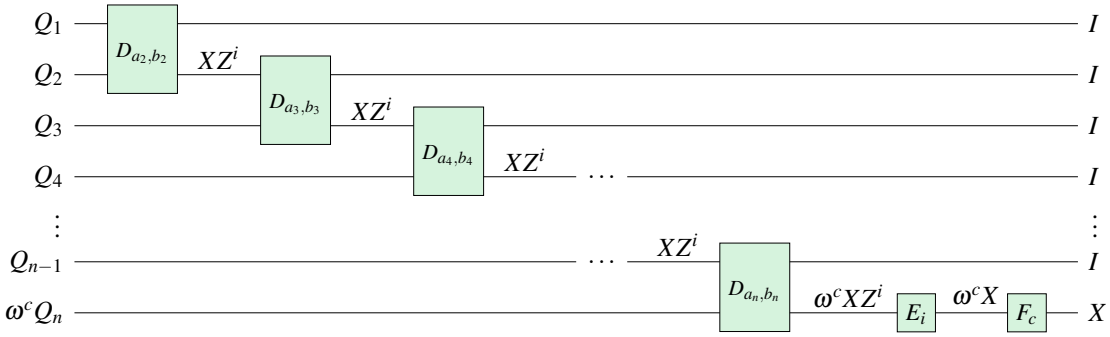
Given a Z-normal circuit L, we can read off its preimage of $Z \otimes I \otimes \cdots \otimes I$ by reading the indices of each normal box in L from right to left.

Lemma 3.6. *For $n \geq 1$, let Q be an n-qutrit Pauli operator and $(Z \otimes I \otimes \cdots \otimes I)Q = \omega Q(Z \otimes I \otimes \cdots \otimes I)$. Then there exists a unique X-normal circuit M such that $M \bullet Q = I \otimes \cdots \otimes I \otimes X$.*

Proof. By Definition 2.2, an n-qutrit Pauli operator Q has the form $Q = \omega^c \bigotimes_{j=1}^{n} Q_j$, $Q_j = X^{a_j} Z^{b_j}$, for $a_j, b_j, c \in \mathbb{Z}_3$. Since $(Z \otimes I \otimes \cdots \otimes I)Q = \omega Q(Z \otimes I \otimes \cdots \otimes I)$, $Q_1 = XZ^i$, for some $i \in \mathbb{Z}_3$. Then

$$Q = Q_1 \otimes Q_2 \otimes \cdots \otimes (\omega^c Q_n) = (XZ^i) \otimes (X^{a_2} Z^{b_2}) \otimes \cdots \otimes (\omega^c X^{a_n} Z^{b_n}). \tag{9}$$

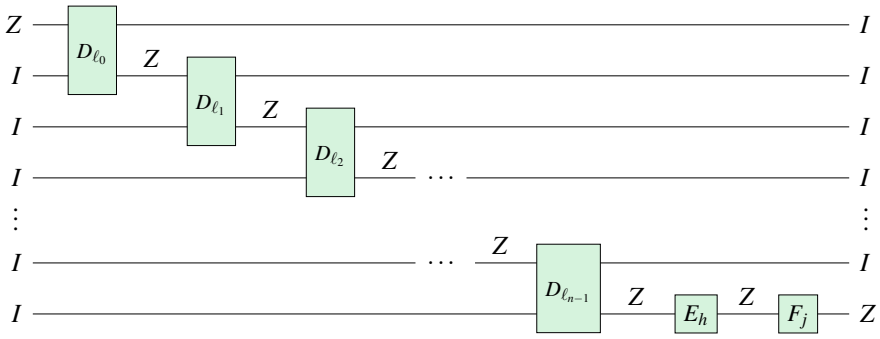
Next, we show how to construct a unique X-normal circuit M based on (9) and the unique normal box actions specified in the second column of Figure 8. Starting from qutrit wires 1 and 2, D_{a_2,b_2} is uniquely determined by $Q_1 \otimes Q_2 = (XZ^i) \otimes (X^{a_2} Z^{b_2})$. As a result, this D box maps $(XZ^i) \otimes Q_2$ to $I \otimes (XZ^i)$. Consequently, D_{a_3,b_3} is uniquely determined by $(XZ^i) \otimes Q_3 = (XZ^i) \otimes (X^{a_3} Z^{b_3})$. Again, this D box maps $(XZ^i) \otimes Q_3$ to $I \otimes (XZ^i)$. Continue this process by concatenating D normal boxes downward, until reaching the bottom two qutrit wires, where D_{a_n,b_n} is uniquely determined by $(XZ^i) \otimes \omega^c Q_n = (XZ^i) \otimes (\omega^c X^{a_n} Z^{b_n})$. Thus, $(XZ^i) \otimes \omega^c Q_n$ is mapped to $I \otimes (\omega^c XZ^i)$ by this last D box. Then, E_i is uniquely determined by XZ^i, and it maps $\omega^c XZ^i$ to $\omega^c X$. Finally, F_c is uniquely determined by $\omega^c X$, and it maps $\omega^c X$ to X. This gives us an X-normal circuit M that is displayed below. It is uniquely determined by Q and $M \bullet Q = I \otimes \cdots \otimes I \otimes X$.



Given an X-normal circuit M, we can read off its preimage of $I \otimes \cdots \otimes I \otimes X$ by reading the indices of each normal box in M from right to left.

Lemma 3.7. *Every X-normal circuit M satisfies $M \bullet (Z \otimes I \otimes \cdots \otimes I \otimes I) = I \otimes I \otimes \cdots \otimes I \otimes Z$.*

Proof. This follows from the additional actions of D, E, and F boxes on certain Pauli operators (see the third column of Figure 8). In particular, every D box maps $Z \otimes I$ to $I \otimes Z$, every E and F box maps Z to Z. It is visualized below.



Intuitively, an X-normal circuit M propagates a Pauli Z from the top wire to the bottom wire.

Lemma 3.8. *For $n \geq 1$, let P, Q be n-qutrit Pauli operators such that $PQ = \omega QP$. Then there exist unique circuits M and L, where M is X-normal and L is Z-normal such that $(ML) \bullet P = I \otimes \cdots \otimes I \otimes Z$ and $(ML) \bullet Q = I \otimes \cdots \otimes I \otimes X$.*

Proof. Since $PQ = \omega QP$, $P \notin \{I, \omega I, \omega^2 I\}$. By Lemma 3.5, there exists a unique Z-normal circuit L such that $L \bullet P = Z \otimes I \otimes \cdots \otimes I$. Moreover, $(L \bullet P)(L \bullet Q) = L \bullet (PQ) = L \bullet (\omega QP) = \omega (L \bullet Q)(L \bullet P)$. Hence $(Z \otimes I \otimes \cdots \otimes I)(L \bullet Q) = \omega (L \bullet Q)(Z \otimes I \otimes \cdots \otimes I)$. By Lemma 3.6, there exists a unique X-normal circuit M such that $M \bullet (L \bullet Q) = I \otimes \cdots \otimes I \otimes X$. By Lemma 3.7, $M \bullet (L \bullet P) = I \otimes \cdots \otimes I \otimes Z$. It follows that

$$ML \bullet P = I \otimes \cdots \otimes I \otimes Z, \qquad ML \bullet Q = I \otimes \cdots \otimes I \otimes X. \tag{10}$$

This proves the existence of the desired Z-normal circuit L and X-normal circuit M. Now suppose towards contradiction that there exist another Z-normal circuit L' and X-normal circuit M' satisfying (10). Since $M' \bullet (L' \bullet P) = I \otimes \cdots \otimes I \otimes Z$, Lemma 3.7 implies that $L' \bullet P = Z \otimes I \cdots \otimes I$. By the uniqueness of the Z-normal circuit in Lemma 3.5, $L = L'$. Then $M' \bullet (L' \bullet Q) = M' \bullet (L \bullet Q) = I \otimes \cdots \otimes I \otimes X$. By the uniqueness of the X-normal circuit in Lemma 3.6, $M' = M$. □

Proposition 3.9. *Let $\phi : \mathcal{P}_n \to \mathcal{P}_n$ be an automorphism of the Pauli group such that ϕ fixes scalars. Then there exists a Clifford circuit C in normal form such that for all $P \in \mathcal{P}_n$, $C \bullet P = \phi(P)$. Moreover, the normal form C is unique up to a scalar $(-\omega)^t$, $t \in \mathbb{Z}_6$.*

Proof. We proceed by induction on n. When $n = 0$, the Pauli operators are scalars of the form ω^t, $t \in \mathbb{Z}_3$. In this case, ϕ is the identity. Setting $C = 1$, uniqueness up to scalar follows from the fact that when $n = 0$, the Clifford operators are scalars of the form $(-\omega)^t, t \in \mathbb{Z}_6$.

Now suppose that our claim is true for $n-1$ and consider the case of n. First we prove existence. Since ϕ is bijective, there exist $P, Q \in \mathcal{P}_n$ such that $\phi(P) = I \otimes \cdots \otimes I \otimes Z$ and $\phi(Q) = I \otimes \cdots \otimes I \otimes X$. That is, $P = \phi^{-1}(I \otimes \cdots \otimes I \otimes Z)$ and $Q = \phi^{-1}(I \otimes \cdots \otimes I \otimes X)$. Then $PQ = \phi^{-1}(I \otimes \ldots \otimes I \otimes ZX)$. Since $I \otimes \ldots \otimes I \otimes Z$ ω-anticommutes with $I \otimes \ldots \otimes I \otimes X$, $PQ = \omega QP$. By Lemma 3.8, there exist a unique X-circuit M and a unique Z-circuit L such that $ML \bullet P = I \otimes \ldots \otimes I \otimes Z = \phi(P)$ and $ML \bullet Q = I \otimes \ldots \otimes I \otimes X = \phi(Q)$. Let $\phi' : \mathcal{P}_n \to \mathcal{P}_n$ be a new automorphism and ϕ' fixes scalars.

$$\phi'(U) = \phi\left((ML)^{-1} \bullet U\right), \tag{11}$$

Then $I \otimes \cdots \otimes I \otimes Z$ and $I \otimes \cdots \otimes I \otimes X$ are fixed points of ϕ', since for $T \in \{X, Z\}$,

$$\begin{aligned}\phi'(I \otimes \cdots \otimes I \otimes T) &= \phi\left((ML)^{-1} \bullet (I \otimes \cdots \otimes I \otimes T)\right) \\ &= \phi\left((ML)^{-1} \bullet (ML) \bullet T\right) = \phi(T) = I \otimes \cdots \otimes I \otimes T.\end{aligned}$$

For any $R \in \mathcal{P}_{n-1}$, since $R \otimes I$ commutes with $I \otimes \cdots \otimes I \otimes Z$ and $I \otimes \cdots \otimes I \otimes X$, $\phi'(R \otimes I)$ commutes with $\phi'(I \otimes \cdots \otimes I \otimes Z) = I \otimes \cdots \otimes I \otimes Z$ and $\phi'(I \otimes \cdots \otimes I \otimes X) = I \otimes \cdots \otimes I \otimes X$. This implies that $\phi'(R \otimes I) = S \otimes I$ for some $S \in \mathcal{P}_{n-1}$. Then there exists an automorphism $\phi'' : \mathcal{P}_{n-1} \to \mathcal{P}_{n-1}$ such that $\phi''(R) = S$. Since ϕ' fixes $I \otimes \cdots \otimes I \otimes Z$ and $I \otimes \cdots \otimes I \otimes X$,

$$\phi' = \phi'' \otimes I. \tag{12}$$

By the induction hypothesis, for all $R \in \mathcal{P}_{n-1}$, there exists $C' \in \mathcal{C}_{n-1}$ in normal form such that

$$C' \bullet R = \phi''(R), \tag{13}$$

Then $C = (C' \otimes I)ML$ is in normal form. Next, we show that $C \bullet U = \phi(U)$ for all $U \in \mathcal{P}_n$. By (11),

$$(ML)^{-1} \bullet U = \phi^{-1}\phi'(U). \tag{14}$$

By taking the inverse of both sides of (14), we have

$$ML = (\phi^{-1}\phi')^{-1} = \phi'^{-1}\phi. \tag{15}$$

It follows that

$$C \bullet U = ((C' \otimes I)ML) \bullet U \overset{(15)}{=\!=} ((C' \otimes I) \bullet (\phi'^{-1}(\phi(U))) \overset{(12)}{=\!=} (C' \otimes I) \bullet (\phi''^{-1} \otimes I)(\phi(U)) \overset{(13)}{=\!=} \phi(U).$$

To prove uniqueness, suppose towards contradiction that $D \in \mathcal{C}_n$ is another Clifford circuit in normal form such that $D \bullet U = \phi(U)$ for all $U \in \mathcal{P}_n$. By induction, $D = (D' \otimes I)M'L'$, where M' is an X-normal circuit, L' is a Z-normal circuit, and D' is a normal Clifford circuit on $n-1$ qutrits. Since $D \bullet P = \phi(P) = I \otimes \cdots \otimes I \otimes Z$, $((D' \otimes I)(M'L')) \bullet P = I \otimes \cdots \otimes I \otimes Z$. Then

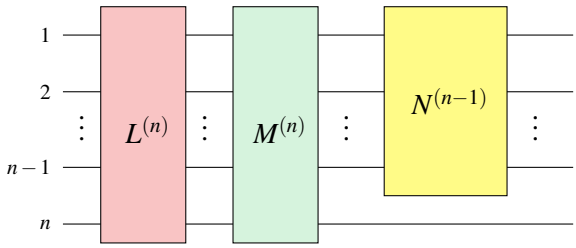
$$(M'L') \bullet P = (D' \otimes I)^{-1} \bullet (I \otimes \cdots \otimes I \otimes Z) = (D'^{-1} \otimes I) \bullet (I \otimes \cdots \otimes I \otimes Z) = I \otimes \cdots \otimes I \otimes Z.$$

Similarly, $D \bullet Q = I \otimes \cdots \otimes I \otimes X$ implies that $(M'L') \bullet Q = I \otimes \cdots \otimes I \otimes X$. By the uniqueness of the X- and Z-normal circuits in Lemma 3.8, $M' = M$ and $L' = L$. By induction hypothesis, $C' = D'$, up to $(-\omega)^t, t \in \mathbb{Z}_6$. Thus, the same is true for C and D. This completes the proof. □

B.3 Proof of Corollary 3.10

Corollary 3.10. $|\mathcal{C}_n| = 6 \cdot \prod_{k=1}^{n} 3(9^k - 1)9^k$.

Proof. By the uniqueness of the normal form and the correspondence established in Proposition 3.9, it is sufficient to count the total number of distinct normal forms. Up to scalar, consider the normal form of an arbitrary n-qutrit Clifford operator:

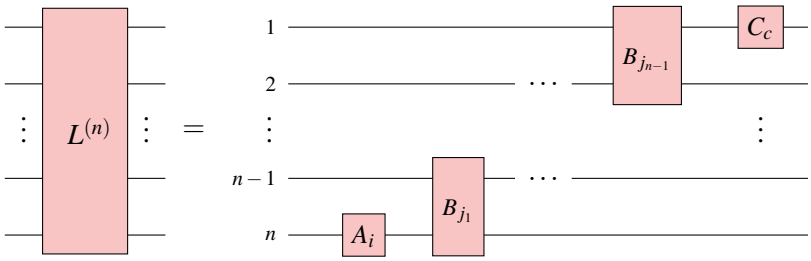


By induction, our problem is reduced to counting different ways of constructing $L^{(n)}$ and $M^{(n)}$.

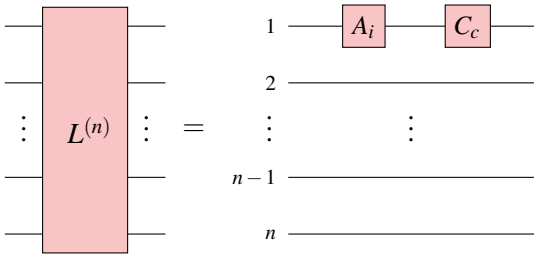
- For $L^{(n)}$, according to Figure 9, there are 8 choices for an A box and 3 choices for a C box. For B boxes, we can start concatenating them upward from any qutrit wire. According to Figure 10a, we proceed by cases.

  In one extreme case: The ladder of B boxes starts from the bottom qutrit wire, as shown below. There are $n-1$ B boxes in $L^{(n)}$. According to Figure 9, there are 9 choices for a B box. Since each choice of a normal box is independent of each other, there are $8 \cdot 3 \cdot 9^{(n-1)} = 24 \cdot 9^{(n-1)}$

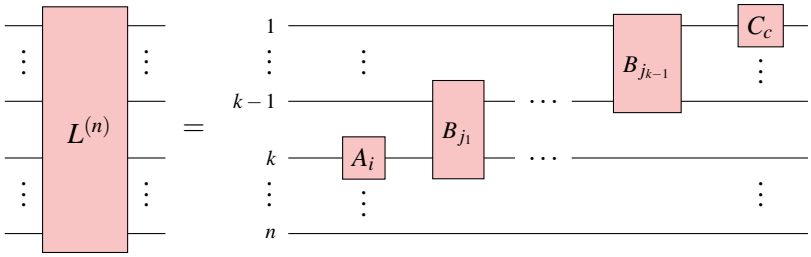
distinct $L^{(n)}$ when it expands over all n qutrits.



In the other extreme case: The ladder of B boxes starts from the top qutrit wire, as shown below. There is no B box in $L^{(n)}$. Since each choice of a normal box is independent of each other, there are $8 \cdot 3 = 24$ distinct $L^{(n)}$ when it expands over 1 qutrit.



In all other remaining cases: The ladder of B boxes starts from qutrit wire k, $1 < k < n$. In the illustration below, there are $(k-1)$ B boxes in $L^{(n)}$. Reasoning analogously as before, there are $8 \cdot 3 \cdot 9^{(k-1)} = 24 \cdot 9^{(k-1)}$ distinct $L^{(n)}$ when it expands over k qutrits.



Considering all cases of concatenating B boxes in $L^{(n)}$, the number of distinct $L^{(n)}$ is

$$\sum_{k=0}^{n-1}(24 \cdot 9^k) = 24 \cdot \sum_{k=0}^{n-1} 9^k = 3 \cdot (9^n - 1).$$

- For $M^{(n)}$, we consider all possible ways of concatenating D, E, and F boxes. According to Figure 9, there are 3 choices for E and F boxes respectively. For each D box, there are 9 choices. According to Figure 10b, there are $n-1$ D boxes in $M^{(n)}$. Since each choice of a normal box is independent of each other, there are $3 \cdot 3 \cdot 9^{(n-1)} = 9 \cdot 9^{(n-1)} = 9^n$ distinct $M^{(n)}$.

Since the choice of $L^{(n)}$ and $M^{(n)}$ is independent of each other, there are $(3 \cdot (9^n - 1)) \cdot 9^n = 3 \cdot 9^n (9^n - 1)$ distinct $M^{(n)}L^{(n)}$. By Corollary 3.10, there are exactly 6 scalars in $\mathcal{C}_n$. According to the normal form outlined in Figure 11, we can calculate the total number of distinct $N^{(n)}$:

$$6 \cdot \prod_{k=1}^{n} 3(9^k - 1)9^k.$$

Since they are in one-to-one correspondence with the elements of the n-qutrit Clifford group, this completes the proof. □

C A Complete Set of Box Relations

In Sections 3 and 4, we define a normal form for $\mathcal{C}_n$ and show how to normalize an arbitrary n-qutrit Clifford operator. Then, our problem of finding a complete set of Clifford relations reduces to finding a complete set of rewrite rules such that we can push all possible dirty gates through the normal boxes of Figure 9. Since we only need to consider the cases where multi-qutrit (derived) generators acting on adjacent qutrits, each box relation involves at most three qutrits. Therefore, the 2 zero-qutrit phase relations in Appendix C.1, the 34 single-qutrit box relations in Appendix C.2, the 173 two-qutrit box relations in Appendix C.3, and the 171 three-qutrit box relations in Appendix C.4 together form a complete set of box relations for $\mathcal{C}_n$.

C.1 Zero-Qutrit Box Relations

For $n = 0$, all Pauli operators are phases. By Corollary B.3, the Clifford operators are phases of the form $(-\omega)^t$, $t \in \mathbb{Z}_6$. Let $-1 = (-\omega)^3$ and $\omega = (-\omega)^4$, then (16) gives the complete set of box relations of $\mathcal{C}_0$.

$$(-1)^2 = 1, \qquad \omega^3 = 1. \tag{16}$$

In Appendices C.2 to C.4, we express box relations parametrically. This reveals some redundancy in the box relations, allowing for more efficient relation reduction in the supplement [22]. This is also an effective approach for streamlining the derivation of multiple box relations at once.

C.2 Single-Qutrit Box Relations

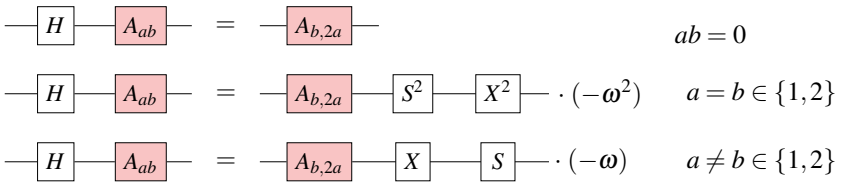


Figure 18: Pushing H through an A box.

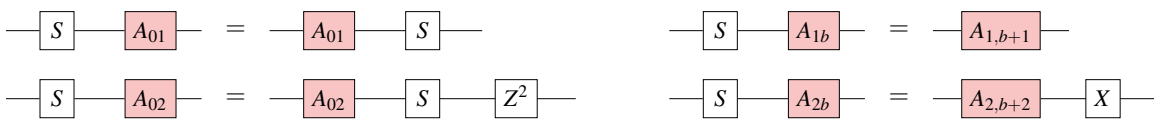


Figure 19: Pushing S through an A box, $b \in \mathbb{Z}_3$.

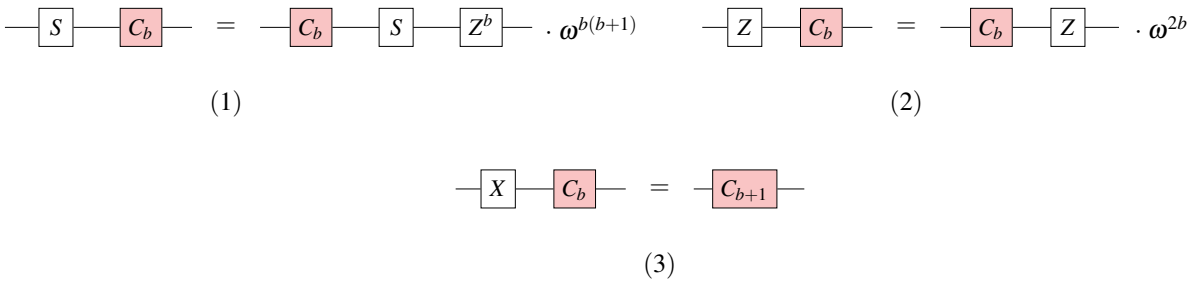


Figure 20: Pushing S, Z, or X through a C box, $b \in \mathbb{Z}_3$.

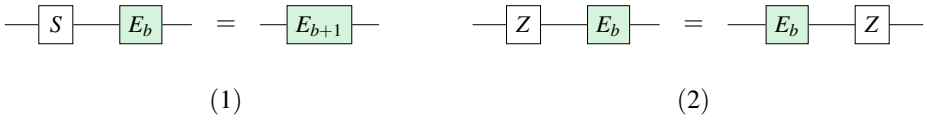


Figure 21: Pushing S or Z through an E box, $b \in \mathbb{Z}_3$.

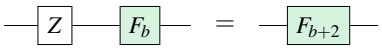


Figure 22: Pushing Z through an F box, $b \in \mathbb{Z}_3$.

C.3 Two-Qutrit Box Relations

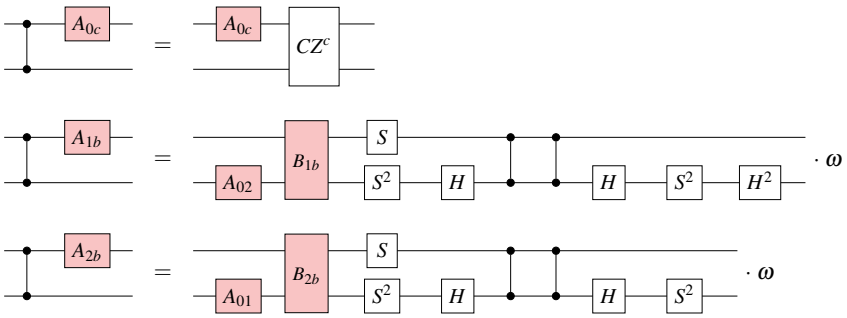


Figure 23: Pushing CZ through an A box, $c \in \{1,2\}$, $b \in \mathbb{Z}_3$.

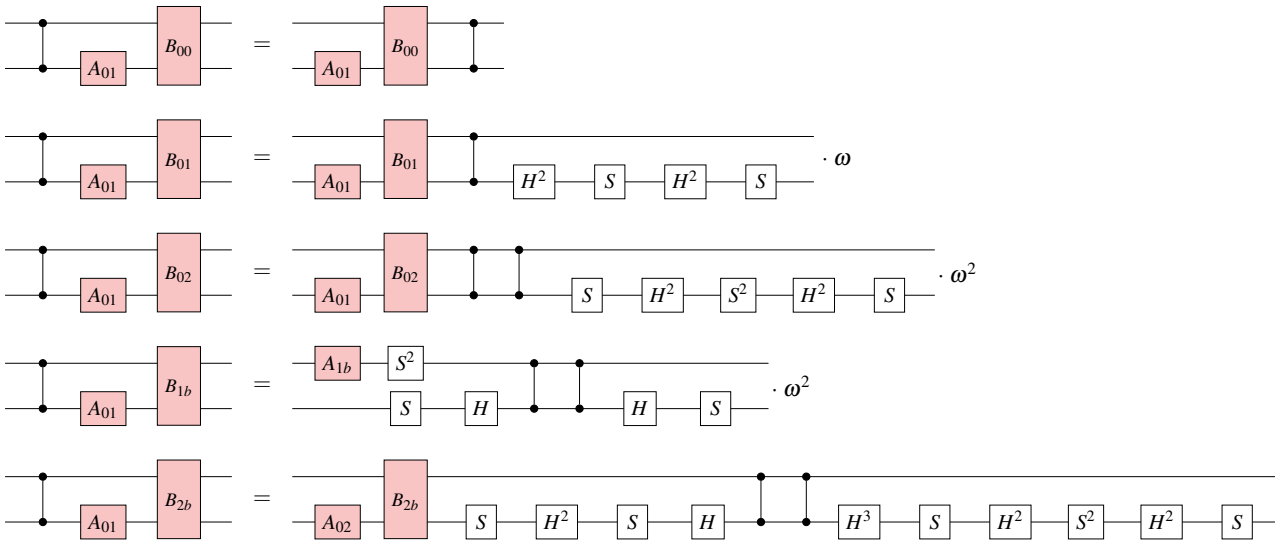


Figure 24: Pushing CZ through A_{01} and a B box, $b \in \mathbb{Z}_3$.

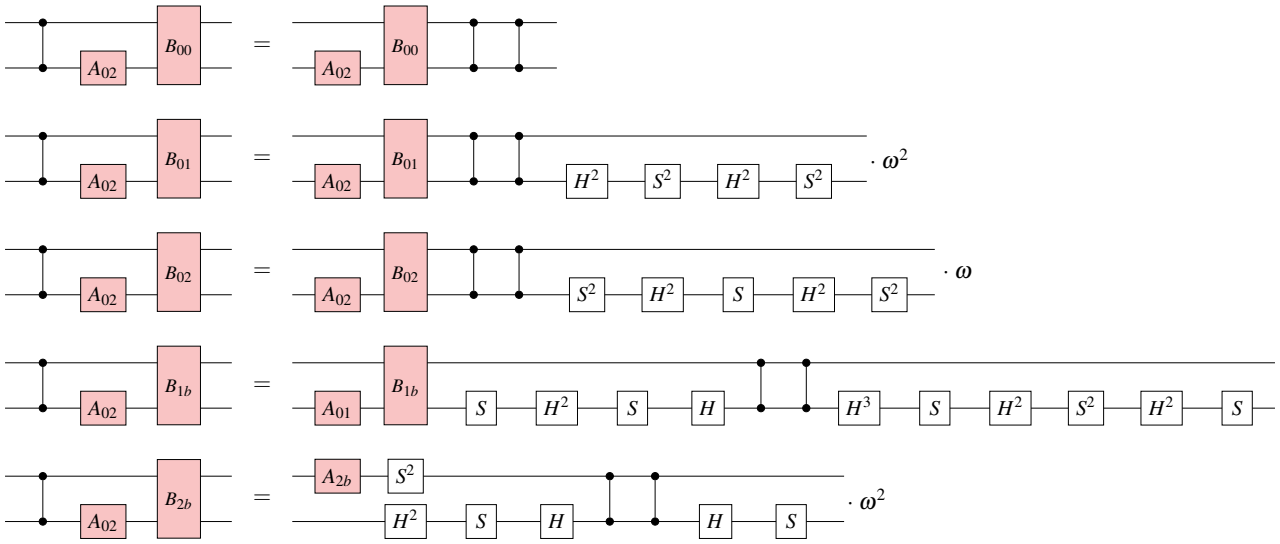


Figure 25: Pushing CZ through A_{02} and a B box, $b \in \mathbb{Z}_3$.

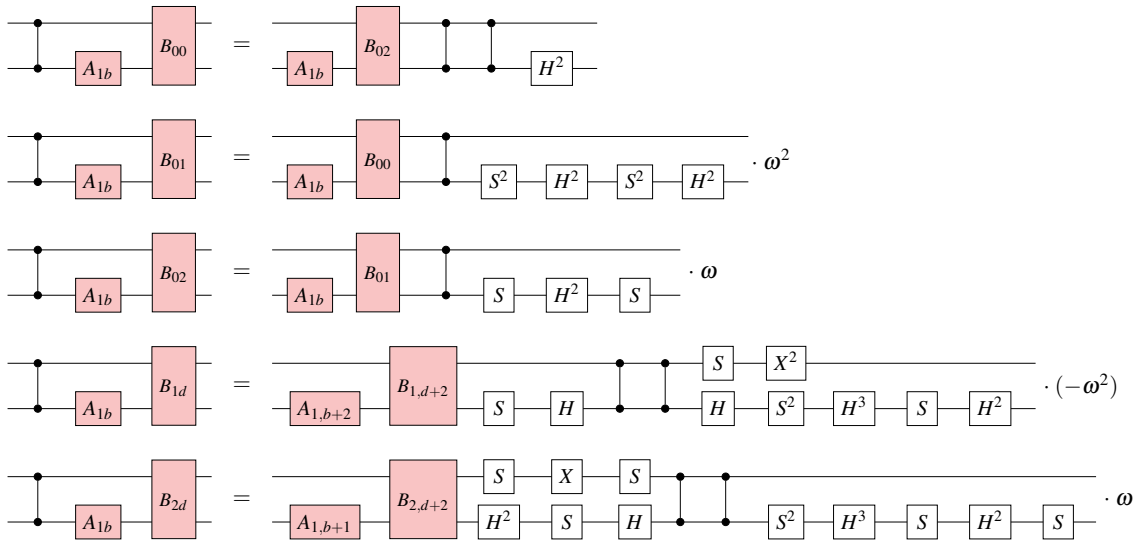


Figure 26: Pushing CZ through A_{1b} and a B box, $b,d \in \mathbb{Z}_3$.

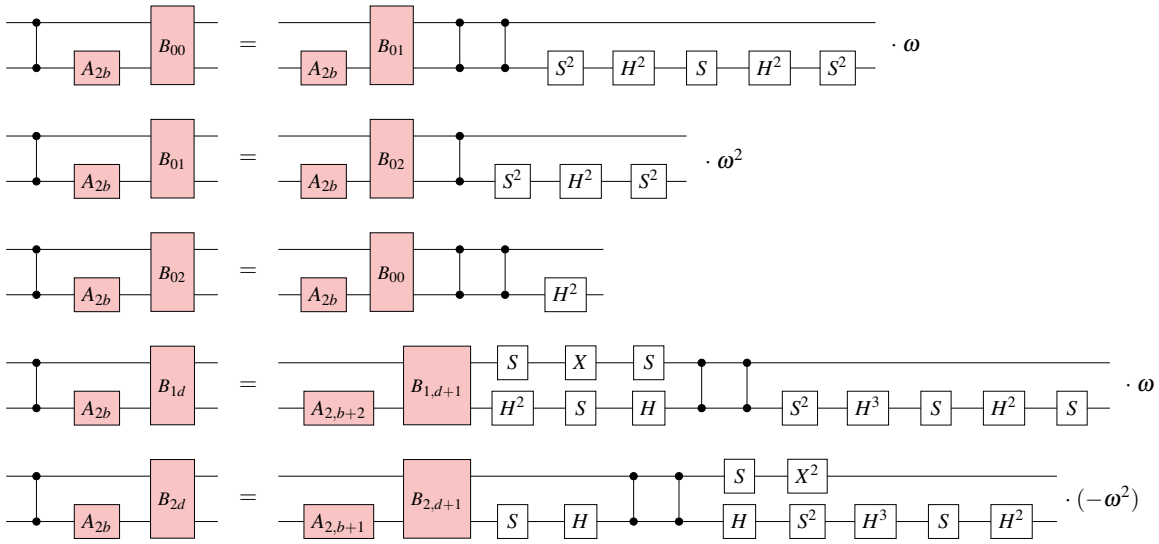


Figure 27: Pushing CZ through A_{2b} and a B box, $b,d \in \mathbb{Z}_3$.

Figure 28: Pushing $H \otimes I$ through a B box.

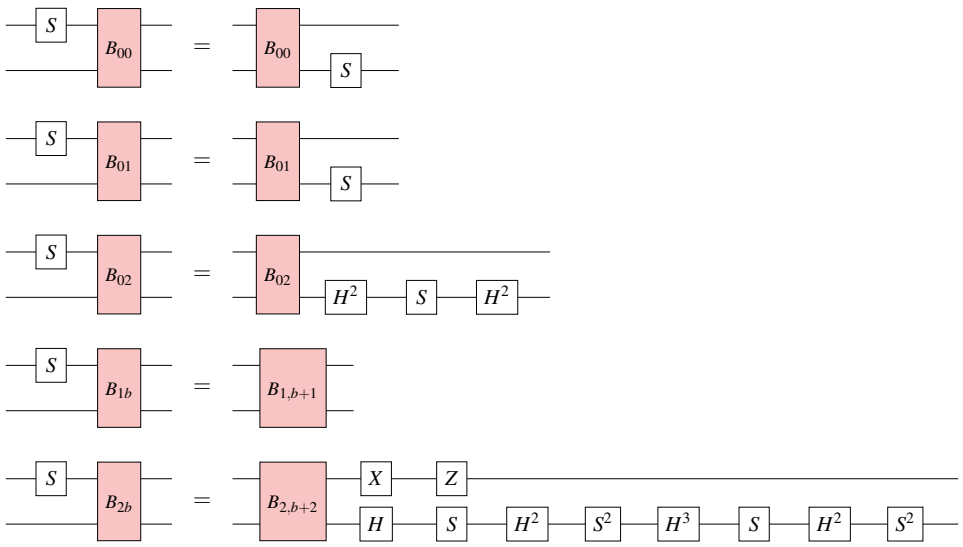


Figure 29: Pushing $S \otimes I$ through a B box, $b \in \mathbb{Z}_3$.

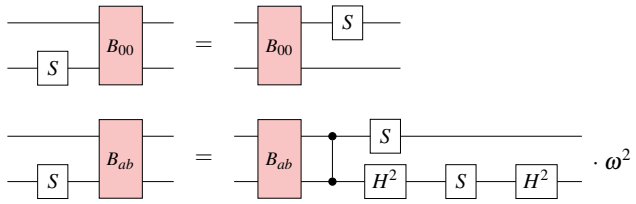


Figure 30: Pushing $I \otimes S$ through a B box, $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$.

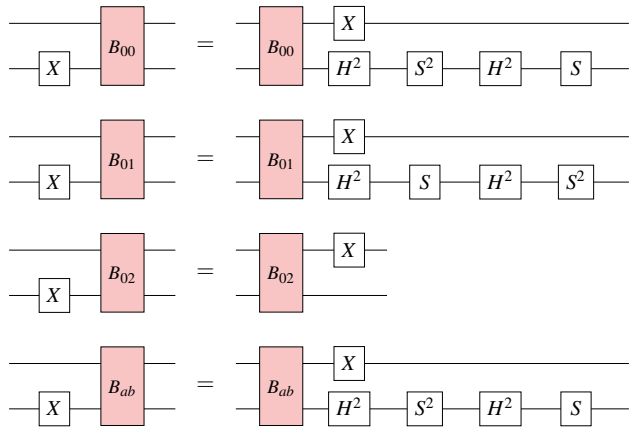


Figure 31: Pushing $I \otimes X$ through a B box, $a \in \{1,2\}$, $b \in \mathbb{Z}_3$.

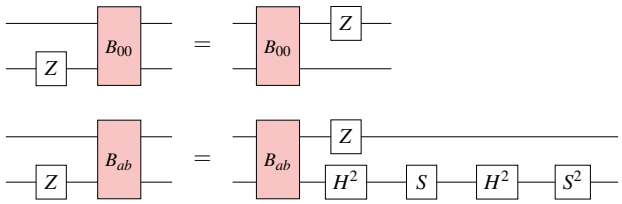


Figure 32: Pushing $I \otimes Z$ through a B box, $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$.

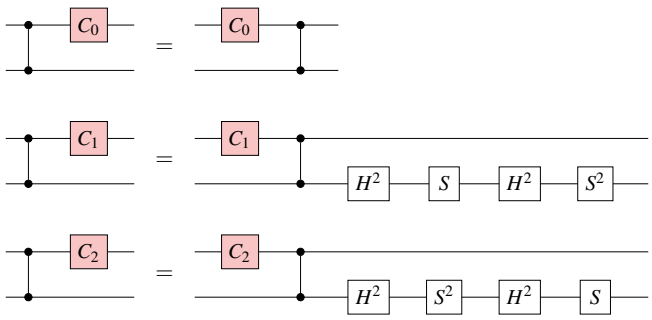


Figure 33: Pushing CZ through a C box.

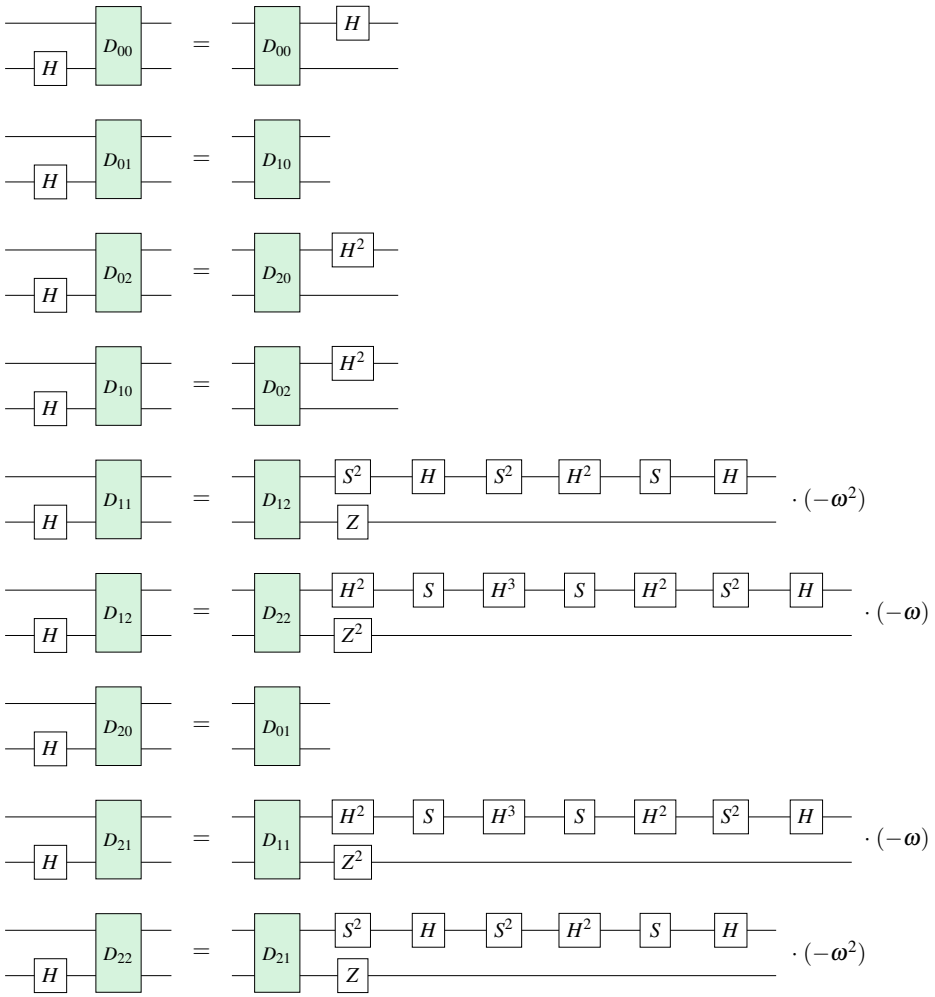


Figure 34: Pushing $I \otimes H$ through a D box.

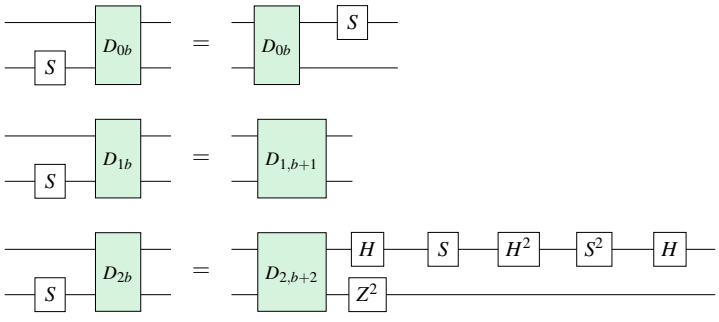


Figure 35: Pushing $I \otimes S$ through a D box, $b \in \mathbb{Z}_3$.

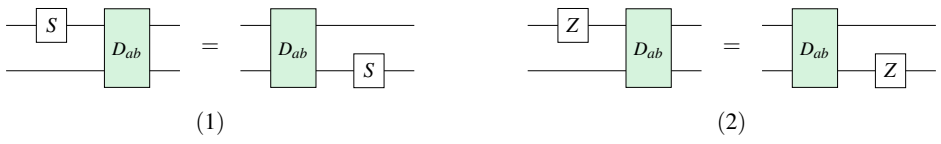


Figure 36: Pushing $S \otimes I$ or $Z \otimes I$ through a D box, $a, b \in \mathbb{Z}_3$.

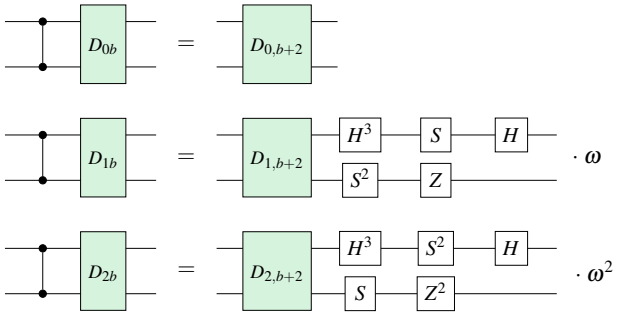


Figure 37: Pushing CZ through a D box, $b \in \mathbb{Z}_3$.

C.4 Three-Qutrit Box Relations

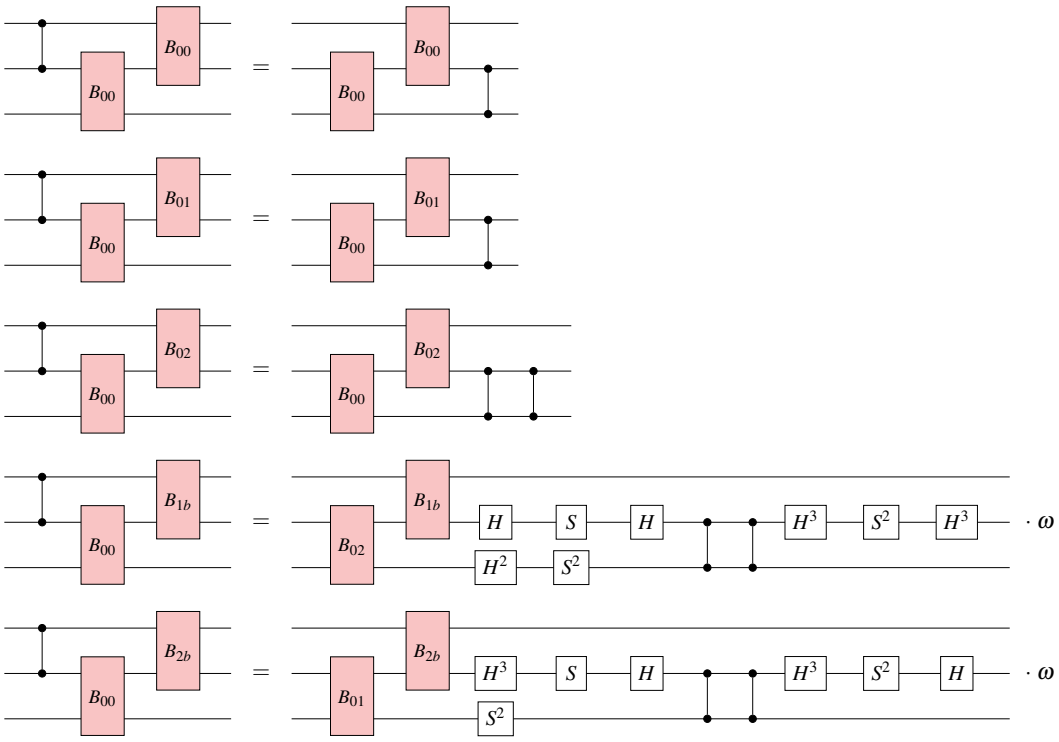


Figure 38: Pushing $\mathrm{CZ} \otimes I$ through two B boxes—B_{00} followed by another B box, $b \in \mathbb{Z}_3$.

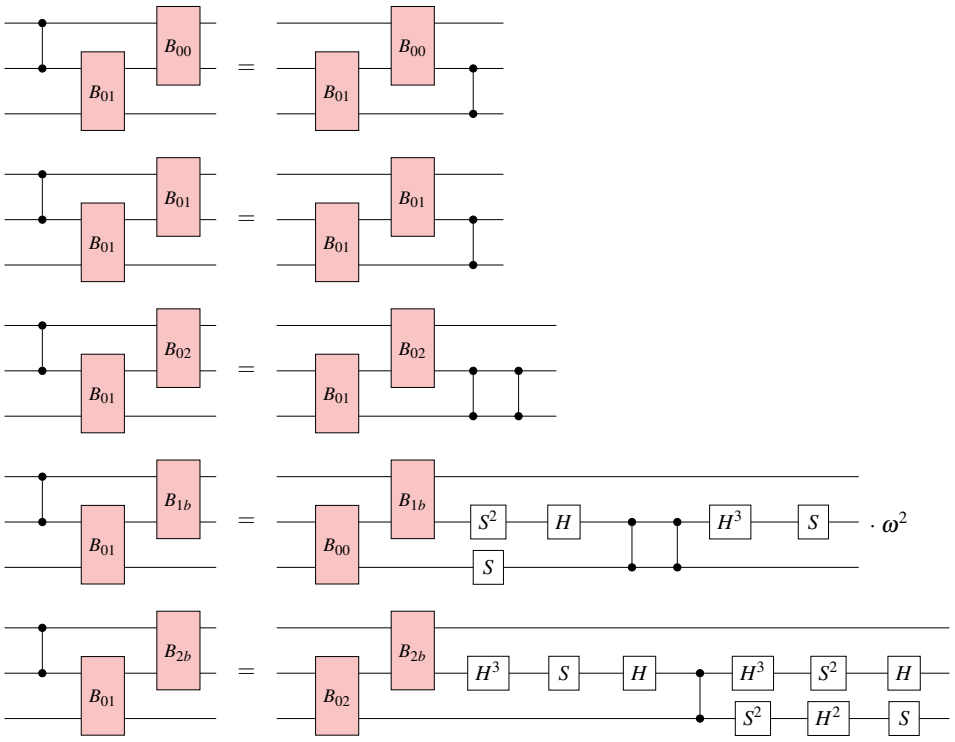


Figure 39: Pushing $\mathrm{CZ} \otimes I$ through two B boxes—B_{01} followed by another B box, $b \in \mathbb{Z}_3$.

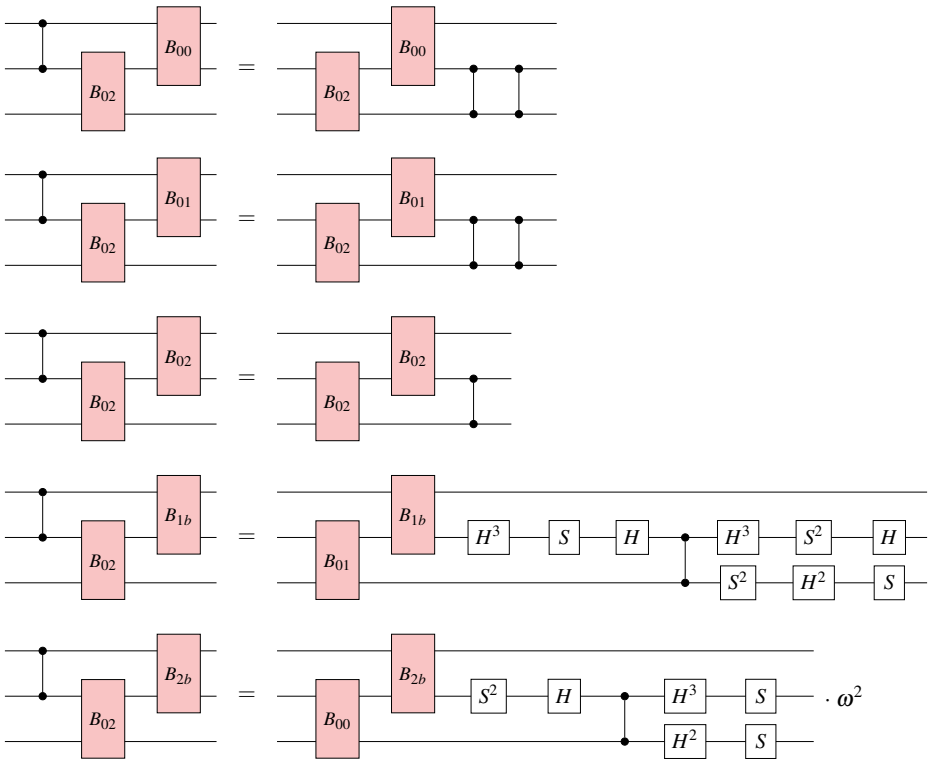


Figure 40: Pushing $\mathrm{CZ} \otimes I$ through two B boxes—B_{02} followed by another B box, $b \in \mathbb{Z}_3$.

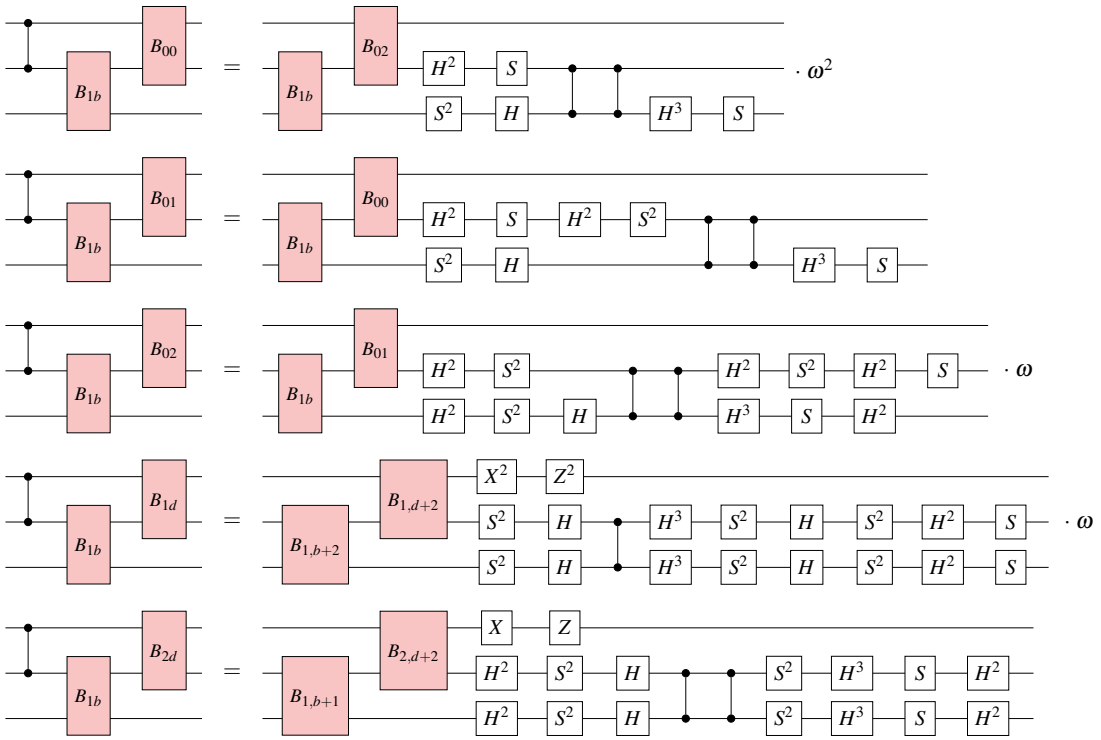


Figure 41: Pushing $\mathrm{CZ}\otimes I$ through two B boxes—B_{1b} followed by another B box, $b,d\in\mathbb{Z}_3$.

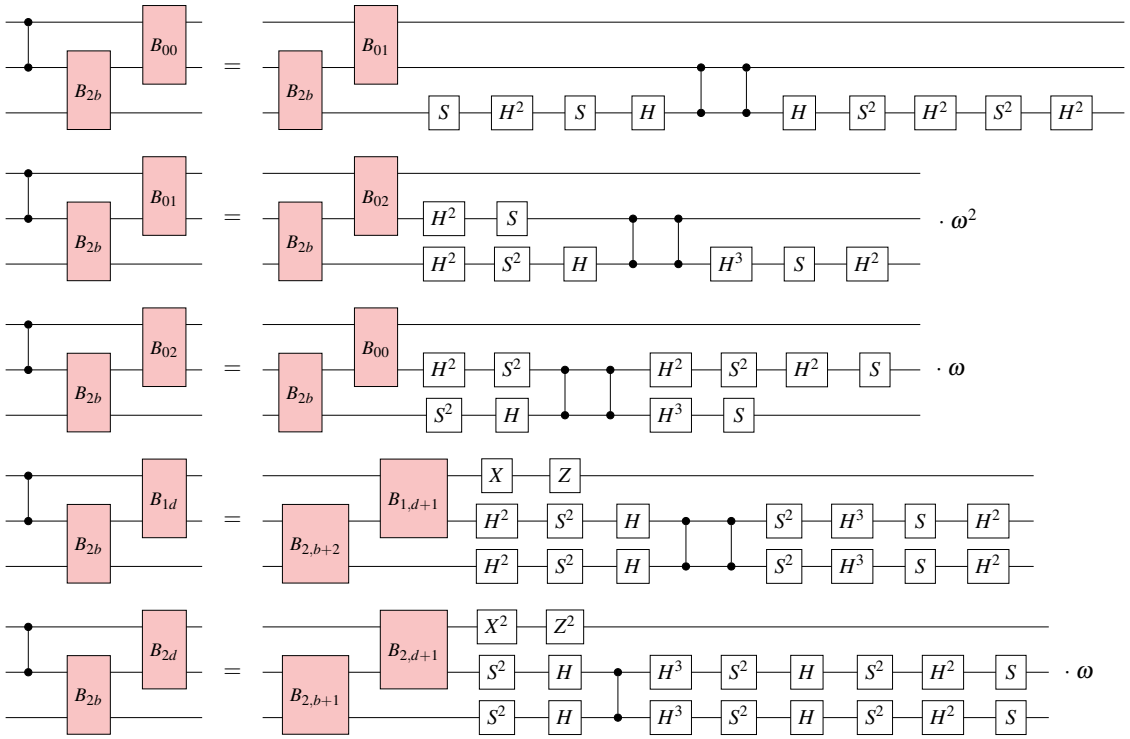


Figure 42: Pushing $\mathrm{CZ}\otimes I$ through two B boxes—B_{2b} followed by another B box, $b,d\in\mathbb{Z}_3$.

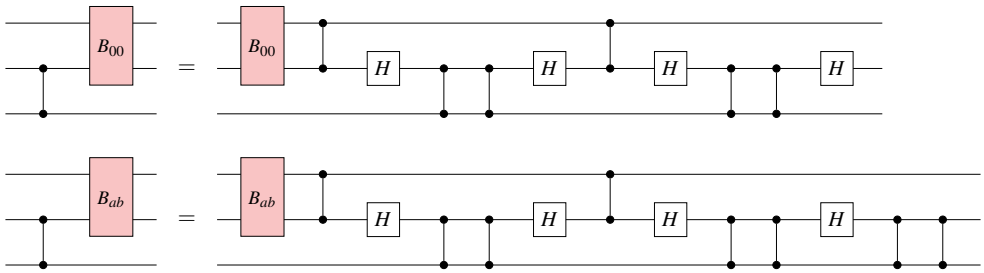


Figure 43: Pushing CZ through a B box, $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$.

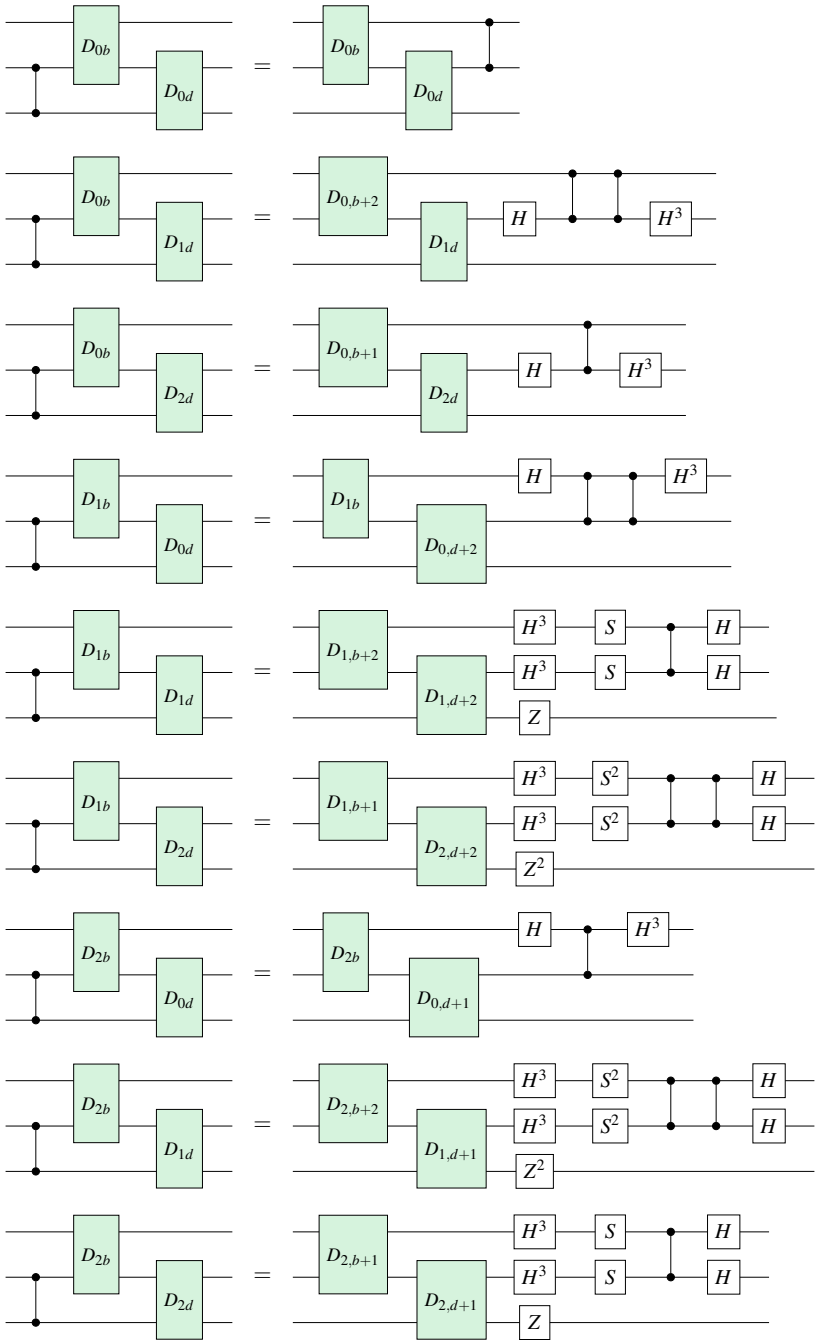
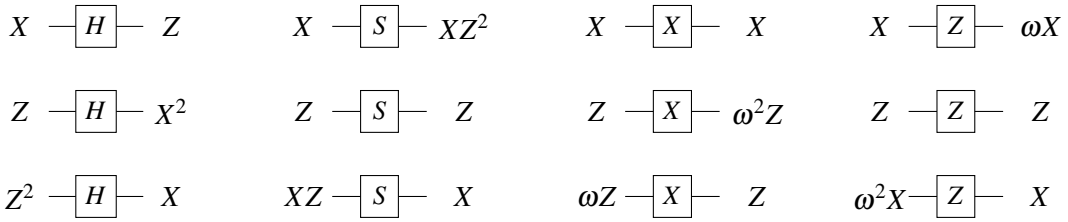


Figure 44: Pushing $I \otimes \mathrm{CZ}$ through two D boxes, $b,d \in \mathbb{Z}_3$.

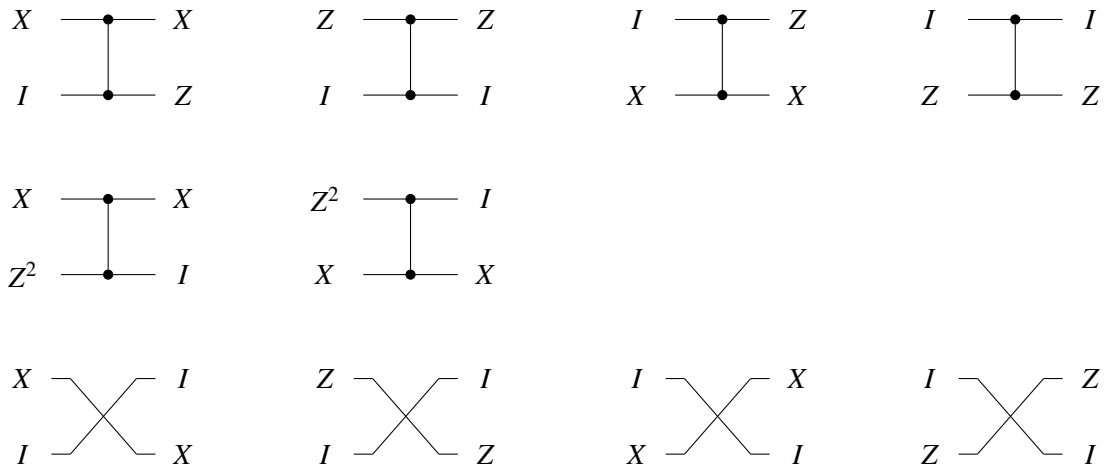
D Actions of the Clifford Generators

According to Section 2.2, we can describe a (derived) Clifford generator by its conjugation on the Pauli generators. That is, let $C \in \mathcal{C}_n$ and $P \in \mathcal{B}_n$, $CPC^\dagger = Q$, for some $Q \in \mathcal{P}_n$. For our purposes, it is also convenient to describe their inverse conjugation on the Pauli generators: $C^\dagger PC = Q'$, for some $Q' \in \mathcal{P}_n$.

Definition D.1. *H, S, X, and Z can be described by their actions on X and Z.*



Definition D.2. *CZ and SWAP can be described by their actions on $X \otimes I$, $Z \otimes I$, $I \otimes X$, and $I \otimes Z$.*



E Designing the Normal Boxes

The multi-qutrit Clifford normal form introduced in Section 3 generalizes the normal forms designed for qubit Clifford operators [24, 31]. We adopt a similar methodology and leverage the underlying mechanics of the normal form to handle the increased degrees of freedom for qutrits. Note that the choice of normal boxes is neither unique nor arbitrary. Some choices are much nicer for the Clifford normalization, because they produce simpler residual dirty gates. In this section, we outline the key ideas behind the design of these normal boxes, paving the way for generalizing multi-qutrit Clifford completeness to other odd prime dimensions.

Recall that $\mathcal{C}_n$ is the collection of n-qutrit Clifford operators generated by $-\omega$, H, S, and CZ gates through matrix multiplication and tensor product. Let $\mathcal{A} = \{-\omega, H^{(j)}, S^{(j)}, \mathrm{CZ}^{(i,i+1)};\ 1 \le j \le n,\ 1 \le i \le n-1\}$, where $-\omega$, $H^{(j)}$, $S^{(j)}$, and $\mathrm{CZ}^{(i,i+1)}$ denote the n-qutrit unitary operators:

$$
\begin{aligned}
-\omega &= (-\omega)\cdot I_{3^n}, \quad I_{3^n} \text{ is the } 3^n \times 3^n \text{ identity matrix.}\\
H^{(j)} &= \bigotimes_{\ell=1}^{n} C_\ell, \quad C_\ell = H \text{ if } \ell = j, \quad C_\ell = I_3 \text{ otherwise.}\\
S^{(j)} &= \bigotimes_{\ell=1}^{n} C_\ell, \quad C_\ell = S \text{ if } \ell = j, \quad C_\ell = I_3 \text{ otherwise.}\\
CZ^{(i,i+1)} &= I_{3^{i-1}} \otimes \mathrm{CZ} \otimes I_{3^{n-i-1}}.
\end{aligned}
$$

Every element of $\mathcal{A}$ is called a *syllable*. We use $\mathcal{A}^*$ to denote the set of all words formed by the syllables in $\mathcal{A}$. These words are composed in diagrammatic order. For $1 \le k \le m$, if $\mathbf{W} = A_1 \ldots A_m$ with $A_k \in \mathcal{A}$, we call $\mathbf{W}$ a *Clifford word over* $\mathcal{A}$, and note that $\mathbf{W} \in \mathcal{A}^*$. Let $[\![A_k]\!]$ denote the matrix representation of A_k. Then the word $\mathbf{W}$ can be *interpreted* as a unitary in $\mathcal{C}_n$ by multiplying $[\![A_k]\!]$ in the reverse order of the syllable composition,

$$[\![\mathbf{W}]\!] = [\![A_1 \ldots A_m]\!] = [\![A_m]\!] \cdots [\![A_1]\!].$$

To normalize $\mathbf{W}$, we first append it on the right with the normal form of the identity operator, as given in (5). Figures 13 and 45 show that Clifford normalization is reduced to one of the two cases: pushing an H or S gate into a Clifford normal form; or pushing a CZ gate into a Clifford normal form. Locally, it suffices to know how to push a Clifford gate through each of the normal boxes specified in Figure 9.

Based on Figures 9 and 14, Figures 52 to 54 summarize the 'allowed' Clifford operators before B, C, D, E, and F boxes. This, on the other hand, introduces the dirty gate restrictions when pushing through a normal box. To ensure the Clifford normalization will eventually terminate, we need to account for these restrictions when designing the normal boxes.

As discussed in Section 4, after pushing a Clifford operator g through a normal box M, the updated normal box M' is uniquely determined by M and g. This is due to the unique box actions specified in the second column of Figure 8. M' is followed by a sequence of Clifford gates, referred to as the residual dirty gates, dir, and it is determined by g, M, and M' collectively. More generally, when M is a certain combination of normal boxes, Procedure E.1 provides a recipe for finding dir by tracking the Clifford conjugation on the Pauli generators.

Procedure E.1. *Let $n \in \mathbb{N}$ and $g \in \mathcal{C}_n$. Let M be some combination of normal boxes and M' be the updated normal boxes after pushing g through M. Based on the dirty normal form illustrated in Figure 13, n can be either 1, 2, or 3, depending on the choice of M. We can find the residual dirty gates* dir *as follows.*

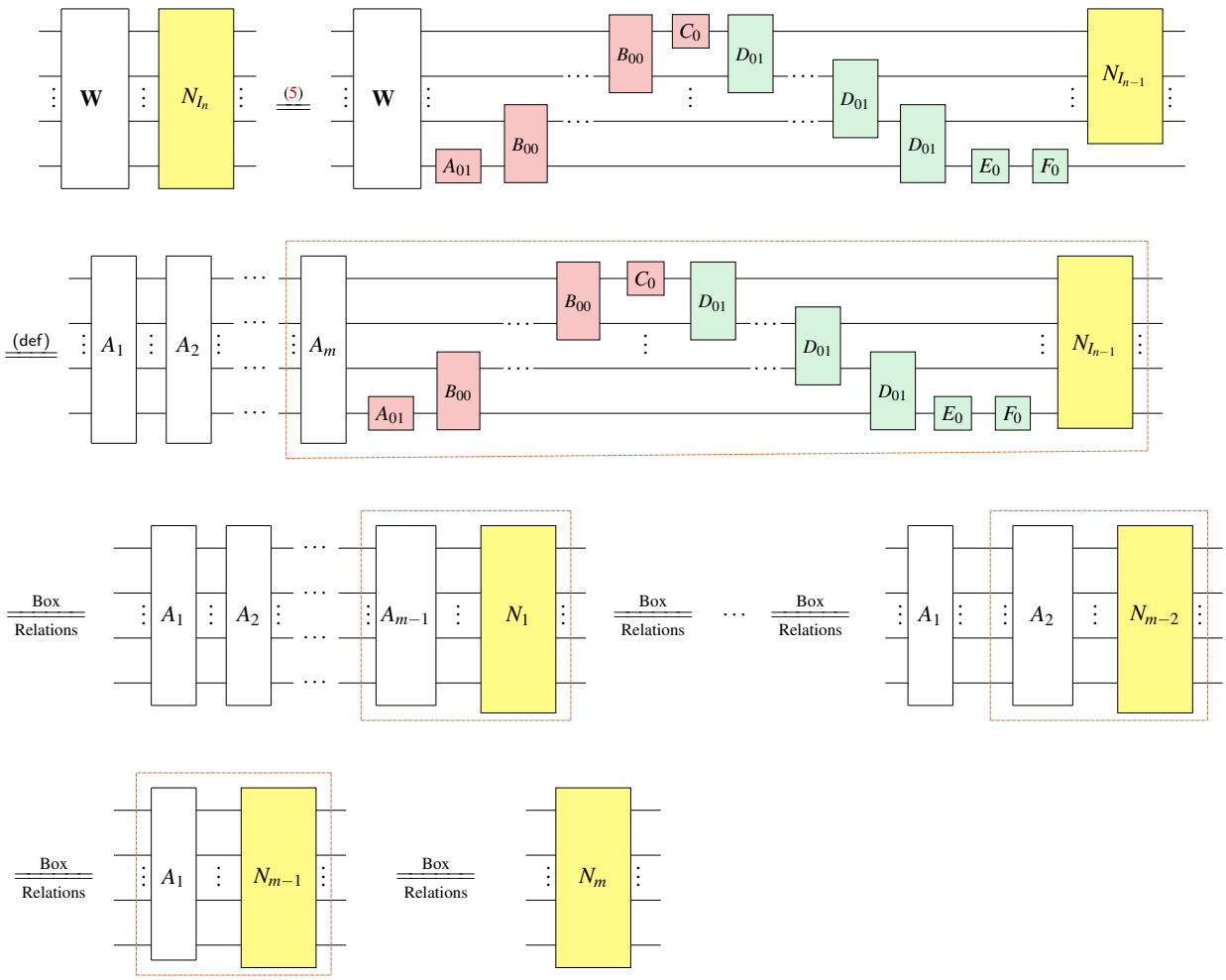
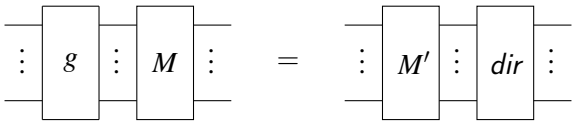
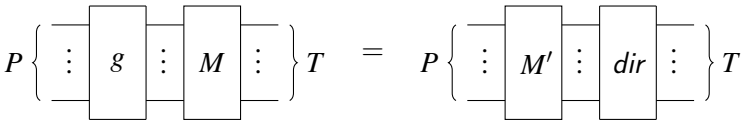


Figure 45: $\mathbf{W} = A_1 \ldots A_m$ is a Clifford word composed of m syllables. Each syllable is either a single-qutrit H or S gate, or a two-qutrit CZ gate acting on adjacent qutrits. After pushing A_m into the normal form of the n-qutrit identity operator, N_{I_n}, this normal form is updated to a new normal form N_1. Then, A_{m-1} is pushed into N_1 and so on and so forth. The normalization process reduces to pushing each Clifford syllable A_k into the updated Clifford normal form.

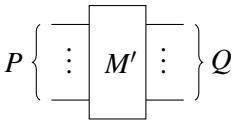


(17)

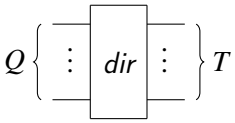
1. *Let $T \in \{X^{(j)}, Z^{(j)};\ 1 \le j \le n\}$. Find the preimage of T in the lefthand side of (17). Let it be P.*



2. *Find the image of P in M'. Let it be Q.*



3. *Then the automorphism of dir is given by Q and T.*



Next, we discuss the design principles behind the specific implementation of the normal boxes in Figure 9. In Appendix E.1, we consider the case of normalizing a single-qutrit Clifford operator, which boils down to pushing an H or S gate through the A, C, E, and F boxes. In Appendix E.2, we consider the case of normalizing a multi-qutrit Clifford operator, which boils down to pushing an H, S or CZ gate through the A, B, C, and D boxes. In both cases, our choice of normal boxes satisfy the dirty gate restrictions outlined in Figures 52 to 54.

E.1 Designing the Single-Qutrit Normal Boxes

In Section 3, we introduce four types of single-qutrit normal boxes. Figure 46 summarizes their unique actions on Pauli operators. These properties completely determine all of the boxes, except for the A boxes. In particular, since a C_b box maps X to X and $\omega^b Z$ to Z, it must be a power of X gates. Since an E_b box maps XZ^b to X and Z to Z, it must be a power of S gates. For all $i \in \mathbb{Z}_3$, since an F_b box maps $\omega^b XZ^i$ to XZ^i and Z to Z, it maps $\omega^b X$ to X. Hence, an F_b box must be a power of Z gates. By Definition D.1, $C_b = X^b$, $E_b = S^b$, and $F_b = Z^{2b}$, which gives us the construction of C, E, and F boxes in Figure 9.

Based on this, Figure 47 presents a generic construction of a single-qutrit Clifford normal form N, and Figure 48 summarizes the dirty gate restrictions when pushing an H or S gate through N.

We thus see that A boxes must be able to handle arbitrary dirty gates at the input, and the residual dirty gates must not contain solitary H gates—these can only be absorbed by the subsequent normal boxes if they appear as part of a Pauli X or Z gate. Therefore, to find A boxes that are suitable for normalization, we need to consider the case when pushing an H or S gate through A_{ab}. The updated A box is uniquely determined by the tuple (a,b,g), $g \in \{H,S\}$.

Lemma E.2. *For all $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$, there exist dir, dir′ $\in \mathcal{C}_1$ such that*

Proof. By the uniqueness of normal form and the required action of A boxes in Figure 46, it suffices to find the preimage of $X^a Z^b$ under the action of H and S gates.

$$H^\dagger (X^a Z^b) H = (H^\dagger X^a H)(H^\dagger Z^b H) \overset{\text{Definition D.1}}{=\!=} Z^{-a} X^b \overset{\text{Definition D.1}}{=\!=} \omega^{-ab} X^b Z^{-a}.$$

Allowed indices	Required action on Pauli operators	Additional action on Pauli operators
$(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$	$X^a Z^b$ — A_{ab} — Z	/
$b \in \mathbb{Z}_3$	$\omega^b Z$ — C_b — Z	X — C_b — X
$b \in \mathbb{Z}_3$	XZ^b — E_b — X	Z — E_b — Z
$b \in \mathbb{Z}_3$	$\omega^b XZ^i$ — F_b — XZ^i for all $i \in \mathbb{Z}_3$	Z — F_b — Z

Figure 46: Single-qutrit normal boxes and their unique actions on Pauli operators. The second column shows the required actions. The third column shows the additional actions that we choose to simplify the normalization process.

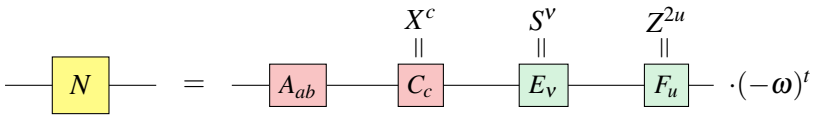


Figure 47: For $(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3 \setminus \{(0,0)\}$, $c, v, u \in \mathbb{Z}_3$, and $t \in \mathbb{Z}_6$, a generic construction of the normal form of a single-qutrit Clifford operator. This is based on the close-up view of the Clifford normal form in Figure 14 and the discussion above.

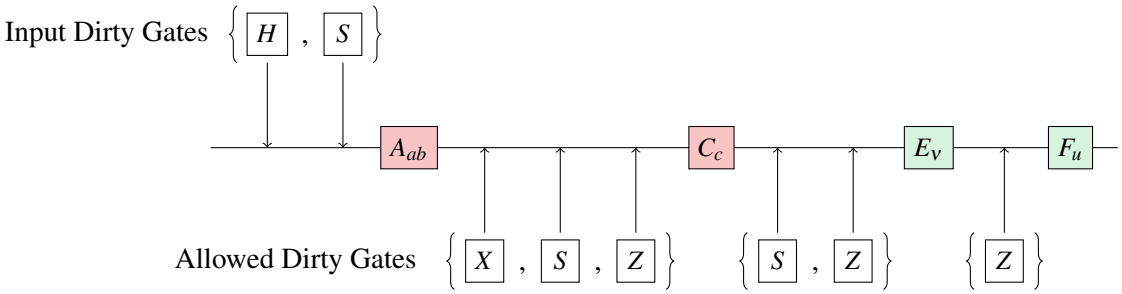
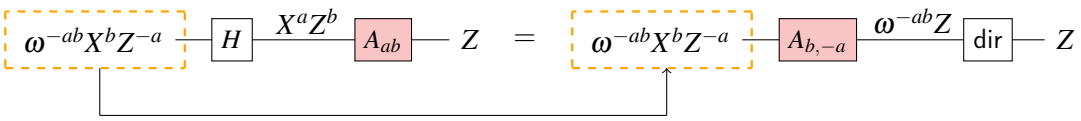


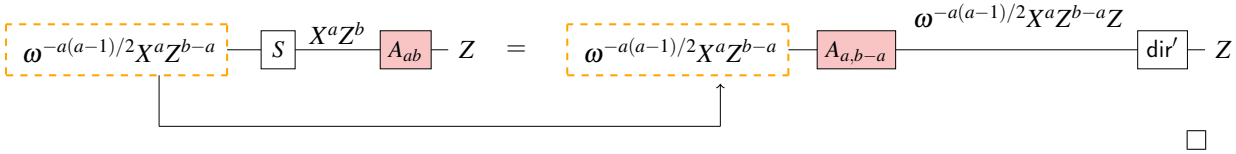
Figure 48: Dirty gate restrictions when pushing an H or S gate through a single-qutrit normal form.

Hence, after pushing H gate through an A_{ab} box, the updated A box must be of the form $A_{b,-a}$.



$$S^\dagger(X^aZ^b)S = (S^\dagger X^a S)(S^\dagger Z^b S) \overset{\text{Definition D.1}}{=\!=} \left((XZ^{-1})\cdots(XZ^{-1})\right)Z^b$$
$$\overset{\text{Definition D.1}}{=\!=} \omega^{\frac{-a(a-1)}{2}} X^a Z^{-a} Z^b = \omega^{\frac{-a(a-1)}{2}} X^a Z^{b-a}.$$

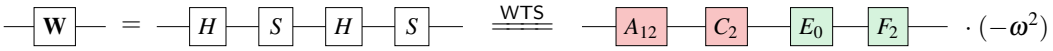
Hence, after pushing an S gate through an A_{ab} box, the updated A box must be of the form $A_{a,b-a}$.



□

Combining Lemma E.2 and procedure E.1 as well as the choice of A boxes in Figure 9, we can find all single-qutrit A box relations. They are demonstrated in Figures 18 and 19. The dirty gates after pushing an H or S through an A box satisfy the restrictions outlined in Figure 48.

Putting everything together, Figure 49 demonstrates how to normalize a single-qutrit Clifford operator $\mathbf{W}$, as shown below. We begin by appending $\mathbf{W}$ on the right with the normal form of the identity operator given in (4). Then we iteratively push every Clifford gate through the normal form, by applying the rules in Appendix C.2.



E.2 Designing the Two-Qutrit Normal Boxes

It is much more involved in finding the appropriate constraints for designing the two-qutrit normal boxes. In Figure 8, we introduce two types of such boxes, whose distinct actions on Pauli operators are summarized in Figure 50. Both actions defined for the D box are essential to ensure the normal form behaves correctly. For the B box, however, the second action is identified as a desirable property, as we will explain later. Note that, in all cases, we specify at most two Pauli actions per box. As a result, the stabilizer tableaus are underspecified, leaving a large design space for the normal boxes. To make it more tractable, we introduce a few guiding principles.

Recall that in Figure 46, an A_{ab} box sends X^aZ^b to Z. This is the same as the Pauli evolution happening on the top input wire of B boxes. To not reinvent the wheel, it is desirable to construct B boxes using A boxes. Moreover, the required action of B and D boxes on Pauli operators is almost vertically symmetric of each other. Ideally, we would like to design D boxes based on how we design B boxes. For example, we also want to use A boxes to construct D boxes.

Finally, among all possible box relations, Figure 51 shows the most complicated case where a CZ is pushed through two consecutive B or D boxes. In each case, there are 81 box relations. It is desirable to choose appropriate B and D boxes such that the dirty gates in each one of the 162 box relations are as simple as possible.

As before, we can uniquely determine the updated B and D boxes after pushing a CZ through them.

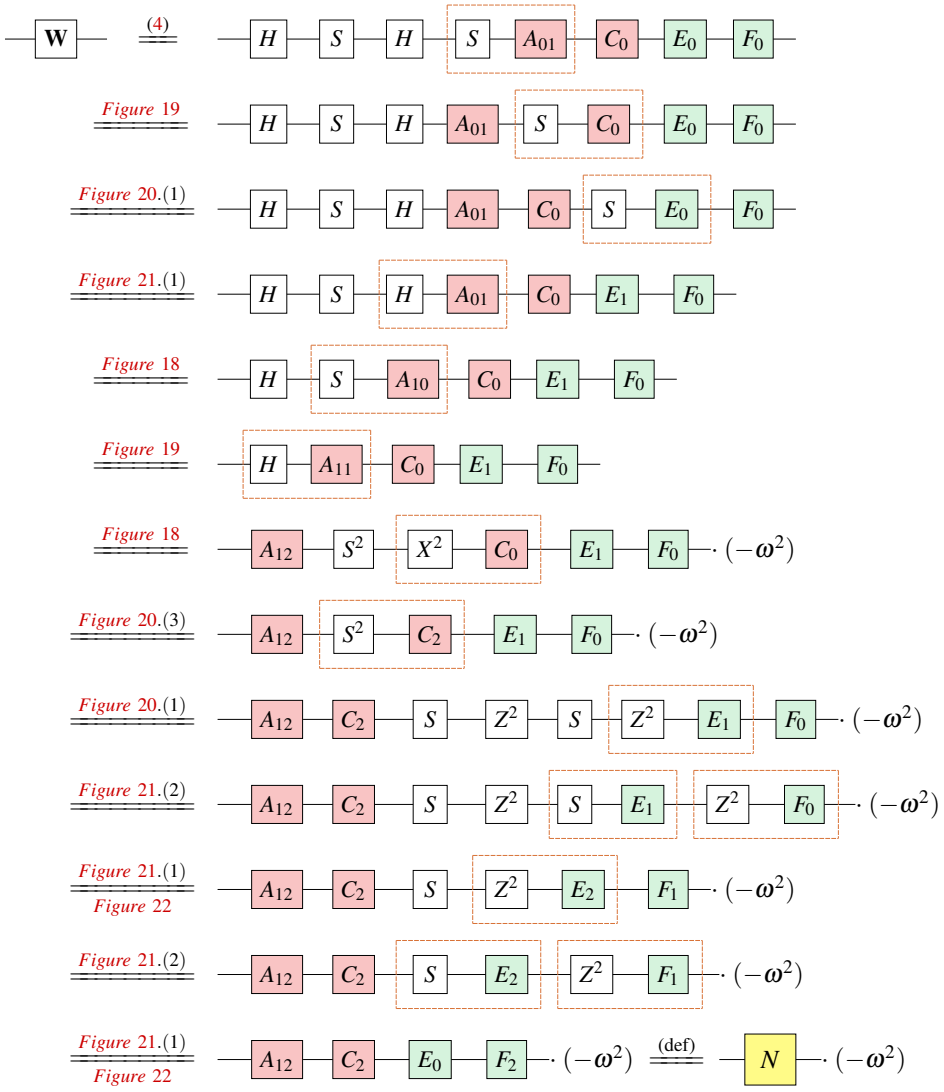
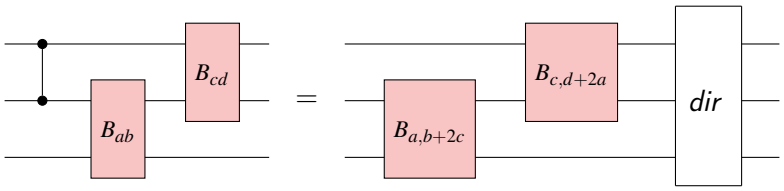
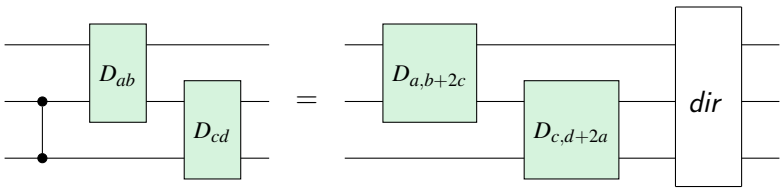


Figure 49: Normalize a single-qutrit Clifford operator. Each orange dashed box highlights the subcircuit where a box relation is applied.

Lemma E.3. *For $a,b,c,d \in \mathbb{Z}_3$ and dir $\in \mathcal{C}_3$,*



(18)



(19)

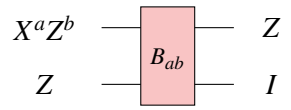
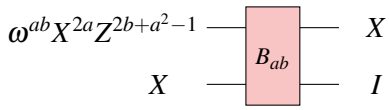
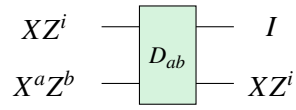
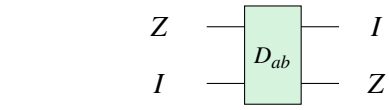
Allowed indices	Required action on Pauli operators	Additional action on Pauli operators
$(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3$		
$(a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3$	 for all $i \in \mathbb{Z}_3$	

Figure 50: Two-qutrit normal boxes and their unique actions on Pauli operators. The second column shows the required actions. The third column shows the additional actions that we choose to simplify the normalization process.

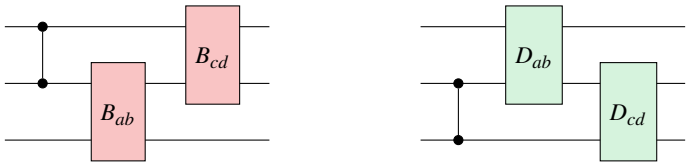
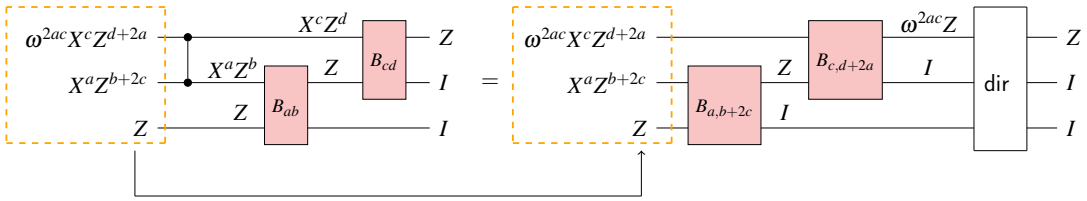


Figure 51: For all $a,b,c,d \in \mathbb{Z}_3$, push a CZ through two consecutive B or D boxes.

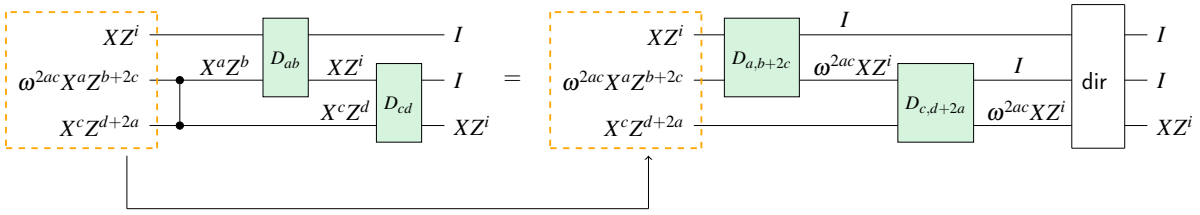
Proof. To determine the updated indices of the B boxes in the righthand side of (18), it suffices to find the preimage of $Z \otimes I \otimes I$ in the lefthand side of (18). By the uniqueness of normal form and the required action of B boxes in Figure 50, our problem is reduced to finding the preimage of $X^cZ^d \otimes X^aZ^b$ in the CZ circuit.

$$\begin{aligned}
\mathrm{CZ}^\dagger(X^cZ^d \otimes X^aZ^b)\mathrm{CZ} &= \mathrm{CZ}^\dagger\left((X^c \otimes I)(Z^d \otimes I)(I \otimes X^a)(I \otimes Z^b)\right)\mathrm{CZ} \\
&= \left(\mathrm{CZ}^\dagger(X^c \otimes I)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(Z^d \otimes I)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(I \otimes X^a)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(I \otimes Z^b)\mathrm{CZ}\right) \\
&= \left(\mathrm{CZ}^\dagger(X \otimes I)\mathrm{CZ}\right)^c\left(\mathrm{CZ}^\dagger(Z \otimes I)\mathrm{CZ}\right)^d\left(\mathrm{CZ}^\dagger(I \otimes X)\mathrm{CZ}\right)^a\left(\mathrm{CZ}^\dagger(I \otimes Z)\mathrm{CZ}\right)^b \\
&\overset{\text{Definition D.2}}{=\!=} (X \otimes Z^2)^c(Z \otimes I)^d(Z^2 \otimes X)^a(I \otimes Z)^b \\
&= (X^c \otimes Z^{2c})(Z^d \otimes I)(Z^{2a} \otimes X^a)(I \otimes Z^b) \\
&= X^cZ^dZ^{2a} \otimes Z^{2c}X^aZ^b \overset{\text{Definition D.1}}{=\!=} \omega^{2ac}X^cZ^{d+2a} \otimes X^aZ^{b+2c}.
\end{aligned}$$



To determine the updated indices of the D boxes in the righthand side of (19), it suffices to find the preimage of $I\otimes I\otimes XZ^i$, for $i\in\mathbb{Z}_3$, in the lefthand side of (19). By the uniqueness of normal form and the required action of D boxes in Figure 50, our problem is reduced to finding the preimage of $X^aZ^b\otimes X^cZ^d$ in the CZ circuit.

$$\begin{aligned}
\mathrm{CZ}^\dagger(X^aZ^b\otimes X^cZ^d)\mathrm{CZ} &= \mathrm{CZ}^\dagger\Big((X^a\otimes I)(Z^b\otimes I)(I\otimes X^c)(I\otimes Z^d)\Big)\,\mathrm{CZ}\\
&= \left(\mathrm{CZ}^\dagger(X^a\otimes I)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(Z^b\otimes I)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(I\otimes X^c)\mathrm{CZ}\right)\left(\mathrm{CZ}^\dagger(I\otimes Z^d)\mathrm{CZ}\right)\\
&= \left(\mathrm{CZ}^\dagger(X\otimes I)\mathrm{CZ}\right)^a\left(\mathrm{CZ}^\dagger(Z\otimes I)\mathrm{CZ}\right)^b\left(\mathrm{CZ}^\dagger(I\otimes X)\mathrm{CZ}\right)^c\left(\mathrm{CZ}^\dagger(I\otimes Z)\mathrm{CZ}\right)^d\\
&\overset{\text{Definition D.2}}{=\!=} (X\otimes Z^2)^a(Z\otimes I)^b(Z^2\otimes X)^c(I\otimes Z)^d\\
&= (X^a\otimes Z^{2a})(Z^b\otimes I)(Z^{2c}\otimes X^c)(I\otimes Z^d)\\
&= X^aZ^bZ^{2c}\otimes Z^{2a}X^cZ^d \overset{\text{Definition D.1}}{=\!=} \omega^{2ac}X^aZ^{b+2c}\otimes X^cZ^{d+2a}.
\end{aligned}$$



□

According to the dirty gate restrictions in Figures 53 and 54, our proposed B box construction is suitable for Clifford normalization if there is no single H gate acting on the top wire of dir. One way to do this is to show that dir is not entangling the first qutrit with the other qutrits, and that only Pauli gates appear on the first qutrit. To this end, let us consider the notion of *separable operators*. We will use their properties to decide if the normal boxes are designed appropriately.

Definition E.4. *Let $k_1,k_2\in\mathbb{N}^{>0}$. When a Clifford operator $C\in\mathcal{C}_{k_1+k_2}$ is a tensor product of two Clifford operators acting on the first k_1 and the remaining k_2 qutrits respectively, we say C is* separable.

When C is separable, it cannot propagate Paulis from the first k_1 qutrits to the other k_2 qutrits. In Proposition E.7, we show that the converse is true.

Definition E.5. *For any subset $\mathcal{A}\subset\mathcal{M}_n(\mathbb{C})$, its commutant is defined as*

$$\mathcal{A}'=\{B\in\mathcal{M}_n(\mathbb{C});\ [B,A]=0 \text{ for all } A\in\mathcal{A}\}.$$

Note that the commutation of all matrices is just the scalars: $\mathcal{M}_n(\mathbb{C})'=\{cI_n : c\in\mathbb{C}\}$. In a composite system $\mathcal{M}_{n_1}(\mathbb{C})\otimes\mathcal{M}_{n_2}(\mathbb{C})$,

$$\left(\mathcal{M}_{n_1}(\mathbb{C})\otimes I_{n_2}\right)'=I_{n_1}\otimes\mathcal{M}_{n_2}(\mathbb{C}).$$

Here $\mathcal{M}_{n_1}(\mathbb{C})\otimes I_{n_2}$ denotes the set $\{A\otimes I_{n_2} : A\in\mathcal{M}_{n_1}(\mathbb{C})\}$. The following lemma is well known; we include a proof to keep this appendix self-contained.

Lemma E.6. *Let U be a unitary operator in $\mathcal{M}_{n_1}(\mathbb{C}) \otimes \mathcal{M}_{n_2}(\mathbb{C})$. Suppose there is a unitary operator $U_1 \in \mathcal{M}_{n_1}(\mathbb{C})$ such that*

$$U(A \otimes I_{n_2})U^\dagger = U_1 A U_1^\dagger \otimes I_{n_2} \tag{20}$$

for all $A \in \mathcal{M}_{n_1}(\mathbb{C})$. Then there is unitary operator $U_2 \in \mathcal{M}_{n_2}(\mathbb{C})$ such that $U = U_1 \otimes U_2$.

Proof. Let $W = (U_1^\dagger \otimes I_{n_2})U$. Equation (20) implies that $A \otimes I_{n_2} = W^\dagger (A \otimes I_{n_2}) W$ for all $A \in \mathcal{M}_{n_1}(\mathbb{C})$. So $W \in \left(\mathcal{M}_{n_1}(\mathbb{C}) \otimes I_{n_2}\right)' = I_{n_1} \otimes \mathcal{M}_{n_2}(\mathbb{C})$. Since W is unitary, we conclude that $W = I_{n_1} \otimes U_2$ for some unitary $U_2 \in \mathcal{M}_{n_2}(\mathbb{C})$. It follows that $U = U_1 \otimes U_2$. □

Recall from Section 2 that the set $\mathcal{B}_k$ contains all k-qutrit Pauli operators that generate the k-qutrit Pauli group $\mathcal{P}_k$, and that the set $\mathcal{C}_n$ of n-qutrit Clifford circuits is generated by H, S, CZ, and $-\omega$ via matrix multiplication and tensor product. For a composite system of $k_1 + k_2$ qutrits, we identify $\mathcal{M}_{3^{k_1+k_2}}(\mathbb{C})$ with $\mathcal{M}_{3^{k_1}}(\mathbb{C}) \otimes \mathcal{M}_{3^{k_2}}(\mathbb{C})$. Under this identification, it is easy to see that $C \otimes I_{3^{k_2}} \in \mathcal{C}_{k_1+k_2}$ if and only if $C \in \mathcal{C}_{k_1}$.

Proposition E.7. *Let $k_1, k_2 \in \mathbb{N}$, and let $C \in \mathcal{C}_{k_1+k_2}$. Suppose that for every $B \in \mathcal{B}_{k_1}$ there exists a $B' \in \mathcal{P}_{k_1}$ such that*

$$C(B \otimes I_{3^{k_2}})C^\dagger = B' \otimes I_{3^{k_2}}. \tag{21}$$

Then there exist $C_1 \in \mathcal{C}_{k_1}$ and $C_2 \in \mathcal{C}_{k_2}$ such that $C = C_1 \otimes C_2$, up to scalar.

Proof. We denote by Tr_2 the partial trace $id \otimes \mathrm{Tr}$, which traces out the second system $\mathcal{M}_{3^{k_2}}(\mathbb{C})$. Since $\mathcal{B}_{k_1}$ generates $\mathcal{P}_{k_1}$, it follows from Equation (21) that the mapping

$$\begin{aligned} \Phi : \mathcal{B}_{k_1} &\to \mathcal{M}_{3^{k_1}}(\mathbb{C}) \\ B &\mapsto \frac{1}{3^{k_2}} \mathrm{Tr}_2\left(C(B \otimes I_{3^{k_2}})C^\dagger\right) \end{aligned}$$

extends to an automorphism Φ of $\mathcal{P}_{k_1}$ such that

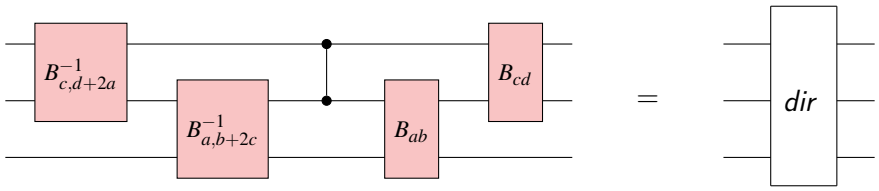
$$C(B \otimes I_{3^{k_2}})C^\dagger = \Phi(B) \otimes I_{3^{k_2}}.$$

Hence, there exists a Clifford operator $C_1 \in \mathcal{C}_{k_1}$ such that $\Phi(B) = C_1 B C_1^\dagger$ for all $B \in \mathcal{P}_{k_1}$. Since $\mathcal{P}_{k_1}$ spans $\mathcal{M}_{3^{k_1}}(\mathbb{C})$, we conclude that

$$C(B \otimes I_{3^{k_2}})C^\dagger = C_1 B C_1^\dagger \otimes I_{3^{k_2}}$$

for all $B \in \mathcal{M}_{3^{k_1}}(\mathbb{C})$. Then by Lemma E.6, $C = C_1 \otimes C_2$ for some unitary $C_2 \in \mathcal{M}_{3^{k_2}}(\mathbb{C})$. Since C and C_1 are both Clifford operators, it follows that $C_2 \in \mathcal{C}_{k_2}$. □

Lemma E.8. *The dirty gate resulting from pushing a CZ through two consecutive B boxes is separable and only has Paulis on the first qutrit:*



(22)

where dir $= C_1 \otimes C_2$ and C_1 is a Pauli.

Proof. Based on Lemma E.3, we can find the residual dirty gates after pushing a CZ through two consecutive B boxes by left-appending $\left(I\otimes B_{a,b+2c}^{-1}\right)$ and $\left(B_{c,d+2a}^{-1}\otimes I\right)$ to both sides of (18) as we see in (22).

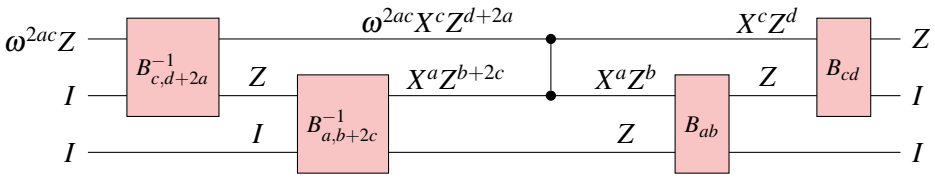
By Proposition E.7 and the single-qutrit Clifford actions in Definition D.1, dir is separable and only has Paulis on the top qutrits when dir maps $X\otimes I_9$ to $\omega^{t_1}X\otimes I_9$ and $Z\otimes I_9$ to $\omega^{t_2}Z\otimes I_9$, for $t_1,t_2\in\mathbb{Z}_3$.

First, let us find the preimage of $Z\otimes I_9$ in the lefthand side of (22). Note that

$$\left(B_{a,b+2c}^{-1}\right)^{\dagger}\left(X^aZ^{b+2c}\otimes Z\right)\left(B_{a,b+2c}^{-1}\right)=\left(B_{a,b+2c}\right)\left(X^aZ^{b+2c}\otimes Z\right)\left(B_{a,b+2c}\right)^{\dagger}\overset{\text{Figure 50}}{=\!=}Z\otimes I. \tag{23}$$

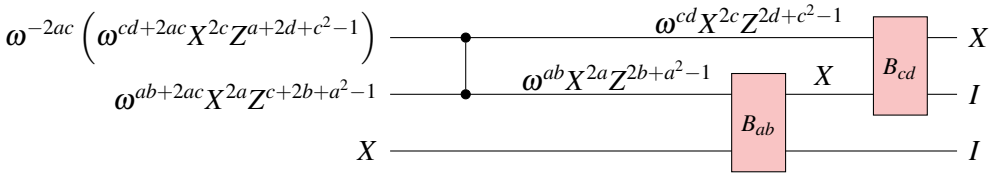
$$\left(B_{c,d+2a}^{-1}\right)^{\dagger}\left(\omega^{2ac}X^cZ^{d+2a}\otimes Z\right)\left(B_{c,d+2a}^{-1}\right)=\omega^{2ac}\left(B_{c,d+2a}\right)\left(X^cZ^{d+2a}\otimes Z\right)\left(B_{c,d+2a}\right)^{\dagger}\overset{\text{Figure 50}}{=\!=}\omega^{2ac}Z\otimes I. \tag{24}$$

(23) and (24) imply that (reading the updating of the Paulis from right-to-left):



(25)

Next, let us find the preimage of $X\otimes I_9$ in the lefthand side of (22). According to the third column of Figure 50, we have



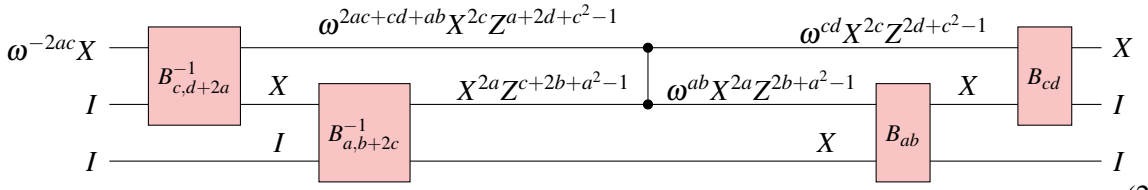
Since $2(b+2c)+a^2-1=2b+4c+a^2-1\equiv_3 2b+c+a^2-1$, we have

$$\begin{aligned}&\left(B_{a,b+2c}^{-1}\right)^{\dagger}\left(\omega^{a(b+2c)}X^{2a}Z^{c+2b+a^2-1}\otimes X\right)\left(B_{a,b+2c}^{-1}\right)\\&=\left(B_{a,b+2c}\right)\left(\omega^{a(b+2c)}X^{2a}Z^{2(b+2c)+a^2-1}\otimes X\right)\left(B_{a,b+2c}\right)^{\dagger}\overset{\text{Figure 50}}{=\!=}X\otimes I.\end{aligned} \tag{26}$$

Since $2(d+2a)+c^2-1=2d+4a+c^2-1\equiv_3 2d+a+c^2-1$, we have

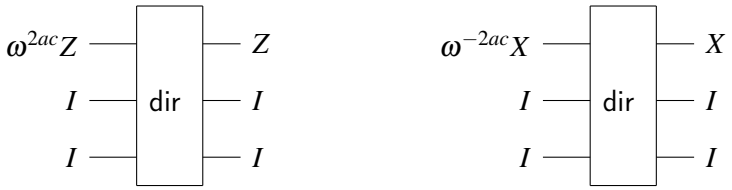
$$\begin{aligned}&\left(B_{c,d+2a}^{-1}\right)^{\dagger}\left(\omega^{-2ac}\omega^{c(d+2a)}X^{2c}Z^{a+2d+c^2-1}\otimes X\right)\left(B_{c,d+2a}^{-1}\right)\\&=\omega^{-2ac}\left(B_{c,d+2a}\right)\left(\omega^{c(d+2a)}X^{2c}Z^{2(d+2a)+c^2-1}\otimes X\right)\left(B_{c,d+2a}\right)^{\dagger}\overset{\text{Figure 50}}{=\!=}\omega^{-2ac}X\otimes I.\end{aligned} \tag{27}$$

(26) and (27) imply that



(28)

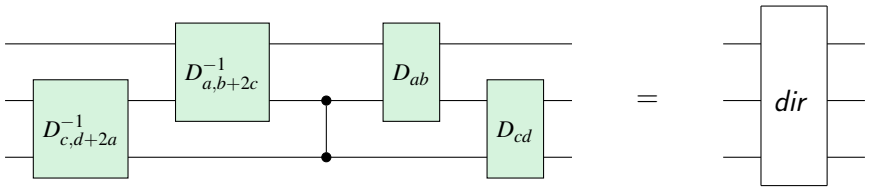
Putting (22), (25), and (28) together, we have



(29)

□

Lemma E.9. *The dirty gate resulting from pushing a CZ through two consecutive D boxes is separable and only has Paulis on the third qutrit:*



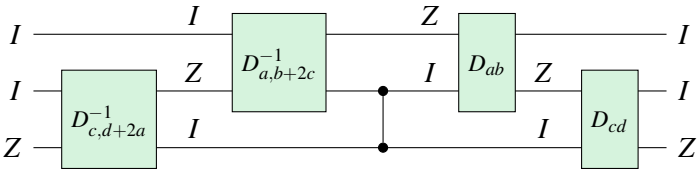
(30)

where $dir = C_1 \otimes C_2$ *and* C_2 *is a Pauli.*

Proof. Based on Lemma E.3, we can find the residual dirty gates after pushing a CZ through two consecutive D boxes by left-appending $\left(D^{-1}_{a,b+2c} \otimes I\right)$ and $\left(I \otimes D^{-1}_{c,d+2a}\right)$ to both sides of (19) as we see in (30).

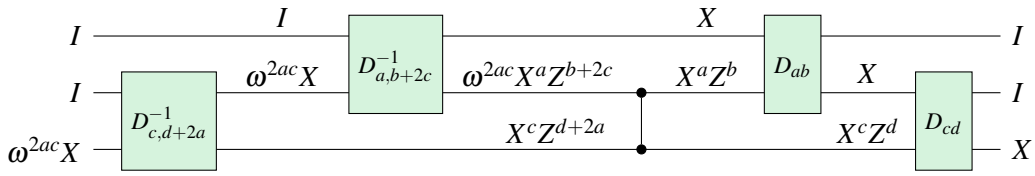
According to the dirty gate restrictions in Figures 52 and 53, our proposed D box construction is suitable for Clifford normalization if there is no single H gate acting on the bottom wire of dir. One way to do this is to show that dir is a separable operator and the bottom wire is only acted on by Pauli X or Z gates. By Proposition E.7 and the single-qutrit Clifford actions in Definition D.1, it suffices to show that dir maps $I_9 \otimes X$ to $I_9 \otimes \omega^{t_1}X$ and $I_9 \otimes Z$ to $I_9 \otimes \omega^{t_2}Z$, for $t_1, t_2 \in \mathbb{Z}_3$.

First, let us find the preimage of $I_9 \otimes Z$ in the lefthand side of (30). According to the third column of Figure 50, we have



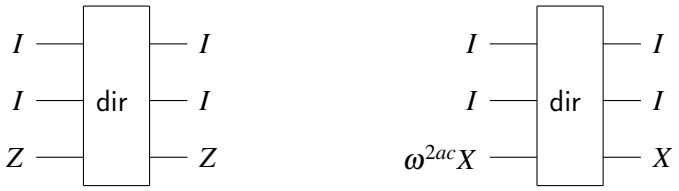
(31)

Next, let us find the preimage of $I_9 \otimes X$ in the lefthand side of (30). According to the second and third columns of Figure 50, a D_{ab} box maps $X \otimes X^a Z^b$ to $I \otimes X$, for all $a, b \in \mathbb{Z}_3$. It follows that



(32)

Putting (30), (31), and (32) together, we have

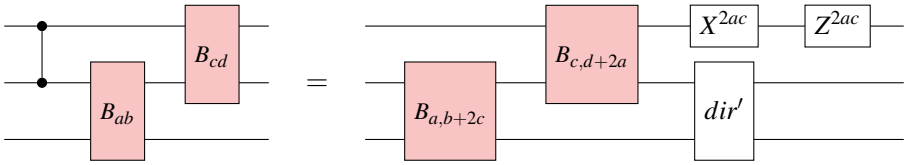


(33)

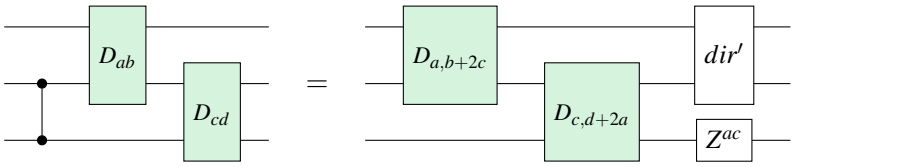
□

Remark E.10. *For the proofs of Lemmas E.8 and E.9 to work, we need the partial automorphism of B and D boxes specified in the second and third columns of Figure 50.*

Lemma E.11. *For $a,b,c,d \in \mathbb{Z}_3$ and $dir' \in \mathcal{C}_2$,*



(34)



(35)

Proof. By Lemma E.3, we can find the updated normal boxes in (18) and (19). When pushing a CZ through two consecutive B boxes, by Lemma E.8 and proposition E.7, dir is a separable operator. According to Appendix D and (29), $\mathsf{dir} = Z^{2ac}X^{2ac} \otimes \mathsf{dir}'$, with some $\mathsf{dir}' \in \mathcal{C}_2$. When pushing a CZ through two consecutive D boxes, by Lemma E.9 and proposition E.7, dir is a separable operator. According to Appendix D and (33), $\mathsf{dir} = \mathsf{dir}' \otimes Z^{ac}$, with some $\mathsf{dir}' \in \mathcal{C}_2$. □

Based on the partial Pauli automorphisms specified in Figure 50, we find concrete implementations of B and D boxes, as shown in Figures 15 and 16. Moreover, the dirty gates after pushing a CZ through two consecutive B and D boxes satisfy the restrictions outlined in Figures 52 to 54.

Summary For the whole procedure (i.e., Clifford normalization) to work, we need to derive all box relations following Procedure E.1, and check that the residual dirty gates satisfy the restrictions given in Figures 52 to 54. This can be done by inspecting every box relation listed in Appendix C. With these specific designs of normal boxes, we complete the task of designing the normal form for an arbitrary qutrit Clifford operator. As a result, the proof of completeness follows from all the box relations.

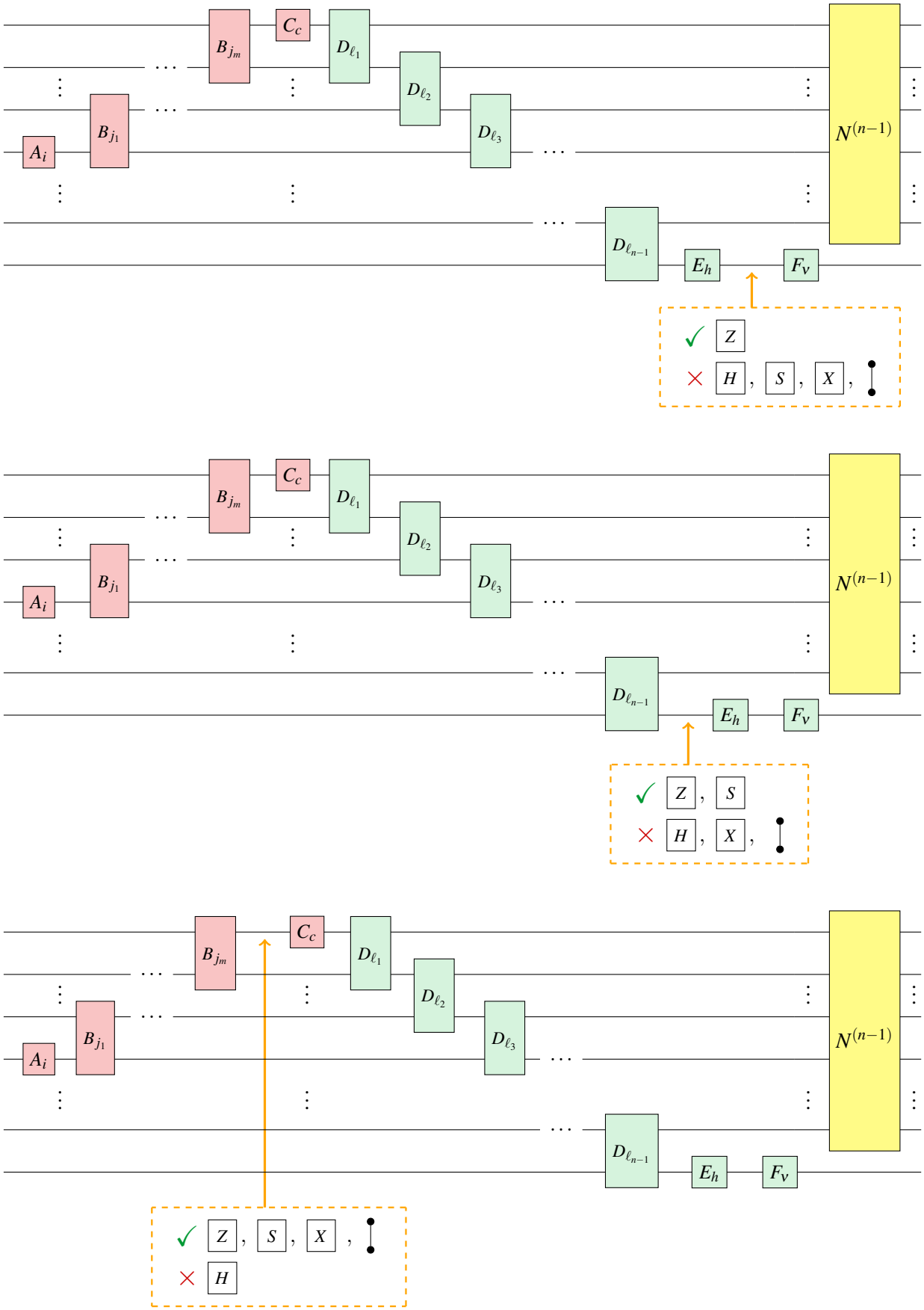


Figure 52: Allowed Clifford operators before C, E, and F boxes.

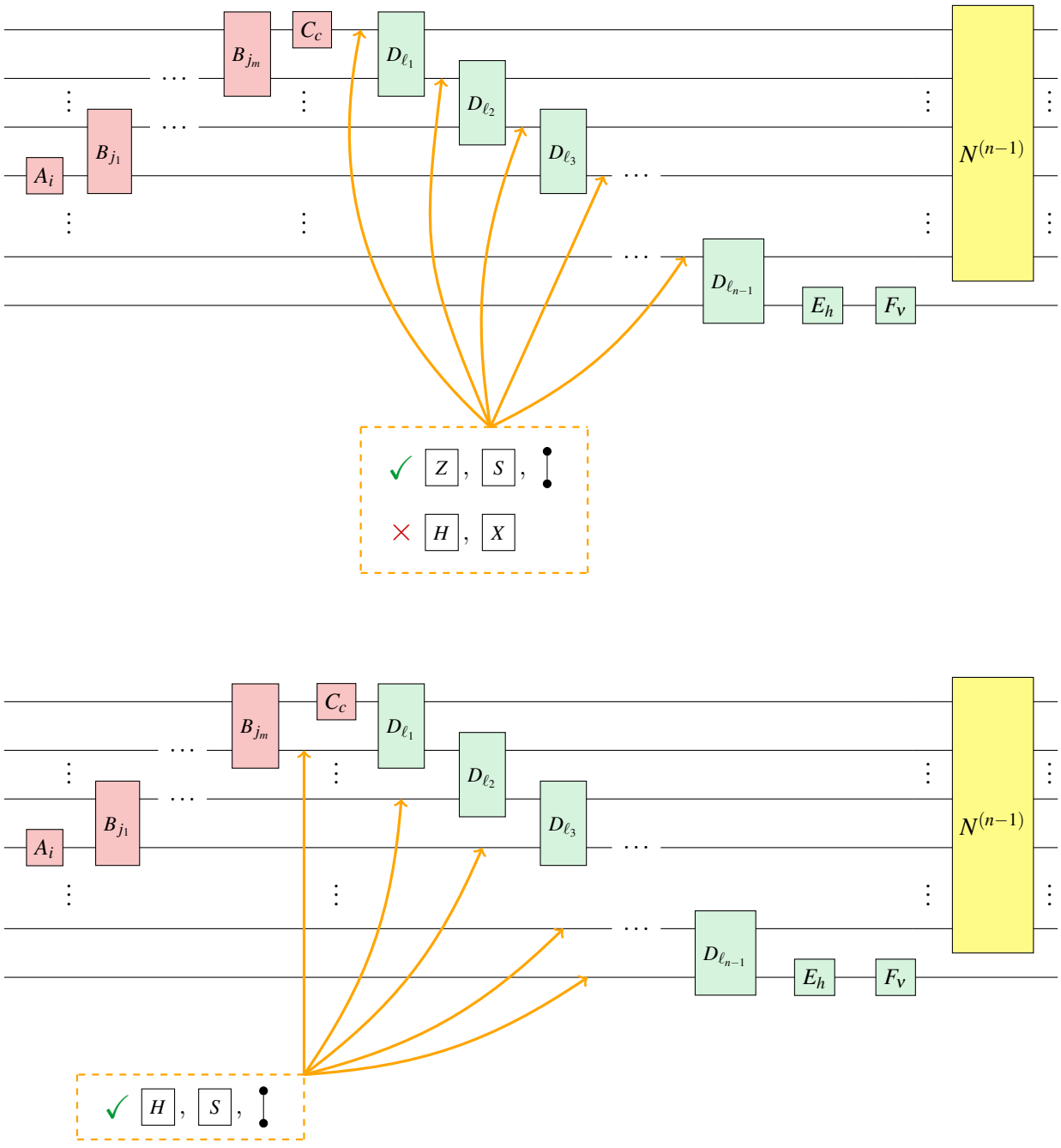


Figure 53: Allowed Clifford operators before D boxes.

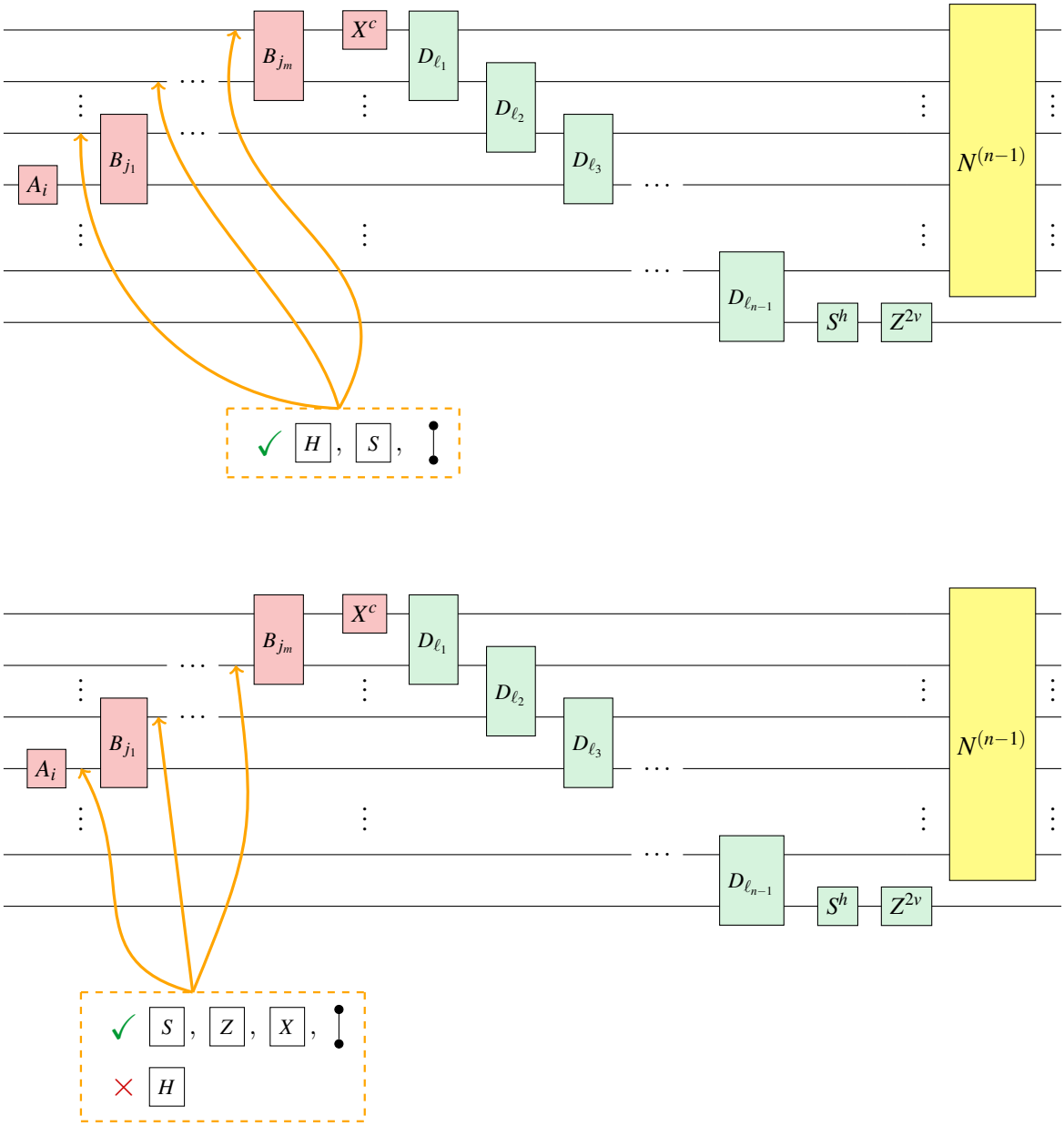


Figure 54: Allowed Clifford operators before B boxes.

On Traces in Categories of Contractions

Extended Abstract

Aaron David Fairbanks Peter Selinger

Dalhousie University, Canada

In memory of Phil Scott, 1947–2023

Traced monoidal categories are used to model processes that can feed their outputs back to their own inputs, abstracting iteration. The category of finite dimensional Hilbert spaces with the direct sum tensor is not traced. But surprisingly, in 2014, Bartha showed that the monoidal subcategory of isometries is traced. The same holds for coisometries, unitary maps, and contractions. This suggests the possibility of feeding outputs of quantum processes back to their own inputs, analogous to iteration. In this paper, we show that Bartha's result is not specifically tied to Hilbert spaces, but works in any dagger additive category with Moore-Penrose pseudoinverses (a natural dagger-categorical generalization of inverses).

1 Introduction

A trace on a symmetric monoidal category $(\mathbf{C}, \oplus)$ is an operation that assigns to each $f\colon A \oplus X \to B \oplus X$ another map $\mathrm{Tr}^X f\colon A \to B$, satisfying some well-known axioms [10, 14, 26]. In string diagrams, traces are represented by looping an output of f back to the corresponding input, as in the following diagram.

In categories of vector spaces, there are two relevant monoidal structures: the "multiplicative" tensor $\otimes$ and the "additive" tensor $\oplus$, also known as biproduct or direct sum. The multiplicative tensor on finite dimensional vector spaces has a well-known trace (induced by the compact closed structure). But in this paper, we are interested in the additive tensor.

A natural way to try to define an additive trace on a category of vector spaces is by the following sum-over-paths formula, which is motivated by the accompanying string diagrams.

$$f = \begin{array}{c} \\ B \\ \oplus \\ X \end{array} \begin{array}{c} \begin{array}{cc} A & \oplus \quad X \end{array} \\ \begin{pmatrix} f_{BA} & f_{BX} \\ f_{XA} & f_{XX} \end{pmatrix} \end{array} \qquad \mathrm{Tr}^X f = f_{BA} + f_{BX} \circ f_{XA} + f_{BX} \circ f_{XX} \circ f_{XA} + f_{BX} \circ (f_{XX})^2 \circ f_{XA} + \cdots$$

The idea is similar to that of matrix multiplication, which can be formulated as a sum over all paths from a given input coordinate to a given output coordinate. However, the sum may not converge (supposing there is even any notion of convergence), and so the sum-over-paths formula does not define a total operation. Indeed, there is no totally defined trace with respect to $\oplus$ on any category of finite (or infinite) dimensional vector spaces [9].

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 79–99, doi:10.4204/EPTCS.426.3

Therefore, it came as a surprise when Bartha showed in [3] that the category of finite dimensional Hilbert spaces and *isometries* has a well-defined additive trace. In particular, not only does Bartha's trace of an isometry always exist, but it is again an isometry. By duality, Bartha's trace also works for coisometries, and therefore also for unitary maps. Moreover, Andrés-Martínez pointed out that Bartha's trace further generalizes to all contractions [1]. These results suggest that there might be some physical interpretation of loops in quantum systems, but we do not know what it is.

In this paper, we show that Bartha's result is not specifically tied to Hilbert spaces, but works in any dagger additive category with suitable additional structure. The specific additional structure that we need to assume is the existence of Moore-Penrose pseudoinverses.

In a nutshell, a pseudoinverse of an arrow $f\colon A \to B$ is an arrow $f^\circ\colon B \to A$ such that both $f \circ f^\circ$ and $f^\circ \circ f$ are self-adjoint and $f \circ f^\circ \circ f = f$ and $f^\circ \circ f \circ f^\circ = f^\circ$. Pseudoinverses are unique when they exist, and they generalize inverses. Moreover, the definition of pseudoinverse is purely algebraic and makes sense in any dagger category [5].

Our main result is the following:

Theorem 1. *Given any dagger additive category with pseudoinverses, there is a totally defined trace on each of the following monoidal subcategories:*

- *the unitaries,*
- *the isometries,*
- *the coisometries, and*
- *the contractions.*

Moreover, in the cases of unitaries and contractions, which are dagger monoidal subcategories, the trace is a dagger trace.

After reviewing some background material in Section 2, we introduce contractions in Section 3 and pseudoinverses in Section 4, and prove some of their required properties. Section 5 is devoted to the proof of the main theorem.

Acknowledgements. We thank JS Lemay and Priyaa Varshinee Srinivasan for helpful discussions.

2 Background

2.1 Dagger categories

We recall some basic definitions and properties of dagger categories to fix the notation for the rest of the paper. For a more detailed treatment, see [24, 8, 11].

Definition 2.1 (Dagger category)**.** A *dagger category* is a category equipped with an identity-on-objects involutive contravariant functor, denoted $(-)^\dagger$. In other words, for $f\colon A \to B$, we have $f^\dagger\colon B \to A$, and we have the following properties:

- $f^{\dagger\dagger} = f$,
- $(1_A)^\dagger = 1_A$, and
- $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$.

For example, the category **Hilb** of Hilbert spaces and bounded linear maps is a dagger category. Its full subcategory **FdHilb** of finite dimensional Hilbert spaces is also a dagger category.

Definition 2.2 (Properties of arrows)**.** An arrow $f\colon A \to B$ in a dagger category is called an *isometry* if $f^\dagger \circ f = 1_A$, a *coisometry* if $f \circ f^\dagger = 1_B$, and *unitary* if it is an isometry and a coisometry. Equivalently, f is unitary if it is invertible and $f^{-1} = f^\dagger$. An arrow $f\colon A \to A$ is *self-adjoint* (or *hermitian*) if $f = f^\dagger$.

In this paper, we use the symbol $\oplus$ to denote the monoidal product, because we are mainly interested in monoidal structures that are induced by biproducts.

Definition 2.3 (Dagger monoidal category)**.** A *dagger monoidal category* is a dagger category that is also monoidal, such that $(-)^\dagger$ is a strict monoidal functor. More explicitly, this means that the monoidal structure isomorphisms (i.e., associators and unitors) are unitary, and for all arrows f and g, we have

$$(f \oplus g)^\dagger = f^\dagger \oplus g^\dagger.$$

In a dagger (monoidal) category, the isometries, the coisometries, and the unitary maps each form a (monoidal) subcategory, i.e., they are closed under compositions (and monoidal products).

Definition 2.4 (Dagger finite biproduct category)**.** A *dagger finite biproduct category* is a dagger category that also has finite biproducts, such that the projection maps $\pi_i\colon A_1 \oplus A_2 \to A_i$ and the inclusion maps $\iota_i\colon A_i \to A_1 \oplus A_2$ satisfy $\pi_i = \iota_i^\dagger$.

The dagger biproducts of course also form a dagger monoidal structure. As usual in any category with finite biproducts, there is a zero object 0, and we can define the addition of arrows $f, g\colon A \to B$ in the usual way by $f + g = A \to A \oplus A \xrightarrow{f \oplus g} B \oplus B \to B$. There are also zero maps $0\colon A \to 0 \to B$. Indeed, every (dagger) finite biproduct category is enriched over commutative monoids.

Definition 2.5 (Negatives, additive category)**.** We say that a (dagger) finite biproduct category *has negatives* if for every $f\colon A \to B$, there exists $-f\colon A \to B$ such that $f + (-f) = 0$. A (dagger) finite biproduct category with negatives is called a (dagger) *additive* category.

Definition 2.6 (Positive map)**.** A map $a\colon A \to A$ in a dagger category is *positive* if there exists $f\colon A \to B$ with $a = f^\dagger \circ f$.

Positive maps in **Hilb** are positive operators in the usual sense. Every positive map is self-adjoint, and the sum of positive maps is positive if we have dagger biproducts. Given two maps $f, g\colon A \to A$, in a dagger finite biproduct category, we say that $f \leq g$ if there exists some positive a such that $g = f + a$.

Lemma 2.7. *Let $f, g\colon A \to A$ and assume $f \leq g$. (a) For all $h\colon A \to B$, we have $h \circ f \circ h^\dagger \leq h \circ g \circ h^\dagger$. (b) For all $f', g'\colon A' \to A'$ with $f' \leq g'$, we have $f \oplus f' \leq g \oplus g'$.*

Proof. Since $g = f + a$ and $g' = f' + a'$ for some positive a and a', we have

$$h \circ g \circ h^\dagger = h \circ f \circ h^\dagger + h \circ a \circ h^\dagger \quad \text{and} \quad g \oplus g' = (f + a) \oplus (f' + a') = (f \oplus f') + (a \oplus a').$$

It is easy to see $h \circ a \circ h^\dagger$ and $a \oplus a'$ are positive, which implies both claims. □

2.2 Matrices

In a finite biproduct category, consider objects $A = A_1 \oplus \cdots \oplus A_m$ and $B = B_1 \oplus \cdots \oplus B_n$. It is well-known that maps $f\colon A \to B$ are in one-to-one correspondence with matrices (f_{ji}), where $f_{ji}\colon A_i \to B_j$. We write

$$f = \begin{array}{c} \\ B_1 \\ \oplus \\ \vdots \\ \oplus \\ B_n \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & \oplus \cdots \oplus & A_m \end{array} \\ \begin{pmatrix} f_{11} & \cdots & f_{1m} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nm} \end{pmatrix} \end{array}.$$

Then composition of morphisms coincides with the usual formula for matrix multiplication, given by

$$(g \circ f)_{ki} = \sum_{j \in J} g_{kj} \circ f_{ji}.$$

In a dagger finite biproduct category, the dagger of a morphism coincides with the adjoint of its matrix:

$$(f^\dagger)_{ij} = (f_{ji})^\dagger.$$

Given a matrix

$$f = \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} : A_1 \oplus A_2 \to B_1 \oplus B_2,$$

we call each $f_{ji} \colon A_i \to B_j$ a *component* of f, we call $\begin{pmatrix} f_{1i} \\ f_{2i} \end{pmatrix} : A_i \to B_1 \oplus B_2$ a *column* of f, and we call $\begin{pmatrix} f_{j1} & f_{j2} \end{pmatrix} : A_1 \oplus A_2 \to B_j$ a *row* of f. We use analogous terminology for larger matrices.

Remark 2.8. In a dagger finite biproduct category, an arrow of the form

$$\begin{array}{c} \\ B \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & \oplus & \cdots & \oplus A_m \end{array} \\ \begin{pmatrix} f_1 & \cdots & f_m \end{pmatrix} \end{array}$$

is an isometry if and only if $f_i^\dagger \circ f_i = 1_{A_i}$ for all i and $f_j^\dagger \circ f_i = 0$ for $i \neq j$.

Remark 2.9. In a dagger additive category, every isometry is a component of a unitary. Indeed, suppose $f \colon A \to B$ is an isometry. Then the following arrow is unitary.

$$\begin{array}{c} \\ A \\ \oplus \\ B \end{array} \begin{array}{c} \begin{array}{ccc} A & \oplus & B \end{array} \\ \begin{pmatrix} 0 & & f^\dagger \\ & & \\ f & & 1_B - f \circ f^\dagger \end{pmatrix} \end{array}$$

2.3 Dagger idempotents

Definition 2.10 (Dagger idempotents)**.** A arrow $p \colon A \to A$ is called a *dagger idempotent* (or *projection*) if $p = p \circ p = p^\dagger$.

Whenever $f \colon B \to A$ is an isometry, then $p = f \circ f^\dagger$ is a dagger idempotent. If p is of this form, we say that p is *dagger split*. When dagger splittings exist, they are unique up to unitary isomorphism. It is well-known that every dagger category can be fully embedded in a dagger category with all dagger splittings, called its *dagger idempotent completion* [25]. Moreover, all structure of interest (e.g., monoidal structure, biproducts, addition, negatives, and, as we will later introduce, pseudoinverses [5]) on a dagger category transports to its dagger idempotent completion.

Definition 2.11 (Complementary idempotents)**.** Two idempotents $p, q \colon A \to A$ in a finite biproduct category are *complementary* if $p + q = 1_A$ and $q \circ p = 0 = p \circ q$.

If the category has negatives, complements always exist, because whenever p is a (dagger) idempotent, so is $1 - p$. It is obvious that the complement is unique in that case. Interestingly, uniqueness even holds without assuming negatives, because if both q_1 and q_2 are complements of p, we have $q_1 = (p + q_2) \circ q_1 = q_2 \circ q_1 = q_2 \circ (p + q_1) = q_2$.

Complementary dagger idempotents are an algebraic abstraction of orthogonal complement subspace projections.

Lemma 2.12 (Direct sum decomposition)**.** *Consider a (dagger) finite biproduct category in which all (dagger) idempotents (dagger) split. Given an object A with complementary idempotents $p,q\colon A\to A$, there exist objects A_1,A_2 with $A=A_1\oplus A_2$ such that*

$$p=\begin{array}{c}\\ A_1\\ \oplus\\ A_2\end{array}\begin{array}{c}\begin{array}{cc}A_1\oplus A_2&\end{array}\\ \begin{pmatrix}1&0\\0&0\end{pmatrix}\end{array}\quad\text{and}\quad q=\begin{array}{c}\\ A_1\\ \oplus\\ A_2\end{array}\begin{array}{c}\begin{array}{cc}A_1\oplus A_2&\end{array}\\ \begin{pmatrix}0&0\\0&1\end{pmatrix}\end{array}.$$

Moreover, the factorization is unique up to (unitary) isomorphisms of the direct sum factors.

Proof idea. Let A_1 be a splitting of p, and let A_2 be a splitting of q. The claimed properties are easy to verify. □

Remark 2.13. In Lemma 2.12 and elsewhere, we write $A=A_1\oplus A_2$ instead of $A\cong A_1\oplus A_2$; this is justified because (dagger) biproducts are defined up to (unitary) isomorphism in the first place.

2.4 Trace

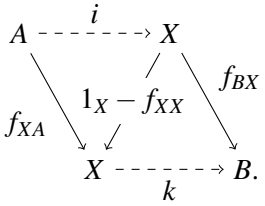
A *trace* on a symmetric monoidal category $\mathbf{C}$ is a family of operations $\mathrm{Tr}^X\colon \mathbf{C}(A\oplus X,B\oplus X)\to\mathbf{C}(A,B)$, subject to a small number of axioms that can be found in [10, 14, 26]. The concept of a *partial trace* is defined similarly, except that Tr^X is a partially defined operation [6]. The axioms are such that a partially traced category in which the trace happens to be totally defined is a (totally) traced category. It was shown in [13, 14] that every partially traced category can be faithfully embedded in a totally traced one, and conversely, every monoidal subcategory of a totally traced category is partially traced.

We will make use of the following construction, which can be found in [14]. It is remarkable because it works in any additive category.

Definition 2.14 (Kernel-image trace)**.** Let $f\colon A\oplus X\to B\oplus X$ be an arrow in an additive category. The *kernel-image trace* $\mathrm{Tr}^X_{\mathrm{ki}}f\colon A\to B$ is defined if there exist arrows $i\colon A\to X$ and $k\colon X\to B$ such that

$$f_{XA}=(1_X-f_{XX})\circ i\qquad\text{and}\qquad k\circ(1_X-f_{XX})=f_{BX},$$

as in the following commutative diagram



In this case, we define

$$\mathrm{Tr}^X_{\mathrm{ki}}f=f_{BA}+k\circ(1_X-f_{XX})\circ i.$$

(Otherwise, the kernel-image trace is undefined.) Note $\mathrm{Tr}^X_{\mathrm{ki}}$ is independent of the choice of each i and k, since

$$f_{BA}+f_{BX}\circ i=\mathrm{Tr}^X_{\mathrm{ki}}f=f_{BA}+k\circ f_{XA}.$$

Proposition 2.15 ([14])**.** *The kernel-image trace is a partial trace.*

Remark 2.16. In a dagger category, a (partial) trace is called a *dagger (partial) trace* if $\mathrm{Tr}(f^\dagger)=(\mathrm{Tr}f)^\dagger$. In a dagger additive category, the kernel-image trace is always a dagger partial trace, because its definition is self-dual.

3 Contractions

3.1 Basic properties

In the category of Hilbert spaces, a *contraction* is a map $f\colon A \to B$ such that for all $v \in A$, $\|f(v)\| \leq \|v\|$. The following definition generalizes this concept to arbitrary dagger additive categories.

Definition 3.1 (Contraction)**.** A *contraction* in a dagger additive category is an arrow $f\colon A \to B$ such that $f^\dagger \circ f \leq 1_A$. In other words, such that there exists an arrow $g\colon A \to B'$ with $f^\dagger \circ f + g^\dagger \circ g = 1_A$. Note that this is the case if and only if the map $\binom{f}{g}\colon A \to B \oplus B'$ is an isometry. A *cocontraction* is defined dually.

In particular, every isometry, coisometry, and unitary map is a contraction. Also, biproduct projections and injections are contractions.

Note that Definition 3.1 could be stated even without assuming negatives, but most of the useful properties of contractions rely on additivity. A point in case is the next proposition, which gives several alternative characterizations of contractions, none of which would be equivalent in the absence of negatives (see counterexamples D.14 and D.15).

Proposition 3.2 (Characterizations of contractions)**.** *Let $f\colon A \to B$ be an arrow in a dagger additive category. The following are equivalent.*

(a) f is a component of a unitary.

(b) f is a contraction.

(c) f is a cocontraction.

(d) f is of the form $e \circ m$ for some isometry $m\colon A \to X$ and coisometry $e\colon X \to B$.

(e) f is a composition of isometries and coisometries.

We delay the proof until we have established some lemmas. The following lemma tells us that contractions, like isometries, form a monoidal subcategory.

Lemma 3.3. *Contractions are closed under composition and monoidal products.*

Proof. For composition, let $f\colon A \to B$ and $g\colon B \to C$ be contractions. Then $f^\dagger \circ f \leq 1_A$ and $g^\dagger \circ g \leq 1_B$. Using Lemma 2.7(a), we get

$$(g \circ f)^\dagger \circ (g \circ f) = f^\dagger \circ g^\dagger \circ g \circ f \leq f^\dagger \circ 1_B \circ f = f^\dagger \circ f \leq 1_A.$$

Therefore, $f \circ g$ is a contraction. For monoidal products, let $f\colon A \to B$ and $g\colon A' \to B'$ be contractions. Using Lemma 2.7(b), we get

$$(f \oplus g)^\dagger \circ (f \oplus g) = (f^\dagger \circ f) \oplus (g^\dagger \circ g) \leq 1_A \oplus 1_{A'} = 1_{A \oplus A'}.$$

Therefore, $f \oplus g$ is a contraction. □

Lemma 3.4 (Contractions as components of unitaries)**.** *In a dagger additive category, contractions are precisely the components of unitaries. In particular, contractions coincide with cocontractions.*

Proof. First, a component of a unitary is a composition of three contractions $u_{jk} = \pi_j \circ u \circ \iota_k$, and is therefore a contraction itself. Conversely, every contraction is a component of an isometry (as remarked in Definition 3.1), which in turn is a component of a unitary by Remark 2.9. Finally, since being a component of a unitary is a self-dual concept, so is being a contraction. □

We can now prove Proposition 3.2.

Proof of Proposition 3.2. The equivalence $(a) \iff (b) \iff (c)$ is Lemma 3.4. For $(b) \implies (d)$, assume $f^\dagger \circ f + g^\dagger \circ g = 1$. Then $f = e \circ m$, where $e = \begin{pmatrix} 1 & 0 \end{pmatrix}$ is a coisometry and $m = \begin{pmatrix} f \\ g \end{pmatrix}$ is an isometry. The implication $(d) \implies (e)$ is trivial, and $(e) \implies (b)$ follows because contractions are closed under composition by Lemma 3.3. □

3.2 Contractions and definiteness

Contractions have even better properties when the underlying dagger category satisfies the following condition.

Definition 3.5 (Definite)**.** A dagger category with a zero object is *definite* if for all arrows f, we have that $f^\dagger \circ f = 0$ implies $f = 0$.

In the familiar context of Hilbert spaces, the columns or rows of a contraction have norm at most 1. An analogue of this principle holds in any definite dagger additive category.

Lemma 3.6 (Maxed-out column)**.** *In a definite dagger additive category, assume $f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ is a contraction. If f_1 is an isometry, then $f_2 = 0$.*

Proof. It suffices to show the result when f is an isometry, because every contraction is a row of an isometry. We have $1_A = f^\dagger \circ f = f_1^\dagger \circ f_1 + f_2^\dagger \circ f_2 = 1_A + f_2^\dagger \circ f_2$. Subtracting 1_A from both sides, we get $0 = f_2^\dagger \circ f_2$. Now by definiteness, $f_2 = 0$. □

Corollary 3.7 (Maxed-out row and column)**.** *In a definite dagger additive category, assume*

$$f = \begin{pmatrix} 1 & f_{12} \\ f_{21} & f_{22} \end{pmatrix}$$

is a contraction. Then $f_{12} = 0$ and $f_{21} = 0$.

Proof. $f_{12} = 0$ follows from Lemma 3.6 and $f_{21} = 0$ follows from its dual. □

The first part of the following is basically Corollary 3.7 in more algebraic language. The second part amounts to the observation that the fixed points of a contraction f are also fixed by $f^\dagger$.

Corollary 3.8 (Fixed points of contraction)**.** *Suppose $f\colon A \to A$ is a contraction and $p\colon A \to A$ is a dagger idempotent in a definite dagger additive category.*

(a) If $p \circ f \circ p = p$, then $f \circ p = p = p \circ f$.

(b) $f \circ p = p$ if and only if $p \circ f = p$.

Proof. Without loss of generality, we can assume all dagger idempotents split, because otherwise we can pass to the dagger idempotent completion. Let $A = A_1 \oplus A_2$ be the decomposition of A obtained by splitting p and its complement as in Lemma 2.12. Write

$$f = \begin{array}{c} \\ A_1 \\ \oplus \\ A_2 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & \oplus \; A_2 \end{array} \\ \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} \end{array}.$$

To prove (a), note that $p \circ f \circ p = p$ means that $f_{11} = 1$, which by 3.7 implies that $f_{12} = 0$ and $f_{21} = 0$, hence $f \circ p = p = p \circ f$. Claim (b) follows from (a). □

4 Pseudoinverses

4.1 Definition of pseudoinverse

Every linear map $f\colon V \to W$ between finite dimensional Hilbert spaces is of the form

$$f \;=\; \begin{array}{c} \\ \mathrm{im}f \\ \oplus \\ (\mathrm{im}f)^{\perp} \end{array} \begin{array}{c} \begin{array}{cc} (\ker f)^{\perp} & \oplus \quad \ker f \end{array} \\ \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix} \end{array},$$

where a is invertible. This section is about dagger additive categories in which an analogous fact holds. Observe that, given the above decomposition of $f\colon V \to W$, we automatically get a map $f^{\circ}\colon W \to V$ in the other direction via

$$f^{\circ} \;=\; \begin{array}{c} \\ (\ker f)^{\perp} \\ \oplus \\ \ker f \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{im}f & \oplus \quad (\mathrm{im}f)^{\perp} \end{array} \\ \begin{pmatrix} a^{-1} & 0 \\ 0 & 0 \end{pmatrix} \end{array}.$$

We note that this "almost inverse" f° of f satisfies the following four properties:

$$f = f \circ f^{\circ} \circ f, \qquad f^{\circ} = f^{\circ} \circ f \circ f^{\circ}, \qquad f^{\circ} \circ f = (f^{\circ} \circ f)^{\dagger}, \qquad f \circ f^{\circ} = (f \circ f^{\circ})^{\dagger}. \tag{1}$$

It so happens that these four laws uniquely determine f° given f.

Definition 4.1 (Pseudoinverse)**.** In a dagger category, a *pseudoinverse* (or *Moore-Penrose pseudoinverse*) of a map $f\colon A \to B$ is an arrow $f^{\circ}\colon B \to A$ such that the equations (1) hold. A *pseudoinverse dagger category* (in [5], *Moore-Penrose dagger category*) is a dagger category in which every arrow has a pseudoinverse.

Before we prove uniqueness, here is a bit of background on pseudoinverses. They were introduced by Moore in [15] and rediscovered by Penrose in [17]. For an overview, see [4] or [2]. Pseudoinverses were studied in abstract dagger categories by Puystjens and Robinson in [18, 19, 20, 21, 22] and recently by Cockett and Lemay in [5].

Example 4.2. In **Hilb**, an arrow $f\colon \mathscr{H} \to \mathscr{H}'$ is pseudoinvertible if and only if the image of f is closed. In **FdHilb**, every arrow is pseudoinvertible.

We note the following equivalent characterization of pseudoinverses; it will simplify the proof of uniqueness in Proposition 4.4 below.

Lemma 4.3 (Second definition of pseudoinverse)**.** *Pseudoinverses f and f° in a dagger category are equivalently characterized by the equations*

$$f = f^{\circ\dagger} \circ f^{\dagger} \circ f, \qquad f = f \circ f^{\dagger} \circ f^{\circ\dagger}, \qquad f^{\circ} = f^{\dagger} \circ f^{\circ\dagger} \circ f^{\circ}, \qquad f^{\circ} = f^{\circ} \circ f^{\circ\dagger} \circ f^{\dagger}. \tag{2}$$

Proof. From (2), we derive

$$f \circ f^{\circ} = f^{\circ\dagger} \circ f^{\dagger} \circ f \circ f^{\circ} = f^{\circ\dagger} \circ f^{\dagger} \quad \text{and} \quad f^{\circ} \circ f = f^{\dagger} \circ f^{\circ\dagger} \circ f^{\circ} \circ f = f^{\dagger} \circ f^{\circ\dagger},$$

i.e., $f \circ f^{\circ} = (f \circ f^{\circ})^{\dagger}$ and $f^{\circ} \circ f = (f^{\circ} \circ f)^{\dagger}$. Hence the two definitions are equivalent as f and f° are permitted to slide past each other, picking up daggers. □

Proposition 4.4 (Uniqueness of pseudoinverse)**.** *If f° and $f^{\bullet}$ are both pseudoinverses of f, then $f^{\circ} = f^{\bullet}$.*

Proof. $f^\circ = f^\dagger \circ f^{\circ\dagger} \circ f^\circ = f^\bullet \circ f \circ f^\dagger \circ f^{\circ\dagger} \circ f^\circ = f^\bullet \circ f \circ f^\circ$. Symmetrically, $f^\bullet = f^\bullet \circ f \circ f^\circ$. □

Note that the notion of pseudoinverse is self-dual and therefore respected by dagger: if $f\colon A \to B$ is pseudoinvertible, then so is $f^\dagger$ with $(f^\dagger)^\circ = (f^\circ)^\dagger$. Also note that if f is pseudoinvertible, then $f \circ f^\circ$ and $f^\circ \circ f$ are dagger idempotents. More specifically, $f \circ f^\circ$ represents projection onto the image of f, and $f^\circ \circ f$ represents projection onto the coimage of f (i.e., the orthogonal complement of the kernel). We hence obtain the following decomposition, which is analogous to what happens in **FdHilb**.

Proposition 4.5 (Generalized singular value decomposition [5])**.** *Let $f\colon A \to B$ be an arrow in a dagger additive category in which all dagger idempotents split. Then f is pseudoinvertible if and only if we can write $A = A_1 \oplus A_2$ and $B = B_1 \oplus B_2$ such that*

$$f = \begin{array}{c} \\ B_1 \\ \oplus \\ B_2 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & \oplus A_2 \end{array} \\ \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix} \end{array} \quad \text{and} \quad f^\circ = \begin{array}{c} \\ A_1 \\ \oplus \\ A_2 \end{array} \begin{array}{c} \begin{array}{cc} B_1 & \oplus B_2 \end{array} \\ \begin{pmatrix} a^{-1} & 0 \\ 0 & 0 \end{pmatrix} \end{array},$$

where $a\colon A_1 \to B_1$ is invertible. Moreover, the factorization of f is unique up to unitary isomorphisms of the direct sum factors.

Proof. Clearly, if f can be written in the stated form, then f is pseudoinvertible with pseudoinverse as stated. For the left-to-right implication, assume f is pseudoinvertible. Consider the dagger idempotents $f^\circ \circ f\colon A \to A$ and $f \circ f^\circ\colon B \to B$. By splitting them and their complements as in Lemma 2.12, we can write $A = A_1 \oplus A_2$ and $B = B_1 \oplus B_2$, where $f^\circ \circ f = \iota_1^A \circ \pi_1^A$ and $f \circ f^\circ = \iota_1^B \circ \pi_1^B$. Let $a = \pi_1^B \circ f \circ \iota_1^A\colon A_1 \to B_1$. Then $f = f \circ f^\circ \circ f \circ f^\circ \circ f = \iota_1^B \circ \pi_1^B \circ f \circ \iota_1^A \circ \pi_1^A = \iota_1^B \circ a \circ \pi_1^A$, hence f is of the claimed form. Moreover, it is easy to verify that $a^{-1} = \pi_1^A \circ f^\circ \circ \iota_1^B$. Uniqueness is as in Lemma 2.12. □

4.2 EP maps

The generalized singular value decomposition of Proposition 4.5 is especially nice if f is a so-called EP-map, which we now define. This definition captures the notion of an endomorphism whose kernel and image are orthogonal complements.

Definition 4.6 (EP maps)**.** An *EP map* (or *range hermitian map*) in a dagger category is a pseudoinvertible endomorphism $f\colon A \to A$ such that $f^\circ \circ f = f \circ f^\circ$.

The term "EP" was introduced by Schwerdtfeger [23], who does not explain what these letters stand for. Given that $f^\circ \circ f$ and $f \circ f^\circ$ are projections that are equal to each other, a useful mnemonic is that EP stands for "equal projections".

Remark 4.7 (Normal operators are EP)**.** If f is pseudoinvertible and $f^\dagger \circ f = f \circ f^\dagger$, then f is EP:

$$f^\circ \circ f = f^\dagger \circ f^{\circ\dagger} = f^\dagger \circ f \circ f^\circ \circ f^{\circ\dagger} = f^\dagger \circ f \circ (f^\dagger \circ f)^\circ = f \circ f^\dagger \circ (f \circ f^\dagger)^\circ = f \circ f^\dagger \circ f^{\circ\dagger} \circ f^\circ = f \circ f^\circ.$$

The following proposition characterizes EP maps in the style of Proposition 4.5.

Proposition 4.8. *Let $f\colon A \to A$ be an arrow in a dagger additive category in which all dagger idempotents split. Then f is EP if and only if we can write $A = A_1 \oplus A_2$ such that*

$$f = \begin{array}{c} \\ A_1 \\ \oplus \\ A_2 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & \oplus A_2 \end{array} \\ \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix} \end{array},$$

where $a\colon A_1 \to A_1$ is invertible.

Proof. Like the proof of Proposition 4.5, but using the fact that the idempotents $f \circ f^\circ$ and $f^\circ \circ f$ are equal and therefore have the same splitting. □

Before we say more about EP maps, we need the following lemma.

Lemma 4.9. *In a dagger category with a zero object, if $f\colon A \to B$ is pseudoinvertible and $f^\dagger \circ f = 0$, then $f = 0$. In particular, every pseudoinverse dagger category with a zero object is definite.*

Proof. Using (2) from Lemma 4.3, we have $f = f^{\circ\dagger} \circ f^\dagger \circ f = 0$. □

We saw in Corollary 3.8 that the fixed points of a contraction f are also fixed by $f^\dagger$. The following lemma is the same fact in different language: $g = 1 - f$ being EP means that $1 - g^\circ \circ g$ (the projection onto the fixed points of f) is equal to $1 - g \circ g^\circ$ (the projection onto the fixed points of $f^\dagger$).

Lemma 4.10 (Contractions and EP maps)**.** *Let $f\colon A \to A$ be a contraction in a pseudoinverse dagger additive category. Then $g = 1_A - f$ is EP.*

Proof. Observe that $(1_A - g) \circ (1_A - g^\circ \circ g) = 1_A - g^\circ \circ g$. By Lemma 4.9, the category is definite, and thus we can apply Corollary 3.8 to obtain $(1_A - g^\circ \circ g) \circ (1_A - g) = 1_A - g^\circ \circ g$. Simplifying, we get $g^\circ \circ g \circ g = g$. Similarly, $g \circ g \circ g^\circ = g$, hence $g^\circ \circ g = g^\circ \circ g \circ g \circ g^\circ = g \circ g^\circ$, as claimed. □

5 Proof of the main result

The purpose of this section is to prove Theorem 1. That is, in a pseudoinverse dagger additive category, the monoidal subcategories of unitaries, isometries, coisometries, and contractions are traced. The proof in the case of isometries proceeds in two steps: In Lemma 5.1, we show that the kernel-image trace of a contraction (and therefore, of an isometry) is always defined. This is the only part of the proof that uses pseudoinverses. In Lemma 5.2, we show that the kernel-image trace of an isometry is again an isometry. These two facts imply that the kernel-image trace is totally defined on the category of isometries. Since it is already known to be a partial trace, these facts are sufficient to prove that the category of isometries is totally traced. The case of contractions is proved similarly, and the other cases are easy consequences.

Lemma 5.1 (Trace is defined for contractions)**.** *In a pseudoinverse dagger additive category, the kernel-image trace is always defined for contractions.*

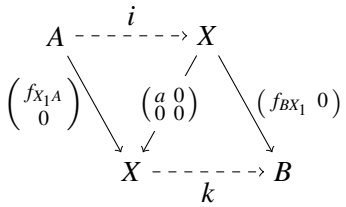
Proof. Let $f\colon A \oplus X \to B \oplus X$ be a contraction. Then f_{XX} is also a contraction, because it can be written as a composition of three contractions $f_{XX} = X \xrightarrow{\iota_X} A \oplus X \xrightarrow{f} B \oplus X \xrightarrow{\pi_X} X$. Then by Lemma 4.10, $1_X - f_{XX}$ is an EP map. For the rest of this proof, we assume that all idempotents split; this is without loss of generality because we can pass to the dagger idempotent completion. Note that the dagger idempotent completion still has pseudoinverses [5]. Because $1_X - f_{XX}$ is an EP map, by Proposition 4.8, we can write

$$1_X - f_{XX} = \begin{array}{c} \\ X_1 \\ \oplus \\ X_2 \end{array} \begin{array}{c} \begin{array}{cc} X_1 \oplus X_2 & \end{array} \\ \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix} \end{array},$$

where we have decomposed X into a sum of two objects $X_1 \oplus X_2$ and $a\colon X_1 \to X_1$ is invertible. Writing f in matrix form, we now have

$$f \quad = \quad \begin{matrix} \\ B \\ \oplus \\ X_1 \\ \oplus \\ X_2 \end{matrix} \begin{matrix} \begin{matrix} A & \oplus & X_1 & \oplus & X_2 \end{matrix} \\ \begin{pmatrix} f_{BA} & f_{BX_1} & f_{BX_2} \\ f_{X_1A} & 1_{X_1} - a & 0 \\ f_{X_2A} & 0 & 1_{X_2} \end{pmatrix} \end{matrix}.$$

Using Corollary 3.7, we get $f_{X_2A} = 0$ and $f_{BX_2} = 0$. To show that the kernel-image trace of f is defined, we must show that there exist i and k to complete the following diagram:



But this can be achieved with $i = \begin{pmatrix} a^{-1} \circ f_{X_1A} \\ 0 \end{pmatrix}$ and $k = \begin{pmatrix} f_{BX_1} \circ a^{-1} & 0 \end{pmatrix}$. □

In the next lemma, we do not assume pseudoinverses, so the kernel-image trace of a given isometry may not exist. However, we show that if it does exist, it is an isometry.

Lemma 5.2 (Trace of isometry)**.** *In a dagger additive category, the kernel-image trace of an isometry, if it exists, is an isometry.*

Proof. Consider an arrow $f\colon A \oplus X \to B \oplus X$ with components

$$f = \begin{matrix} \\ B \\ \oplus \\ X \end{matrix} \begin{matrix} \begin{matrix} A & \oplus & X \end{matrix} \\ \begin{pmatrix} f_{BA} & f_{BX} \\ f_{XA} & f_{XX} \end{pmatrix} \end{matrix}$$

Assume that f is an isometry, so that we have

$$\begin{aligned} f_{BX}^\dagger \circ f_{BX} + f_{XX}^\dagger \circ f_{XX} &= 1_X, \\ f_{BA}^\dagger \circ f_{BA} + f_{XA}^\dagger \circ f_{XA} &= 1_A, \\ f_{BX}^\dagger \circ f_{BA} + f_{XX}^\dagger \circ f_{XA} &= 0. \end{aligned} \tag{3}$$

Also assume that $\mathrm{Tr}^X_{\mathrm{ki}} f$ exists, so in particular there exists $i\colon A \to X$ satisfying

$$f_{XA} = (1_X - f_{XX}) \circ i. \tag{4}$$

To show that $\mathrm{Tr}^X_{\mathrm{ki}} f$ is an isometry, we calculate

$$\begin{aligned}
(\mathrm{Tr}^X_{\mathrm{ki}} f)^\dagger \circ \mathrm{Tr}^X_{\mathrm{ki}} f &= (f^\dagger_{BA} + i^\dagger \circ f^\dagger_{BX}) \circ (f_{BA} + f_{BX} \circ i) \\
&= f^\dagger_{BA} \circ f_{BA} + f^\dagger_{BA} \circ f_{BX} \circ i + i^\dagger \circ f^\dagger_{BX} \circ f_{BA} + i^\dagger \circ f^\dagger_{BX} \circ f_{BX} \circ i \\
\text{(by (3))} \quad &= f^\dagger_{BA} \circ f_{BA} - f^\dagger_{XA} \circ f_{XX} \circ i - i^\dagger \circ f^\dagger_{XX} \circ f_{XA} + i^\dagger \circ (1_X - f^\dagger_{XX} \circ f_{XX}) \circ i \\
\text{(by (4))} \quad &= f^\dagger_{BA} \circ f_{BA} - i^\dagger \circ (1_X - f^\dagger_{XX}) \circ f_{XX} \circ i - i^\dagger \circ f^\dagger_{XX} \circ (1_X - f_{XX}) \circ i + i^\dagger \circ (1_X - f^\dagger_{XX} \circ f_{XX}) \circ i \\
&= f^\dagger_{BA} \circ f_{BA} + i^\dagger \circ (-f_{XX} + f^\dagger_{XX} \circ f_{XX}) \circ i + i^\dagger \circ (-f^\dagger_{XX} + f^\dagger_{XX} \circ f_{XX}) \circ i + i^\dagger \circ (1_X - f^\dagger_{XX} \circ f_{XX}) \circ i \\
&= f^\dagger_{BA} \circ f_{BA} + i^\dagger \circ (1_X - f^\dagger_{XX} - f_{XX} + f^\dagger_{XX} \circ f_{XX}) \circ i \\
&= f^\dagger_{BA} \circ f_{BA} + i^\dagger \circ (1_X - f^\dagger_{XX}) \circ (1_X - f_{XX}) \circ i \\
\text{(by (4))} \quad &= f^\dagger_{BA} \circ f_{BA} + f^\dagger_{XA} \circ f_{XA} \\
\text{(by (3))} \quad &= 1_A . \qquad \square
\end{aligned}$$

Note that the proof only used the i of the kernel-image trace and not the k. We use this fact to immediately obtain the following.

Lemma 5.3 (Trace of contraction)**.** *In a dagger additive category, the kernel-image trace of a contraction, if it exists, is a contraction.*

Proof. Suppose $f \colon A \oplus X \to B \oplus X$ is a contraction. By the definition of contraction, there exists an object B' and an arrow $g \colon A \oplus X \to B'$ such that $f^\dagger \circ f + g^\dagger \circ g = 1_{A \oplus X}$, or in other words, such that

$$h = \begin{array}{c} \\ B' \\ \oplus \\ B \\ \oplus \\ X \end{array} \begin{array}{c} \begin{array}{cc} A & X \end{array} \\ \begin{pmatrix} g_{B'A} & g_{B'X} \\ f_{BA} & f_{BX} \\ f_{XA} & f_{XX} \end{pmatrix} \end{array}$$

is an isometry. Now assume that the kernel-image trace of f exists. While this does not necessarily imply that the kernel-image trace of h exists, we nevertheless get the existence of $i \colon A \to X$ such that $f_{XA} = (1_X - f_{XX}) \circ i$. As seen in the proof of Lemma 5.2, this is sufficient to show that

$$\begin{pmatrix} g_{B'A} \\ f_{BA} \end{pmatrix} + \begin{pmatrix} g_{B'X} \\ f_{BX} \end{pmatrix} \circ i = \begin{pmatrix} g_{B'A} + g_{B'X} \circ i \\ f_{BA} + f_{BX} \circ i \end{pmatrix}$$

is an isometry. Thus, the kernel-image trace of f, which is $f_{BA} + f_{BX} \circ i$, is a contraction, as claimed. $\square$

We are now ready to prove our main theorem.

Proof of Theorem 1. Lemmas 5.1, 5.2, and 5.3 show that the kernel-image trace of the ambient category is total on isometries and contractions. Dually, the same holds for coisometries. Hence the trace is also total on their intersection, the unitaries. Moreover, in the cases of unitaries and contractions, the trace is a dagger trace by Remark 2.16. $\square$

Conclusion and future work

We showed that in every pseudoinverse dagger additive category, each of the subcategories of isometries, coisometries, unitary maps, and contractions forms a totally traced category. This generalizes a result by Bartha in the case of finite dimensional Hilbert spaces. One of the main ingredients of this construction is the notion of pseudoinverse, which was originally studied for matrices by Moore and Penrose, but makes sense in any dagger category. Contractions can also be defined in any dagger category (as compositions of isometries and coisometries), but they only behave as expected if one assumes additive structure and definiteness. The latter follows from the existence of pseudoinverses.

One might ask whether Bartha's trace has a physical interpretation. We do not know the answer, but some potential evidence to the contrary is that the trace on contractions is not a continuous operation, and that it does not exist in infinite dimensional spaces. See Appendix B for more details.

In future work, it would be interesting to investigate whether the assumptions under which contractions are traced could be further reduced. In fact, there are examples of dagger additive categories in which the contractions are totally traced but not all pseudoinverses exist; see Counterexample D.1. On the other hand, it is not sufficient to merely assume, say, the existence of dagger kernels; see Remark B.2.

References

[1] Pablo Andrés-Martínez (2022): *Unbounded Loops in Quantum Programs: Categories and Weak While Loops*. Ph.D. thesis, University of Edinburgh, doi:10.48550/arXiv.2212.05371.

[2] O. M. Baksalary & G. Trenkler (2021): *The Moore-Penrose Inverse: A Hundred Years on a Frontline of Physics Research*. The European Physical Journal H 46, pp. 1–10, doi:10.1140/epjh/s13129-021-00011-y.

[3] Miklos Bartha (2014): *Quantum Turing Automata*. In: *Proceedings 8th International Workshop on Developments in Computational Models*, *Electronic Proceedings in Theoretical Computer Science* 143, pp. 17–31, doi:10.4204/EPTCS.143.2.

[4] A. Ben-Israel (2002): *The Moore of the Moore-Penrose Inverse*. The Electronic Journal of Linear Algebra 9, pp. 150–157, doi:10.13001/1081-3810.1083.

[5] Robin Cockett & Jean-Simon Pacaud Lemay (2023): *Moore-Penrose Dagger Categories*. In: *Proceedings of the 20th International Conference on Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 384, pp. 171–186, doi:10.4204/eptcs.384.10.

[6] Esfandiar Haghverdi & Philip J. Scott (2010): *Towards a Typed Geometry of Interaction*. *Mathematical Structures in Computer Science* 20(3), pp. 1–49, doi:10.1017/S096012951000006X.

[7] Chris Heunen & Martti Karvonen (2019): *Limits in Dagger Categories*. *Theory and Applications of Categories* 24(18), pp. 468–513, doi:10.48550/arXiv.1803.06651.

[8] Chris Heunen & Jamie Vicary (2019): *Categories for Quantum Theory: An Introduction*. Oxford University Press, doi:10.1093/oso/9780198739623.001.0001.

[9] Naohiko Hoshino (2018): *Partial Traces on Additive Categories*. In: *Proceedings of the 34th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIV)*, *Electronic Notes in Theoretical Computer Science* 341, pp. 219–237, doi:10.1016/j.entcs.2018.11.011.

[10] André Joyal, Ross Street & Dominic Verity (1996): *Traced Monoidal Categories*. *Mathematical Proceedings of the Cambridge Philosophical Society* 119, pp. 447–468, doi:10.1017/S0305004100074338.

[11] Martti Karvonen (2019): *The Way of the Dagger*. Ph.D. thesis, University of Edinburgh, doi:10.48550/arXiv.1904.10805.

[12] Saunders Mac Lane (1998): *Categories for the Working Mathematician*, 2nd edition. *Graduate Texts in Mathematics* 5, Springer-Verlag, doi:10.1007/978-1-4757-4721-8.

[13] O. Malherbe (2010): *Categorical Models of Computation: Partially Traced Categories and Presheaf Models of Quantum Computation*. Ph.D. thesis, Department of Mathematics and Statistics, University of Ottawa, doi:10.48550/arXiv.1301.5087.

[14] Octavio Malherbe, Philip J. Scott & Peter Selinger (2012): *Partially Traced Categories*. *Journal of Pure and Applied Algebra* 216(12), pp. 2563–2585, doi:10.1016/j.jpaa.2012.03.026. Also available from arXiv:1107.3608.

[15] E.H. Moore (1920): *On the Reciprocal of the General Algebraic Matrix*. *Bulletin of the American Mathematical Society* 26(9), pp. 394–395, doi:10.1090/S0002-9904-1920-03322-7.

[16] Martin H. Pearl (1968): *Generalized Inverses of Matrices with Entries Taken from an Arbitrary Field*. *Linear Algebra and its Applications* 1(4), pp. 571–587, doi:10.1016/0024-3795(68)90028-1.

[17] Roger Penrose (1955): *A Generalized Inverse for Matrices*. *Mathematical Proceedings of the Cambridge Philosophical Society* 51, pp. 406–413, doi:10.1017/S0305004100030401.

[18] R. Puystjens & D. W. Robinson (1981): *The Moore-Penrose Inverse of a Morphism with Factorization*. *Linear Algebra and its Applications* 40, pp. 129–141, doi:10.1016/0024-3795(81)90145-2.

[19] R. Puystjens & D. W. Robinson (1984): *The Moore-Penrose Inverse of a Morphism in an Additive Category*. *Communications in Algebra* 12(3), pp. 287–299, doi:10.1080/00927878408823004.

[20] R. Puystjens & D. W. Robinson (1985): *EP Morphisms*. *Linear Algebra and its Applications* 64, pp. 157–174, doi:10.1016/0024-3795(85)90273-3.

[21] R. Puystjens & D. W. Robinson (1987): *Generalized Inverses of Morphisms with Kernels*. *Linear Algebra and its Applications* 96, pp. 65–86, doi:10.1016/0024-3795(87)90336-3.

[22] R. Puystjens & D. W. Robinson (1990): *Symmetric Morphisms and the Existence of Moore-Penrose Inverses*. *Linear Algebra and its Applications* 131, pp. 51–69, doi:10.1016/0024-3795(90)90374-L.

[23] Hans Schwerdtfeger (1950): *Introduction to linear algebra and the theory of matrices*. P. Noordhoff.

[24] Peter Selinger (2007): *Dagger Compact Closed Categories and Completely Positive Maps*. In: *Proceedings of the 3rd International Workshop on Quantum Programming Languages, QPL 2005, Chicago*, *Electronic Notes in Theoretical Computer Science* 170, Elsevier, pp. 139–163, doi:10.1016/j.entcs.2006.12.018.

[25] Peter Selinger (2008): *Idempotents in Dagger Categories*. In: *Proceedings of the 4th International Workshop on Quantum Programming Languages, QPL 2006, Oxford*, *Electronic Notes in Theoretical Computer Science* 210, Elsevier, pp. 107–122, doi:10.1016/j.entcs.2008.04.021.

[26] Peter Selinger (2011): *A Survey of Graphical Languages for Monoidal Categories*. In Bob Coecke, editor: *New Structures for Physics*, *Lecture Notes in Physics* 813, Springer, pp. 289–355, doi:10.1007/978-3-642-12821-9_4. Also available from arXiv:0908.3347.

A The pseudotrace is not a trace

In the proof of our main theorem, pseudoinverses play a minor, but crucial role: they are only used to prove that the kernel-image trace is total. In Bartha's original work [3], pseudoinverses play a larger part, because he uses them directly to define the trace on the category of finite dimensional Hilbert spaces and isometries via the following formula:

$$\mathrm{Tr}^X_{\mathrm{ps}} f = f_{BA} + f_{BX} \circ (1_X - f_{XX})^\circ \circ f_{XA}. \tag{5}$$

In fact, the above formula is defined for all linear maps f (not necessarily isometries), and, as we show below, it agrees with the kernel-image trace whenever the latter exists. However, Bartha's operation is not a trace on the category of all linear maps, because it fails to satisfy the trace axioms. We call it the *pseudotrace*.

Definition A.1 (Pseudotrace)**.** In a dagger additive category, the *pseudotrace* of $f\colon A \oplus X \to B \oplus X$ is defined by (5), if the pseudoinverse of $1_X - f_{XX}$ exists, and undefined otherwise. In particular, if the category has pseudoinverses, this is a totally defined operation.

Warning A.2. In general, the pseudotrace is not a (partial or total) trace. In **FdHilb**, let $X = \mathbb{C}^2$ and $A = \mathbb{C}$, and consider $f\colon A \oplus X \to A \oplus X$ and $g\colon X \to X$ defined by

$$f = \left(\begin{array}{c|cc} 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right) \qquad \text{and} \qquad g = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}.$$

Then we find

$$\mathrm{Tr}^X_{\mathrm{ps}}((1_A \oplus g) \circ f) = 0 \qquad \text{and} \qquad \mathrm{Tr}^X_{\mathrm{ps}}(f \circ (1_A \oplus g)) = -1,$$

violating dinaturality (one of the laws of traces) [14].

Lemma A.3 (Pseudotrace and kernel-image trace)**.** *In a dagger additive category, whenever the pseudotrace and kernel-image trace are both defined, they coincide.*

Proof. Let $f\colon A \oplus X \to B \oplus X$ be an arrow with both $\mathrm{Tr}^X_{\mathrm{ki}} f$ and $\mathrm{Tr}^X_{\mathrm{ps}} f$ defined. Taking i and k as in Definition 2.14, we have

$$\begin{aligned} \mathrm{Tr}^X_{\mathrm{ki}} f &= f_{BA} + k \circ (1_X - f_{XX}) \circ i \\ &= f_{BA} + k \circ (1_X - f_{XX}) \circ (1_X - f_{XX})^\circ \circ (1_X - f_{XX}) \circ i \\ &= f_{BA} + f_{BX} \circ (1_X - f_{XX})^\circ \circ f_{XA} \\ &= \mathrm{Tr}^X_{\mathrm{ps}} f. \end{aligned}$$

□

B Non-physicality of the trace?

One may ask whether the trace operation on Hilbert spaces and unitaries (or isometries, or contractions) has a physical interpretation, e.g., whether there is some physical device that can perform this operation when presented with an input unitary in the form of a black box. One potential issue is that the trace is not a continuous operation, i.e., an infinitesimal variation in the input may cause a large variation in the output. Another potential issue is that neither the pseudotrace (Definition A.1) nor the kernel-image trace is total on infinite dimensional Hilbert spaces, even when restricted to contractions, isometries, coisometries, or unitaries. The next two remarks make this precise.

Remark B.1 (Non-continuity of trace)**.** The trace on finite dimensional Hilbert spaces with unitary maps is not a continuous operation. Take for example the θ-parameterized family of rotations

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

The trace along the second row and column is 1 for $\theta = 0$ but is -1 for $0 < \theta < 2\pi$. On the other hand, the trace is continuous on strict contractions, because in that case the pseudoinverse in (5) is an actual inverse, which is a continuous operation.

Remark B.2 (Nonexistence of trace in infinite dimensions)**.** Let $f\colon \ell^2 \to \ell^2$ be the contraction on the Hilbert space of square-summable sequences that multiplies the nth term of every sequence by $\frac{1}{n}$. Consider the unitary map $\ell^2 \oplus \ell^2 \to \ell^2 \oplus \ell^2$ defined by

$$\begin{pmatrix} -(1_{\ell^2} - f) & \sqrt{1_{\ell^2} - (1_{\ell^2} - f)^2} \\ \sqrt{1_{\ell^2} - (1_{\ell^2} - f)^2} & 1_{\ell^2} - f \end{pmatrix}.$$

The pseudotrace along the second row and column does not exist, as f is not pseudoinvertible (i.e., f does not have a closed image; see Example 4.2). Indeed, if there were an induced invertible map from the coimage of f (here the entire space) to the image of f, then its inverse would have to be unbounded. Neither does the kernel-image trace exist, as $\sqrt{1_{\ell^2} - (1_{\ell^2} - f)^2}$ does not factor through f. Indeed, if there were k with $\sqrt{1_{\ell^2} - (1_{\ell^2} - f)^2} = k \circ f$, then k would have to be unbounded.

C More on pseudoinverses

Pseudoinverses arise inevitably in relation to dagger idempotents. Recall that a morphism of idempotents $f\colon p \to q$ is an arrow f such that $q \circ f \circ p = f$. As shown in [5], the pseudoinvertible arrows in any dagger category $\mathbf{C}$ exactly correspond to isomorphisms of dagger idempotents (that is, isomorphisms in the dagger idempotent completion of $\mathbf{C}$):

Proposition C.1 (Pseudoinverses via dagger idempotents [5])**.** *In a dagger category, an arrow $f\colon A \to B$ is pseudoinvertible if and only if there are dagger idempotents $p\colon A \to A$ and $q\colon B \to B$ such that f is an isomorphism of dagger idempotents $f\colon p \to q$. Furthermore, the inverse isomorphism of dagger idempotents $q \to p$ is given by f°, and we have $p = f^\circ \circ f$ and $q = f \circ f^\circ$.*

Proof. To say $f\colon p \to q$ is an isomorphism of dagger idempotents with inverse $g\colon q \to p$ means $f = q \circ f \circ p$ and $g = p \circ g \circ q$ with $g \circ f = p$ and $f \circ g = q$. Since p and q are dagger idempotents, we have $(g \circ f)^\dagger = g \circ f$ and $(f \circ g)^\dagger = f \circ g$. Moreover $f \circ g \circ f = f$ and $g \circ f \circ g = g$, so $g = f^\circ$.

Conversely, assume f is pseudoinvertible and let $p = f^\circ \circ f$ and $q = f \circ f^\circ$. We have that $f\colon p \to q$ and $f^\circ\colon q \to p$ are morphisms of dagger idempotents, because $f = f \circ f^\circ \circ f \circ f^\circ \circ f$ and $f^\circ = f^\circ \circ f \circ f^\circ \circ f \circ f^\circ$. The compositions $f^\circ \circ f$ and $f \circ f^\circ$ are respectively the identities on p and q. □

Unlike inverses, pseudoinverses do not in general compose; see Counterexample D.5. Still, we do have the following.

Corollary C.2 (Composition of pseudoinverses)**.** *In a dagger category, if $f\colon A \to B$ and $g\colon B \to C$ are pseudoinvertible with $f \circ f^\circ = g^\circ \circ g$, then $g \circ f$ is pseudoinvertible with $(g \circ f)^\circ = f^\circ \circ g^\circ$. Moreover, $g \circ f \circ f^\circ \circ g^\circ = g \circ g^\circ$ and $f^\circ \circ g^\circ \circ g \circ f = f^\circ \circ f$.*

Proof. This follows from Proposition C.1, as a composition of isomorphisms of idempotents is an isomorphism of idempotents. □

In a dagger additive category, we obtain four dagger idempotents of interest (written in the matrix form of Proposition 4.5, assuming that the dagger splittings exist):

$$f \circ f^\circ = \begin{array}{c} \\ \mathrm{im} f \\ \oplus \\ (\mathrm{im} f)^\perp \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{im} f & \oplus \; (\mathrm{im} f)^\perp \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad f^\circ \circ f = \begin{array}{c} \\ (\ker f)^\perp \\ \oplus \\ \ker f \end{array} \begin{array}{c} \begin{array}{cc} (\ker f)^\perp & \oplus \; \ker f \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$

$$1 - f \circ f^\circ = \begin{array}{c} \\ \mathrm{im} f \\ \oplus \\ (\mathrm{im} f)^\perp \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{im} f & \oplus \; (\mathrm{im} f)^\perp \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{array} \qquad 1 - f^\circ \circ f = \begin{array}{c} \\ (\ker f)^\perp \\ \oplus \\ \ker f \end{array} \begin{array}{c} \begin{array}{cc} (\ker f)^\perp & \oplus \; \ker f \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

They are, respectively, the projections onto the image, the coimage, the cokernel, and the kernel of f. The following propositions show that these names are justified.

Proposition C.3 (Image via pseudoinverse)**.** *Let $f\colon A \to B$ be a pseudoinvertible arrow in a dagger category such that $f \circ f^\circ$ splits via the mono $m\colon X \to B$. Then m is the image of f (i.e., the universal subobject through which f factors).*

Proof. We have $m = f \circ f^\circ \circ m$, so m factors through every arrow that f factors through. □

Note that f° and $f^\dagger$ have the same image projection, namely $f^\circ \circ f = f^\dagger \circ f^{\dagger\circ}$. This is also the coimage projection of f (and of $f^{\dagger\circ}$). Dually, f° and $f^\dagger$ have the same coimage projection, namely $f \circ f^\circ = f^{\dagger\circ} \circ f^\dagger$, which is also the image projection of f (and of $f^{\dagger\circ}$).

Proposition C.4 (Kernel via pseudoinverse)**.** *Let $f\colon A \to B$ be a pseudoinvertible arrow in a dagger finite biproduct category, with p and $f^\circ \circ f$ complementary dagger idempotents. Then f has a (dagger) kernel (in the standard sense of [7]) if and only if p (dagger) splits. The splitting is given by the inclusion map of the kernel.*

Proof. Observe that for all arrows $m\colon X \to A$, we have $f \circ m = 0$ if and only if $m = p \circ m$. Indeed, if $f \circ m = 0$, then $m = (f^\circ \circ f + p) \circ m = p \circ m$. Conversely, if $m = p \circ m$ then $f \circ m = f \circ f^\circ \circ f \circ p \circ m = 0$. A universal such arrow m is equivalently characterized as a kernel of f or as an equalizer of p and 1_A. But an equalizer of an idempotent and an identity is the same as a mono splitting the idempotent, as desired. Moreover, a dagger kernel is by definition a kernel that is an isometry. It suffices to observe that every splitting of a dagger idempotent by an isometry m is a dagger splitting:

$$e = e \circ m^\dagger \circ m = m^\dagger \circ e^\dagger \circ m^\dagger = m^\dagger.$$

□

Lemma C.5 (Pseudoinvertible monos split)**.** *In a dagger category, every pseudoinvertible mono (dually, epi) is split by its pseudoinverse.*

Proof. Suppose $f\colon A \to B$ is a pseudoinvertible mono. We have $f = f \circ f^\circ \circ f$. Thus by cancellation $f^\circ \circ f = 1_A$. □

Proposition C.6. *Every pseudoinverse dagger additive category in which all dagger idempotents split is an abelian category (in the standard sense of [12]).*

Proof. By Proposition C.4, all kernels exist. Moreover, every mono m is normal as m is a kernel of $1 - m \circ m^\circ$, using both Proposition C.4 and Lemma C.5. Dually, all cokernels exist and every epi is normal. □

Although this paper is about pseudoinverse dagger additive categories, so far we have not given many examples.

Example C.7. Let $\mathbb{F}$ be any dagger subfield of $\mathbb{C}$ (i.e., a subfield closed under conjugation). Then $\mathbf{Mat}(\mathbb{F})$ is a pseudoinverse dagger additive category. Indeed, it was observed in [16] that the pseudoinverse of a matrix f over an arbitrary dagger field exists if and only if $\mathrm{rank}(f) = \mathrm{rank}(f^\dagger \circ f) = \mathrm{rank}(f \circ f^\dagger)$, and the rank of a matrix does not change upon passing to a larger field.

Lemma C.8. *In a pseudoinverse dagger additive category, $\mathbb{Q}$ embeds into the endomorphism ring at each non-zero object.*

Proof. Let $n \in \mathbb{N}_{\geq 1}$, and let $\delta \colon A \to A^n$ be the canonical n-ary diagonal map. By symmetry, we have $\delta^\circ = \begin{pmatrix} d & \cdots & d \end{pmatrix}$ for some $d \colon A \to A$. Since δ is mono, by Lemma C.5, we have $\delta^\circ \circ \delta = 1_A$, hence $n \cdot d = 1_A$. Thus $n \cdot 1_A$ has a multiplicative inverse. As $\mathbb{Q}$ is the universal ring in which every natural number has a multiplicative inverse, and $\mathbb{Q}$ has no proper quotients, the result follows. □

Example C.9 (Free pseudoinverse dagger additive categories)**.** Since pseudoinverse dagger additive categories are essentially algebraic, we can consider freely generated structures. It follows from Lemma C.8 that the free pseudoinverse dagger additive category on an object is equivalent to $\mathbf{Mat}(\mathbb{Q})$. But more exotic examples exist, such as the free pseudoinverse dagger category on an object with an endomorphism; see Counterexample D.2.

D Counterexamples

This section consists of several counterexamples, which preclude various strengthenings of results in the paper. Our main theorem gives a sufficient condition for the monoidal subcategory of contractions in a dagger additive category to be traced: the existence of pseudoinverses. However, this is not a necessary condition.

Counterexample D.1 (Trace without pseudoinverses)**.** The kernel-image trace on the dagger additive category $\mathbf{Mat}(\mathbb{Z})$ of integer-valued matrices is totally defined on contractions. Indeed, since contractions are equivalently submatrices of unitaries, they are the matrices with entries in $\{-1, 0, 1\}$ with at most one nonzero entry per row and per column, and one may check that all kernel-image traces of such matrices exist. However, not all arrows (even those of the form $1 - f$ with f a contraction) are pseudoinvertible, e.g., the matrix $\begin{pmatrix} 2 \end{pmatrix}$.

Given the examples we have seen so far, it is reasonable to ask whether all pseudoinverse dagger additive categories are sub dagger additive categories of matrices over the complex numbers. However, this is not the case.

Counterexample D.2 (Non complex matrix pseudoinverse dagger additive category)**.** The free pseudoinverse dagger additive category on an object $*$ and an arrow $f \colon * \to *$ does not embed into any dagger additive category of matrices over a field. Indeed, given an endomorphism f of a finite dimensional vector space, the image eventually stabilizes with repeated application, i.e. (assuming relevant pseudoinverses exist), $f^n \circ (f^n)^\circ = f^{n+1} \circ (f^{n+1})^\circ$ for some n. But such n can be arbitrarily high, so in the free instance this cannot happen for any particular n.

We saw in Remark 2.9 that in a dagger additive category, every isometry is a component of a unitary. If moreover all dagger idempotents dagger split, then every isometry is a *column* of a unitary, using Lemma 2.12. We note that this stronger statement does not hold without the assumption of dagger splittings.

Counterexample D.3 (Non unitary-column isometry)**.** Consider the full dagger finite biproduct subcategory of **FdHilb** of spaces with dimension not equal to 1. The inclusion of a 2-dimensional subspace into a 3-dimensional space is then not a column of a unitary.

We saw in Corollary 3.7 that for a contraction in a definite dagger additive category, any row or column with a 1 has all other entries 0. This does not hold without assuming definiteness.

Counterexample D.4 (Non maxed-out row)**.** In $\mathbf{Mat}(\mathbb{F}_2)$, the matrix $\begin{pmatrix}1 & 1 & 1\end{pmatrix}$ is a coisometry.

Next, we give some more counterexamples relating to pseudoinverses. Pseudoinverses do not compose: in general we do not have $(g \circ f)^\circ = f^\circ \circ g^\circ$. However, this does hold when the image projection of f and the coimage projection of g coincide, as in Corollary C.2. We observe that it is not sufficient to merely assume that the image projection of f factors through the coimage projection of g.

Counterexample D.5 (Non composition of pseudoinverses)**.** Consider the dagger idempotent $p = \left(\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right)$ and the invertible matrix $a = \left(\begin{smallmatrix}1 & 1\\ 0 & 1\end{smallmatrix}\right)$ in **FdHilb**. We have $a \circ p = p$, and thus $(a \circ p)^\circ = p^\circ = p$, whereas $p^\circ \circ a^\circ = p \circ a^{-1} = \left(\begin{smallmatrix}1 & -1\\ 0 & 0\end{smallmatrix}\right)$.

On the other hand, the next example shows that it is still possible to have $(g \circ f)^\circ = f^\circ \circ g^\circ$ in cases where the image projection of f and coimage projection of g are incompatible (do not even commute); in particular, the sufficient condition given in Corollary C.2 is not necessary. (See [5] for a necessary and sufficient condition for pseudoinverses to compose.)

Counterexample D.6 (Composition of pseudoinverses)**.** In the dagger category of finite sets and relations (i.e., boolean-valued matrices), the pseudoinverse of the idempotent $p = \left(\begin{smallmatrix}1 & 0\\ 1 & 0\end{smallmatrix}\right)$ is the idempotent $p^\circ = \left(\begin{smallmatrix}1 & 1\\ 0 & 0\end{smallmatrix}\right)$. Hence $(pp)^\circ = p^\circ \circ p^\circ$. However, the image projection $p \circ p^\circ = \left(\begin{smallmatrix}1 & 1\\ 1 & 1\end{smallmatrix}\right)$ does not commute with the coimage projection $p^\circ \circ p = \left(\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right)$.

We saw in Lemma C.5 that every pseudoinvertible mono is a split mono. However, the converse does not hold in general.

Counterexample D.7 (Non pseudoinvertible split mono)**.** In $\mathbf{Mat}(\mathbb{Z})$, $m = \left(\begin{smallmatrix}1\\ 1\end{smallmatrix}\right)$ is a split mono. If its pseudoinverse existed, it would also be the pseudoinverse in $\mathbf{Mat}(\mathbb{Q})$. However, the pseudoinverse in $\mathbf{Mat}(\mathbb{Q})$ is $\left(\begin{smallmatrix}\frac{1}{2} & \frac{1}{2}\end{smallmatrix}\right)$, which is not in $\mathbf{Mat}(\mathbb{Z})$.

The following counterexamples show that when we do not assume the existence of all pseudoinverses, the pseudotrace (Definition A.1) may be defined in cases where the kernel-image trace is undefined or vice versa (even restricting to unitaries).

Counterexample D.8 (Non pseudotrace kernel-image trace)**.** In $\mathbf{Mat}(\mathbb{Z})$, the unitary $\left(\begin{smallmatrix}1 & 0\\ 0 & -1\end{smallmatrix}\right)$ does not have a pseudotrace along the second row and column, as $1-(-1)=2$ is not pseudoinvertible. It does have a kernel-image trace equal to $1+0(2)0=1$.

Counterexample D.9 (Non kernel-image trace pseudotrace)**.** In $\mathbf{Mat}(\mathbb{Z}[x]/\langle x^2\rangle)$, the unitary $\left(\begin{smallmatrix}-1 & x\\ x & 1\end{smallmatrix}\right)$ has pseudotrace along the second row and column, as $1-1=0$ is pseudoinvertible. It does not have a kernel-image trace, as x is not of the form $0 \circ i$ for any i.

Finally, we give several counterexamples that show certain results about dagger additive categories do not hold in the absence of negatives. It follows from Proposition C.4 that any isometry m in a dagger additive category is the dagger kernel of $1 - m \circ m^\dagger$. However, isometries need not be kernels if we do not assume the existence of negatives.

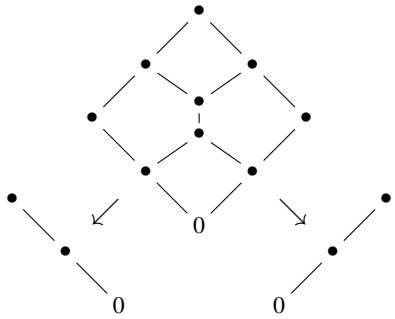
Counterexample D.10 (Non kernel isometry)**.** In the dagger finite biproduct category of finite sets and relations (i.e., boolean-valued matrices), the isometry $\binom{1}{1}$ is not a kernel.

We saw in Lemma 2.12 that complementary split idempotents are tantamount to direct sum decompositions. In an additive category, idempotents p and q are complementary if and only if $p+q=1$, which does not hold in the absence of negatives.

Counterexample D.11 (Non direct sum idempotent sum)**.** Consider the (dagger) finite biproduct category of finite sets and relations. Letting $p=q=1_{\{*\}}$, we have $p+q=1_{\{*\}}$, but $pq\neq 0$. Thus, p and q are not complementary. Also, both p and q are split via $\{*\}$, but $\{*\}$ is not isomorphic to $\{*\}\oplus\{*\}$.

Complementary split idempotents are split by (co)kernels of one another, by an argument similar to the proof of Proposition C.4. Thus each idempotent can be recovered from the other as the (necessarily unique) idempotent split by its kernel and cokernel. Hence, in the absence of negatives, it is reasonable to ask whether idempotents that are split by (co)kernels of one another are necessarily complementary. However, this is not the case.

Counterexample D.12 (Non complementary mutual (co)kernels)**.** In the finite biproduct category of bounded semilattices (equivalently, modules over the booleans), consider the following semilattice, with evident "projection-like" idempotents onto the shown sublattices:

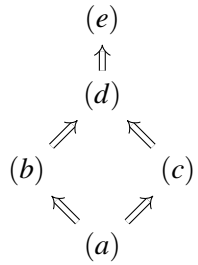


These idempotents are split by (co)kernels of one another, but they are not complementary.

The kernel-image trace is a partial trace on any additive category. It is reasonable to ask whether the same formula works in an arbitrary finite biproduct category, simply leaving the trace undefined where the relevant subtraction is not defined. However, this does not give a partial trace in general.

Counterexample D.13 (Kernel-image non trace)**.** Consider $\mathbf{Mat}(\mathbb{N}[x,y]/\langle xy\rangle)$. The matrix $\binom{xy}{}=0$ has a negative, so the kernel-image trace formula (tracing out the entire input and output) $0+0(1-0)0$ is defined. On the other hand, the matrix $\binom{yx}{}$ does not have a negative, so the kernel-image trace formula is undefined. This violates the dinaturality law for partial traces [14].

In Proposition 3.2 we saw five equivalent characterizations of contractions in a dagger additive category. However, these conditions are not equivalent in a mere dagger finite biproduct category (i.e., without assuming negatives). In fact, they are all distinct, with implications between them as follows:



To distinguish them, it suffices to see that (b) is distinct from (c) and that (d) is distinct from (e); it is then clear that the self-dual definitions are distinct from the non-self-dual definitions. To say (c) is distinct from (d) means that contractions are not the same as cocontractions.

Counterexample D.14 (Non cocontraction contraction)**.** Consider the dagger finite biproduct category of finite sets and relations (i.e., boolean-valued matrices). The isometries are the matrices featuring at least one 1 per column and at most one 1 per row. The matrix $\binom{1}{1}$ is an isometry, thus a contraction, but not a component of a coisometry, thus not a cocontraction.

To say (d) is distinct from (e) means that not every coisometry followed by an isometry is equal to an isometry followed by a coisometry.

Counterexample D.15 (Non isometry then coisometry coisometry then isometry)**.** Consider the free dagger rig on an isometry x, i.e., $\mathbb{N}[x,x^\dagger]\langle x^\dagger x = 1\rangle$. Its elements have explicit normal forms, as finite expressions $\sum_{i,j\geq 0} n_{i,j} x^j (x^\dagger)^i$. In the corresponding dagger finite biproduct category of matrices, the isometries are the matrices having entries in $\{0,1,x,x^2,\ldots\}$ with one nonzero entry per column and at most one nonzero entry per row. Clearly the matrix $\begin{pmatrix} xx^\dagger \end{pmatrix}$ cannot be expressed as an isometry followed by a coisometry.

Inserting Planar-Measured Qubits into MBQC Patterns while Preserving Flow

Miriam Backens
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Thomas Perez
Inria, Palaiseau, France
LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
CPHT, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

In the one-way model of measurement-based quantum computation (MBQC), computation proceeds via single-qubit measurements on a resource state. Flow conditions ensure that the overall computation is deterministic in a suitable sense, and are required for efficient translation into quantum circuits. Procedures that rewrite MBQC patterns – e.g. for optimisation, or adapting to hardware constraints – thus need to preserve the existence of flow. Most previous work has focused on rewrites that reduce the number of qubits in the computation, or that introduce new Pauli-measured qubits.

Here, we consider the insertion of planar-measured qubits into MBQC patterns, i.e. arbitrary measurements in a plane of the Bloch sphere spanned by a pair of Pauli operators; such measurements are necessary for universal MBQC. We extend the definition of causal flow, previously restricted to XY-measurements only, to also permit YZ-measurements and derive the conditions under which a YZ-insertion preserves causal flow. Then we derive conditions for YZ-insertion into patterns with gflow or Pauli flow, in which case the argument straightforwardly extends to XZ-insertions as well. We also show that the 'vertex splitting' or 'neighbour unfusion' rule previously used in the literature can be derived from YZ-insertion and pivoting. This work contributes to understanding the broad properties of flow-preserving rewriting in MBQC and in the ZX-calculus more broadly, and it will enable more efficient optimisation, obfuscation, or routing.

1 Introduction

The one-way model of measurement-based quantum computing (MBQC) is a universal model of quantum computation driven by single-qubit measurements on a graph state [30]. Individual measurements are probabilistic, yet computations can be deterministic overall by modifying later measurements depending on the outcomes of earlier ones. MBQC is a promising model for physically building quantum computers (e.g. based on photonics [33, 36, 13]) as well as for secure delegated quantum computation [4, 20]. It also has theoretical applications in quantum secret-sharing protocols [21] and verifying quantum computations [17, 20]. The higher flexibility of MBQC compared to quantum circuits means that it is easier to optimise computations in the one-way model, e.g. to trade depth against number of ancillas [5], reduce the number of non-stabiliser operations [14], or the number of entangling operations [32].

Many of these applications involve translating a computation given as a quantum circuit into MBQC and then back to a circuit [14, 2, 23, 24] using the ZX-calculus, a diagrammatic language that can represent both quantum circuits over the CNOT and single-qubit gate set, and MBQC [7]. The ZX-calculus also comes with multiple complete equational theories, that allow purely graphical reasoning for translation between the two models as well as for optimisation or other applications [34]. Yet the existing axioms do not always preserve the 'MBQC-form', nor the property of being robustly deterministic.

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 100–126, doi:10.4204/EPTCS.426.4

The widely-used notion of *robust determinism* is captured by a family of flow properties that depend on the graph state underlying the MBQC, as well as the types of measurements, which can be 'Pauli' or 'planar' (see Section 2) [6, 27]. This information is also captured in the ZX-diagram describing a one-way computation, so it is meaningful to consider 'flow-preserving rewrite rules' for MBQC-form ZX-diagrams. This research is of broader importance, as there exist polynomial-time algorithms for translating ZX-diagrams with flow into quantum circuits [14, 2, 31] for physical implementation, whereas translation to circuits is #P-hard in general [12]. A complete set of flow-preserving rewrite rules is known for the Clifford fragment: one-way computations containing only Pauli measurements; this includes the insertion of Pauli-Z measured qubits [23]. Other flow-preserving rewrite rules either reduce or leave invariant the number of qubits [14, 2, 18]. Additionally, a version of 'vertex splitting' (where one qubit is replaced by a chain of three qubits instead) is known to preserve Pauli flow [24].

Here, we look more broadly at the idea of inserting new planar-measured qubits into one-way computations. We first straightforwardly extend the definition of causal flow (the most restricted type of flow, which forces MBQC to be most 'circuit-like') to allow measurements in the YZ-plane as well as the standard XY-plane[1], and then derive the conditions under which YZ-measurements can be added to a computation with causal flow. Next, we derive similar conditions for the insertion of YZ-measured qubits into MBQC with the more general gflow and Pauli flow: the algebraic formulation of focused Pauli flow [28] allows both cases to be treated simultaneously. A simple parity change allows those conditions to be adapted to XZ-measurements instead. Finally, we combine YZ-insertions with the flow-preserving pivot operation [14, 2, 31] (which can change measurement labels) to generalise the vertex splitting rule: this now effectively allows the insertion of pairs of XY-measured qubits under certain conditions. Such insertions of planar-measured vertices instead of Pauli-measured ones are useful if one wishes to stay within the setting of gflow (instead of Pauli flow), which is sometimes desirable. They also have applications in generating variational ansätze in a measurement- or ZX-based setting such as [16, 15].

In the following, we give the preliminary definitions and existing results in Section 2. YZ-insertion for MBQC with causal flow is analysed in Section 3, and for Pauli flow in Section 4. Finally, insertion of measurements in other planes is considered in Section 5, followed by the conclusions in Section 6.

2 Preliminaries

We begin by formalising the graph-theoretic notions and related results that will be used later.

Definition 2.1. A *labelled open graph* is a tuple (G,I,O,λ) consisting of a simple graph $G=(V,E)$, subsets $I,O\subseteq V$ called the *input* and *output vertices*, and a function $\lambda : \bar{O}\to\{X,Y,Z,XY,XZ,YZ\}$ called the *measurement labelling*.

Given a labelled open graph, let $\mathscr{X} := \{v\in V\setminus(I\cup O) \mid \lambda(v)\in\{XY,X,Y\}\}$ be the set of 'X-like' internal vertices and $\mathscr{Z} := \{v\in V\setminus(I\cup O) \mid \lambda(v)\in\{XZ,YZ,Z\}\}$ the set of 'Z-like' internal vertices. Define $\mathscr{L} := \{v\in V\setminus(I\cup O) \mid \lambda(v)\in\{XY,XZ,YZ\}\}$ to be the set of planar measured internal vertices.

Throughout this paper, we denote by Δ the symmetric difference of two sets, which consists of those elements contained in exactly one of the original sets: i.e. $\mathscr{A}\,\Delta\,\mathscr{D} = (\mathscr{A}\cup\mathscr{D})\setminus(\mathscr{A}\cap\mathscr{D})$. The operation is associative so it straightforwardly extends to a family of sets: the symmetric difference $\Delta_{1\leq k\leq n}\mathscr{A}_k$ contains exactly those elements that appear in an odd number of the constituent sets $\mathscr{A}_k$.

Definition 2.2. Let $G=(V,E)$ have vertices V and edges E. For any vertex $u\in V$, we denote by $N_G(u) := \{v\in V \mid \{u,v\}\in E\}$ the *neighbourhood of* u and by $N_G[u] := N_G(u)\,\Delta\,\{u\}$ the *closed neighbourhood*.

[1] This extension was also suggested independently by Calum Holker [18].

Similarly, for any subset $\mathscr{A} \subseteq V$, we denote by $\mathsf{Odd}_G(\mathscr{A}) := \bigtriangleup_{v\in\mathscr{A}} N_G(v)$ the *odd neighbourhood of* $\mathscr{A}$ and by $\mathsf{Odd}_G[\mathscr{A}] := \bigtriangleup_{v\in\mathscr{A}} N_G[v] = \mathsf{Odd}_G(\mathscr{A}) \bigtriangleup \mathscr{A}$ the *closed odd neighbourhood of* $\mathscr{A}$. The subscript designating the graph may be left out if no confusion is likely to result.

Definition 2.3. Let $u \in V$, then the graph $G\star u$ resulting from a *local complementation about* u is $G\star u := (V, E \bigtriangleup \{\{v,w\} \mid v,w \in N_G(u), v \neq w\})$. Let $u \in V$ and $v \in N_G(u)$, then the graph $G \wedge uv$ resulting from a *pivot about the edge* $\{u,v\}$ is $G \wedge uv := G\star u\star v\star u$, i.e. it arises by a sequence of local complementations.

Labelling the pivot by the edge $\{u,v\}$ is well-defined as $G\star u\star v\star u = G\star v\star u\star v$ [3].

2.1 Flow properties

Having established the relevant graph-theoretic concepts, we are now ready to define flow properties.

Definition 2.4 (Causal flow [10, Definition 2])**.** A *causal flow* for a labelled open graph $\Gamma = (G,I,O,\lambda)$ is a pair $(c,\prec)$ where $c : \overline{O} \to \overline{I}$ and $\prec$ is a strict partial order, such that for all $u \in V$:

- $c(u) \in N(u)$
- $u \prec c(u)$
- $\forall v \in N(c(u)), u \neq v \Rightarrow u \prec v$.

The presence of causal flow implies that a one-way computation can be implemented robustly deterministically, but it is not necessary [6]. For labelled open graphs (i.e. computations where no partial order is specified *a priori*), the following property is both necessary and sufficient [6, 27].

Definition 2.5 (Pauli flow [6, Definition 5])**.** A *Pauli flow* for a labelled open graph (G,I,O,λ) is a pair $(c,\prec)$, where $c : \bar{O} \to 2^I$ and $\prec$ is a strict partial order on $\bar{O}$, such that for all $u \in \bar{O}$:

(P1) $\forall v \in c(u).u \neq v \wedge \lambda(v) \notin \{X,Y\} \Rightarrow u \prec v$
(P2) $\forall v \in \mathsf{Odd}(c(u)).u \neq v \wedge \lambda(v) \notin \{Y,Z\} \Rightarrow u \prec v$
(P3) $\forall v \in \bar{O}.\neg(u \prec v) \wedge u \neq v \wedge \lambda(v) = Y \Rightarrow v \notin \mathsf{Odd}[c(u)]$
(P4) $\lambda(u) = XY \Rightarrow u \notin c(u) \wedge u \in \mathsf{Odd}(c(u))$
(P5) $\lambda(u) = XZ \Rightarrow u \in c(u) \wedge u \in \mathsf{Odd}(c(u))$
(P6) $\lambda(u) = YZ \Rightarrow u \in c(u) \wedge u \notin \mathsf{Odd}(c(u))$
(P7) $\lambda(u) = X \Rightarrow u \in \mathsf{Odd}(c(u))$
(P8) $\lambda(u) = Z \Rightarrow u \in c(u)$
(P9) $\lambda(u) = Y \Rightarrow u \in \mathsf{Odd}[c(u)]$.

The sets $c(v)$ for $v \in \bar{O}$ are called the *correction sets* and c is called the *correction function*.

We will sometimes use the term *correction function* to refer to any function $c : \bar{O} \to 2^{\bar{I}}$ on some given labelled open graph (G,I,O,λ) which satisfies properties (P4)–(P9).

A Pauli flow on a labelled open graph in which all measurements are planar is called a gflow[2] and has been widely studied in its own right. Both gflow and Pauli flow are not generally unique; thus a more restricted version satisfying the following 'focusing conditions' is often useful to work with. Focusing was defined first for XY-only gflow [25, Definition 3.1] before being extended [2, Proposition 3.14].

Definition 2.6 ([31, Definition 4.3])**.** Given (G,I,O,λ), a set $\mathscr{A} \subseteq \bar{I}$ is *focused over* $S \subseteq \bar{O}$ if:

(F1) $\forall w \in S \cap \mathscr{A}.\lambda(w) \in \{XY,X,Y\}$,
(F2) $\forall w \in S \cap \mathsf{Odd}(\mathscr{A}).\lambda(w) \in \{XZ,YZ,Y,Z\}$,
(F3) $\forall w \in S.\lambda(w) = Y \Rightarrow w \notin \mathsf{Odd}[\mathscr{A}]$.

A *focused set* $\mathscr{A}$ for Γ is focused over $\bar{O}$. A correction function c is *focused* if $c(v)$ is focused over $\bar{O} \setminus \{v\}$ for all $v \in \bar{O}$. A Pauli flow is called *focused* if its correction function is focused.

Lemma 2.7 ([31, Lemma 4.6])**.** For any labelled open graph, if a Pauli flow exists, then there also exists a focused Pauli flow.

[2] In the literature, the term 'gflow' is sometimes used to refer specifically to a Pauli flow in which all measurement labels are XY, in which case the property allowing XZ and YZ measurements may be called 'extended gflow'.

2.2 Algebraic Pauli flow

A major advantage of focused Pauli flow is that it is amenable to an algebraic characterisation, replacing conditions (P1)–(P9) and (F1)–(F3) by two conditions on certain matrices derived from the labelled open graph. First, we define an 'extended adjacency matrix' to be the adjacency matrix of the graph that results from some labelled open graph (G,I,O,λ) by adding self-loops to all vertices measured Y or XZ. This convention will make the following definitions simpler.

Definition 2.8 ([28, Definition 3.16])**.** Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph. Then the *extended adjacency matrix* A of Γ satisfies $A_{v,w}=1$ if and only if either $\{v,w\}\in E$ or $v=w\wedge\lambda(v)\in\{Y,XZ\}$.

Definition 2.9 ([28, Definition 3.4])**.** Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph with extended adjacency matrix A. The *flow-demand matrix* M_Γ is the $(n-n_O)\times(n-n_I)$ matrix with rows corresponding to non-outputs and columns corresponding to non-inputs, whose v-labelled row satisfies the following properties: if $\lambda(v)\in\{X,Y,XY\}$, then $M_{v,w}=A_{v,w}$ for all $w\in\bar{I}$, and if $\lambda(v)\in\{Z,YZ,XZ\}$, then $M_{v,v}=1$ and $M_{v,w}=0$ for all $w\in\bar{I}\setminus\{v\}$.

Definition 2.10 ([28, Definition 3.5])**.** Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph with extended adjacency matrix A. The *order-demand matrix* N_Γ is the $(n-n_O)\times(n-n_I)$ matrix with rows corresponding to non-outputs and columns corresponding to non-inputs, whose v-labelled row satisfies the following properties: if $\lambda(v)\in\{X,Y,Z\}$, then $N_{v,w}=0$ for all $w\in\bar{I}$; if $\lambda(v)\in\{XZ,YZ\}$, then $N_{v,w}=A_{v,w}$ for all $w\in\bar{I}$; and if $\lambda(v)=XY$, then $N_{v,v}=1$ (provided that the v column exists) and $N_{v,w}=0$ for all $w\in\bar{I}\setminus\{v\}$.

If we represent a set of vertices $\mathscr{A}\subseteq\bar{I}$ as a column vector, which contains a 1 in the row labelled v if and only if $v\in\mathscr{A}$, and multiply this vector by the flow-demand matrix, then the result encodes specific relationships between each non-output vertex u and the set $\mathscr{A}$ that depend on the measurement label $\lambda(u)$ of the vertex under consideration.

Lemma 2.11 ([28, Lemma 3.9])**.** Let (G,I,O,λ) be a labelled open graph. Let $\mathscr{A}\subseteq\bar{I}$ and let $\mathbf{a}$ be the indicator vector for this set. Let $u\in\bar{O}$. Then the product of the u-th row of M with the indicator vector of $\mathscr{A}$ satisfies $M_{u,*}\mathbf{a}=(M\mathbf{a})_u=1$ if and only if: $\lambda(u)\in\{X,XY\}$ and $u\in\mathsf{Odd}(\mathscr{A})$, or $\lambda(u)\in\{XZ,YZ,Z\}$ and $u\in\mathscr{A}$, or $\lambda(u)=Y$ and $u\in\mathsf{Odd}[\mathscr{A}]$.

In addition to the flow-demand and order-demand matrix, we now also encode the correction function as a matrix by assembling the column indicator vectors for each correction set. That definition then enables the statement of the algebraic formulation of Pauli flow theorem.

Definition 2.12 ([28, Definition 3.6])**.** Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph and let c be a correction function on Γ. The *correction matrix* C encoding the function c is the $(n-n_I)\times(n-n_O)$ matrix where $C_{u,v}=1$ if and only if $u\in c(v)$. In other words, the v-labelled column of C encodes the set $c(v)$.

Theorem 2.13 (Algebraic formulation of Pauli flow [28, Theorem 3.1])**.** Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph, c a correction function on Γ, M the flow-demand matrix of Γ, and N the order-demand matrix of Γ. Then there exists a strict partial order $\prec$ such that $(c,\prec)$ is a focused Pauli flow on Γ if and only if $M_\Gamma C=Id_{\bar{O}}$ and $N_\Gamma C$ is the adjacency matrix of a directed acyclic graph, where C is the correction matrix for c.

2.3 Further properties of Pauli flow

We will not need the following definition and lemma individually, but together they yield a useful observation that will simplify handling changes to the partial order of a Pauli flow that result from vertex insertions in later sections.

Definition 2.14 ([28, Definition 2.10])**.** Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph and let c be a correction function on Γ. The *induced relation* $\lhd_c$ is the minimal relation on $\bar{O}$ implied by (P1), (P2), (P3). That is, $u \lhd_c v$ if and only if at least one of the following holds: $v \in c(u) \wedge u \neq v \wedge \lambda(v) \notin \{X,Y\}$ (corresponding to (P1)), $v \in \mathsf{Odd}(c(u)) \wedge u \neq v \wedge \lambda(v) \notin \{Y,Z\}$ (corresponding to (P2)), and $v \in \mathsf{Odd}[c(u)] \wedge u \neq v \wedge \lambda(v) = Y$ (corresponding to (P3)). If the transitive closure of $\lhd_c$ is a partial order, denote it by $\prec_c$ and call it the *induced partial order of* c.

Lemma 2.15 ([28, Lemma 3.12])**.** Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph and let c be a focused correction function on Γ. Then, for any $u,v \in \bar{O}$ such that $u \neq v$, we have $(NC)_{u,v} = 1$ if and only if $v \lhd_c u$. In other words, the off-diagonal part of the matrix NC encodes the relation $\lhd_c$.

Combining Definition 2.14 and Lemma 2.15, we find the following.

Observation 2.16. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph and let C be the matrix of a focused correction function c on Γ. Then, for any $u,v \in \bar{O}$ such that $u \neq v$, we have $(NC)_{u,v} = 1$ if and only if at least one of the following holds:

• $v \in c(u) \wedge \lambda(v) \notin \{X,Y\}$, • $v \in \mathsf{Odd}(c(u)) \wedge \lambda(v) \notin \{Y,Z\}$, • $v \in \mathsf{Odd}[c(u)] \wedge \lambda(v) = Y$.

Using the property that c is focused (i.e. Definition 2.6 holds), the above conditions reduce to $v \in c(u)$ and $\lambda(v) = XY$, or $v \in \mathsf{Odd}(c(u))$ and $\lambda(v) \in \{XZ,YZ\}$.

The induced partial order of Definition 2.14 plays an important role: if a correction function c forms a Pauli flow with some partial order $\prec$, then $\prec_c \subseteq \prec$, i.e. any Pauli flow partial order must contain the induced partial order [28, Lemma 2.11]. Moreover, there exists a partial order $\prec$ such that $(c,\prec)$ is a Pauli flow if and only if $(c,\prec_c)$ is a Pauli flow [28, Theorem 2.12]. Therefore, in Sections 4 and 5, we will generally work with induced partial orders as that simplifies the proofs. By the above results, this is without loss of generality. One advantage of the induced partial order for a *focused* correction function is that all Pauli measurements are initial, which can be seen e.g. from Observation 2.16.

While in this paper we will consider the conditions under which inserting planar measured vertices into one-way computations preserves the existence of flow, for Pauli-Z measured vertices, this question has already been resolved: such vertices can be inserted with any set of neighbours.

Proposition 2.17 (Z-insertion [23, Proposition 4.1])**.** Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph with Pauli flow, where $G = (V,E)$ and let $W \subseteq V$ be some arbitrary subset of the vertices. Then $\Gamma' = (G',I,O,\lambda')$ has Pauli flow, where $G' = (V',E')$ with $V' = V \cup \{z\}$, $E' = E \cup \{\{w,z\} \mid w \in W\}$, and $\lambda'(v) = \lambda(v)$ for $v \neq z$ and $\lambda'(z) = Z$.

2.4 ZX-calculus toolkit for MBQC

Labelled open graphs provide a way to reason about the properties of a resource state that are necessary for robust determinism. Yet we are interested in rewrites that preserve both the existence of flow and also the computation itself. The latter is ensured by the ZX-calculus [7], a graphical language for reasoning about quantum processes. We give a short introduction, more detail can be found e.g. in [8, 34].

Definition 2.18. The ZX-calculus is the language generated by green Z-spiders, red X-spiders and yellow Hadamard nodes with the following interpretations:

$$\left[\!\!\left[\;\alpha\;\right]\!\!\right] = |0...0\rangle\langle 0...0| + e^{i\alpha}\,|1...1\rangle\langle 1...1|$$

$$\left[\!\!\left[\;\alpha\;\right]\!\!\right] = |+...+\rangle\langle +...+| + e^{i\alpha}\,|-...-\rangle\langle -...-|$$

$$[\!\![\;\square\;]\!\!] = |0\rangle\langle +| + |1\rangle\langle -|$$

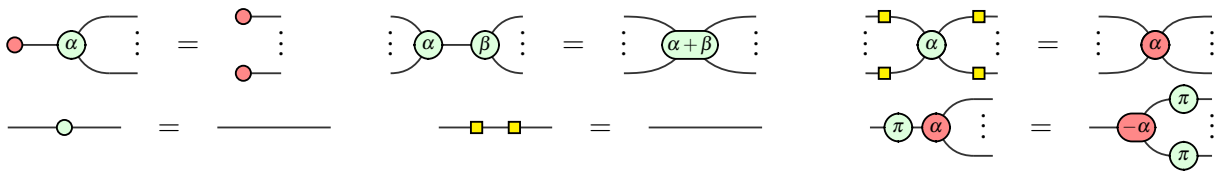


Figure 1: The above rewrite rules for the ZX-calculus are sound.

When green or red nodes have phase $\alpha \equiv 0 \bmod 2\pi$, we omit the phase label in the diagram. The components compose in parallel and in series like gates in a quantum circuit.

A Hadamard node between two spiders can be thought of as a special sort of edge, represented as follows: -○┈┈○- := -○-▪-○-. The ZX-calculus comes with an equational theory, which allows rewriting of diagrams in a way that preserves their interpretation. There are multiple complete sets of rewrite rule [1, 19, 35], but we introduce only a subset of rules that is useful for our purposes, see Figure 1. Furthermore, we implicitly use the *only connectivity matters* rule, which says that ZX-diagrams can in fact be treated as graphs without changing their interpretation. This comes in handy to represent the open graphs underlying one way computations: each qubit can be represented by a green node, connected to its neighbours via Hadamard edges, with a dangling edge on which we can plug the measurement effects, and potentially an additional dangling edge for input qubits. The measurement effects are represented by —(α) for XY, —(α) for YZ, —(π/2)(α) for XZ, —○ for X, —(−π/2) for Y, and —● for Z.

Example 2.19. The open graph where G is drawn on the left below, with $I = \{i_1, i_2\}$ and $O = \{o_1, o_2\}$ is represented the ZX-diagram on the right:

3 YZ-insertion into patterns with extended causal flow

It might seem strange to discuss the insertion of YZ-measured vertices into measurement patterns with causal flow, since the traditional definition of causal flow allows only XY-measured vertices (cf. Definition 2.4). Yet the following generalisation to include YZ-measurements is straightforward[3].

Definition 3.1. Let (G, I, O, λ) be a labelled open graph such that $\lambda(V) \subset \{XY, YZ\}$. G has an *extended causal flow* when there exists a map $c : \overline{O} \to \overline{I}$ and a strict partial order $\prec$ over V such that $\forall u \in V$:

(C1) $u \prec c(u)$ or $u = c(u)$
(C2) $\lambda(u) = XY \Rightarrow c(u) \in N_G(u)$
(C3) $\lambda(u) = YZ \Rightarrow u = c(u)$
(C4) $\forall v \in N_G(c(u)), u \neq v \Rightarrow u \prec v$

Measurement patterns with causal flow are very close to quantum circuits over the CNOT and single-qubit gate set. Allowing YZ-measurements corresponds to allowing a new kind of gate: *phase gadgets*, multi-qubit diagonal gates whose phase depends on the parity of the values of all qubits involved. In other words, an n-qubit phase gadget with phase φ acts on computational basis states as $|x_1 \ldots x_n\rangle \mapsto e^{i\varphi(x_1 \oplus \ldots \oplus x_n)} |x_1 \ldots x_n\rangle$, i.e. it uses XOR where a standard multi-controlled phase gate would use AND.

Definition 3.2. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph. Define $\Gamma' = (G', I, O, \lambda')$ to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood $S \subseteq V$, where

[3] Indeed, while the authors came up with this definition during an internship in the spring/summer of 2023 [29], Calum Holker independently introduced an analogous generalisation around the same time [18, Definition 27].

$G' = (V \cup \{z\}, E \cup \{\{z,v\} \mid v \in S\})$ and $\lambda' : (\bar{O} \cup \{z\}) \to \{XY, YZ\}$ is the extension of λ satisfying $\lambda'(z) = YZ$ and $\lambda'(v) = \lambda(v)$ for all $v \in \bar{O}$.

Observation 3.3. Put together, conditions (C3) and (C4) of Definition 3.1 forbid YZ-measured vertices to be adjacent to one another in a labelled open graph with extended causal flow. We can therefore restrict ourselves to the case $\lambda(S) = \{XY\}$ without loss of generality.

Analogous to gflow or Pauli flow [14, 2, 31], YZ-measured vertices can be removed while preserving the existence of extended causal flow, as stated in the following lemma. Proofs are in Appendix A.

Lemma 3.4. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph, and let Γ' be the labelled open graph from Definition 3.2, with $\lambda(S) \subseteq \{XY\}$. If $(c', \prec')$ is an extended causal flow for Γ', then $(c'_{|V}, \prec'_{|V \times V})$ is an extended causal flow for Γ.

To determine when YZ-insertion into a labelled open graph Γ preserves the existence of extended causal flow, we will first consider a fixed causal flow $(c, \prec)$ for Γ, and then generalise to any causal flow on Γ using Lemma 3.4. Note that the uniqueness result for causal flow correction functions on labelled open graphs where $|I| = |O|$ [11, p. 6] immediately generalises to extended causal flow since there is no choice for the correction set of a YZ-measured vertex. For the first step, we will need to extend the strict partial order $\prec$ to the newly inserted vertex z. The following lemma characterises when this is possible.

Lemma 3.5. Let $\prec$ be a strict partial order on a set V, and suppose $z \notin V$. Let $S_1, S_2 \subset V$, and let $\prec'$ be the transitive closure of $\prec''$, with $\prec'' := \prec \cup (S_1 \times \{z\}) \cup (\{z\} \times S_2)$. Then $\prec'$ is a strict order if and only if $\forall v \in S_1, \forall w \in S_2. \neg(w \prec v \vee v = w)$.

In other words, provided that no element of S_2 is equal to or precedes any element of S_1 in the original order $\prec$, we can extend the order $\prec$ to include z such that z succeeds all elements of S_1 and precedes all elements of S_2.

Due to the above lemma, it is natural to define an extended causal flow $(c', \prec')$ for Γ' given an extended causal flow $(c, \prec)$ for Γ as follows.

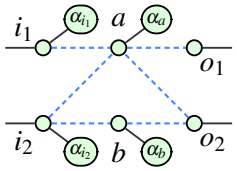
Definition 3.6. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph with extended causal flow $(c, \prec)$, and let Γ' be the labelled open graph from definition 3.2, with $\lambda(S) \subseteq \{XY\}$. Further define c' to be the extension of c to domain $(V \setminus O) \cup \{z\}$ that satisfies $c'(z) = z$, and let $\prec'$ be the transitive closure of $\prec \cup (\{z\} \times S) \cup (c^{-1}(S) \times \{z\})$.

We can now state the main theorem of this section, characterising when the insertion of a YZ-measured vertex into a labelled open graph preserves the existence of extended causal flow.

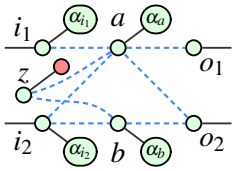
Theorem 3.7. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph. Let $\Gamma' = (G', I, O, \lambda')$ be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood $S \subseteq V$ as in Definition 3.2, with $\lambda(S) \subseteq \{XY\}$. Then Γ' has extended causal flow if and only if there exists an extended causal flow $(c, \prec)$ for Γ such that $\forall v \in S, \forall w \in c^{-1}(S). \neg(v \prec w \vee v = w)$.

In other words we can extend the causal flow to Γ' if and only if no vertex of S is equal to or precedes any vertex of $c^{-1}(S)$ in the original order $\prec$.

Example 3.8. Consider YZ-insertion into the labelled open graph Γ in (1), which has extended causal flow $(c, \prec)$ with c given in (1) and $i_1 \prec i_2 \prec b \prec a$. Let $S = \{a, b\}$. Then we have $c^{-1}(S) = \{i_1, i_2\}$, and $\forall v \in S, \forall w \in c^{-1}(S). \neg(v \prec w \vee v = w)$, so by Theorem 3.7, the labelled open graph Γ' of (1) has extended causal flow where $c'(z) = z$, $c'(v) = c(v)$ otherwise, and $i_1 \prec' i_2 \prec' z \prec' b \prec' a$.

Γ : 

$$c : \begin{cases} i_1 & \mapsto & a \\ i_2 & \mapsto & b \\ a & \mapsto & o_1 \\ b & \mapsto & o_2 \end{cases}$$

Γ' : 

(1)

As a straightforward consequence of Theorem 3.7, if the YZ-insertion is to be flow preserving, then the number of neighbours of the new output can be no more than the number of outputs.

Corollary 3.9. Suppose inserting a YZ-measured vertex z with neighbourhood S as in Definition 3.2 into a labelled open graph (G,I,O,λ) preserves the existence of extended causal flow. Then $|S| \leq |O|$.

We consider now the complexity of checking the conditions of Theorem 3.7. For comparison, the complexity of the (XY-only) causal flow finding algorithm is linear in the number of edges in the labelled open graph [26]. The algorithm generalises to extended causal flow by correcting YZ-measurements as soon as all their neighbours have been corrected. Extended causal flows are unique for labelled open graphs where $|I| = |O|$ since causal flows are unique [11] and each YZ-measured vertex corrects itself.

The flow-preservation conditions involve only the neighbours of the new vertex and their preimages under the flow function. On the other hand, the theorem gives an existence condition, so for labelled open graphs with $|I| < |O|$, different extended causal flows may need to be considered. Additionally, local verification of the partial order conditions may not be possible since the flow finding algorithm does not return the induced partial order, but a layering of the vertices [26]. Thus the flow-preservation conditions for YZ-insertion into extended causal flow may be of theoretical more than practical interest.

4 YZ-insertion into patterns with gflow and Pauli flow

We will now consider the insertion of YZ-measured vertices into MBQC patterns with gflow and Pauli flow. As gflow is a special case of Pauli flow, the two can be considered together: if a pattern has Pauli flow and all the measurements are planar, then it has gflow. Inserting a YZ-measurement, which is planar, does not change that. Thus, considering only Pauli flow is without loss of generality. Furthermore, by Lemma 2.7, it suffices to consider only focused Pauli flow. Throughout this section we will assume that in any Pauli flow $(c,\prec)$, $\prec$ is the partial order induced by c (cf. Definition 2.14), which is also without loss of generality. We begin by proving an intermediate lemma that will be useful for the main theorem. Proofs for this section are in Appendix B.

Lemma 4.1. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph with flow-demand matrix M. Let $\mathscr{A} \subseteq \bar{I}$ and denote the corresponding indicator vector (within $\bar{I}$) by $\mathbf{a}$: i.e. $a_u = 1 \Leftrightarrow u \in \mathscr{A}$ and $a_u = 0 \Leftrightarrow u \in \bar{I} \setminus \mathscr{A}$. Then $\mathscr{A}$ is focused over the set $S \subseteq \bar{O}$ if and only if $(M\mathbf{a})_u = 0$ for all $u \in S$.

We are now ready to define the YZ-insertion and prove the associated flow-preservation theorem for gflow and Pauli flow.

Definition 4.2. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph. Define $\Gamma' = (G',I,O,\lambda')$ to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood $S \subseteq V$, where $G' = (V \cup \{z\}, E \cup \{\{z,v\} \mid v \in S\})$, and $\lambda' : (\bar{O} \cup \{z\}) \to \{X,Y,Z,XY,XZ,YZ\}$ is the extension of λ satisfying $\lambda'(z) = YZ$ and $\lambda'(v) = \lambda(v)$ for all $v \in \bar{O}$.

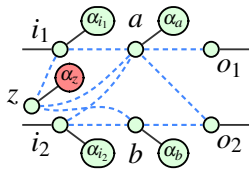
In the following, (unprimed) $\bar{I}$, $\bar{O}$, $\mathscr{L}$ etc. will always refer to the vertices of the original labelled open graph Γ only. The YZ-insertion with a given set of neighbours is flow-preserving only if certain properties are satisfied: there must be a suitable focused correction set for the newly-inserted vertex, and the relation induced by the new correction function must be a strict partial order. The first two conditions of the theorem ensure the existence of the focused correction set and the third condition captures the requirements for the partial order.

Theorem 4.3. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph and let Γ' be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood $S \subseteq V$ as in Definition 4.2. Then Γ' has Pauli flow if and only if there exists a Pauli flow $(c,\prec)$ on Γ and a set $K \subseteq \bar{I}$ such that:

1. K is focused over $\bar{O}\setminus(S\cap\mathscr{X})$, and moreover $\bar{O}\setminus(S\cap\mathscr{X})$ is the largest set over which K is focused.
2. $|S\cap K|\equiv 0 \bmod 2$.
3. If $\mathscr{W}_{pred} := \{w\in\bar{O} : |c(w)\cap S|\equiv 1 \bmod 2\}$ and $\mathscr{V}_{succ} := \mathscr{L}\cap(K\cup(\mathsf{Odd}_G(K)\,\Delta\,S))$, then all $w\in\mathscr{W}_{pred}$ and all $v\in\mathscr{V}_{succ}$ satisfy $\neg(v\prec w\vee v=w)$; i.e. no vertex of $\mathscr{V}_{succ}$ is equal to or precedes any vertex of $\mathscr{W}_{pred}$ for the original order $\prec$.

If the properties hold, the focused Pauli flow on $\Gamma'$is given by the extension c' of c to domain $\bar{O}\cup\{z\}$ satisfying $c'(z)=K\cup\{z\}$, and the transitive closure of $\prec\cup\{(w,z)\mid w\in\mathscr{W}_{pred}\}\cup\{(z,v)\mid v\in\mathscr{V}_{succ}\}$.

Example 4.4. Consider the labelled open graph Γ from Example 3.8, this has focused Pauli flow $(c,\prec)$ (which is actually a gflow) with $c(i_1)=\{a,b\}$, $c(i_2)=\{b\}$, $c(a)=\{o_1\}$, $c(b)=\{o_1,o_2\}$ and $i_1,i_2\prec a,b$. Let $S=\{i_1,a,b\}$ and $K=\{a,b,o_2\}$. First, $\bar{O}\setminus(S\cap\mathscr{X})=\{i_2\}$. As $\mathsf{Odd}_G(K)=\{i_1,a,b\}$, one can check that indeed $\{i_2\}$ is the largest set on which K is focused. Furthermore, $|K\cap S|=|\{1,2\}|\equiv 0 \bmod 2$. Finally, $\mathscr{W}_{pred}=\{i_2\}$ and $\mathscr{V}_{succ}=\{a,b\}$, thus for all $w\in\mathscr{W}_{pred}$ and $v\in\mathscr{V}_{succ}$ we have $\neg(v\prec w\vee v=w)$. Thus by Theorem 4.3, the following labelled open graph has focused gflow $(c',\prec')$:



with $c'(z)=\{z,1,2,o_2\}$, and $\begin{cases} i_2 \prec' z \prec' a,b \prec' o_1,o_2 \\ \qquad\; i_1 \prec' a,b \prec' o_1,o_2. \end{cases}$

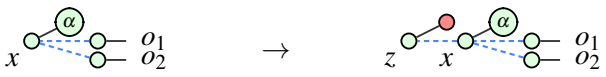
It has long been known that the 'Z-deletion' rule is sound also for YZ-measurements: they can always be deleted from a labelled open graph without breaking the existence of flow [14, 2, 31]. This is captured by our theorem via the following corollary, which follows from the second part of the proof above.

Corollary 4.5. The three conditions in Theorem 4.3 are satisfied for any focused Pauli flow on Γ', therefore YZ-deletion is always flow-preserving.

In the other direction, for YZ-insertion, it must be checked each time whether the three conditions are satisfied by the chosen set of neighbours S of the new vertex and the proposed correction set (minus the vertex itself). Yet there are some cases where the conditions are always satisfied: for example, Z-measured neighbours 'come for free' in the sense that they will not affect whether or not the YZ-insertion is flow-preserving: this can be seen by inserting the YZ-measurement u with non-Z neighbours only, and then in turn deleting each desired Z-measured neighbour and re-inserting it with the extra edge to u. As Z-deletions and -insertions always preserve Pauli flow (cf. Proposition 2.17), the entire process is flow-preserving if and only if the original YZ-insertion preserves the existence of flow. Any proof that the old labelled open graph has Pauli flow if the new labelled open graph has it follows immediately from Corollary 4.5, so we only have to show that Γ having Pauli flow implies that Γ' has Pauli flow. Two additional corollaries handling all-input and all-output neighbours, respectively, are in Appendix B.

Corollary 4.6. Let $\Gamma=(G,I,O,\lambda)$ be a labelled open graph with focused Pauli flow $(c,\prec)$ and let $x\in V$ be s.t. $\lambda(x)=XY$. Define Γ' to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood $S=\{x\}$ as in Definition 4.2. Then Γ' has focused Pauli flow $(c',\prec')$ where c' is the extension of c satisfying $c'(z)=c(x)\cup\{z\}$, and $\prec'=\prec\cup\{(w,z)\mid w\prec x\}\cup\{(z,v)\mid x\prec v\}$.

Remark 4.7. Corollary 4.6 shows one way of getting a Pauli flow on Γ' for any focused Pauli flow on the original labelled open graph Γ. Yet not all focused Pauli flows on Γ' necessarily arise in this way: consider, for example, the line graph $G=(\{x,o_1,o_2\},\{\{o_1,x\},\{x,o_2\}\})$ and the resulting labelled open graph $(G,\emptyset,\{o_1,o_2\},x\mapsto XY)$.



This has two focused Pauli flows, with $c(x) = \{o_1\}$ or $c(x) = \{o_2\}$. After inserting a new YZ-type measurement z with sole neighbour x, there are four focused Pauli flows on the resulting labelled open graph: either of the correction sets $\{z, o_1\}$ and $\{z, o_2\}$ works for z, independent of the choice of correction set for x.

If a labelled open graph represents a unitary operator – i.e. the number of outputs equals the number of inputs – then focused Pauli flow, if it exists, is unique (since the flow-demand matrix is square and thus has at most one right inverse). In this case, the possible correction set of the new vertex is entirely determined by the choice of S.

Observation 4.8. Suppose $\Gamma = (G, I, O, \lambda)$ is a labelled open graph such that $|I| = |O|$, and it has Pauli flow. Let $S \subseteq V$ and define Γ' to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood S as in Definition 4.2. Write M for the flow-demand matrix of Γ, $\mathbf{x}$ for the indicator vector of $S \cap \mathscr{X}$ in $\bar{O}$, and $\mathbf{k}$ for the indicator vector of an (as yet unknown) set $K \subseteq \bar{I}$ such that $K \cup \{z\}$ is a focused correction set for z. Condition 1 is equivalent to $M\mathbf{k} = \mathbf{x}$ by Lemma 4.1.

Since Γ has Pauli flow, M has a right inverse C. Now, $|I| = |O|$ implies that M is square, therefore C is in fact the unique two-sided inverse for M. Hence $M\mathbf{k} = \mathbf{x}$ if and only if $\mathbf{k} = C\mathbf{x}$. The set corresponding to $C\mathbf{x}$ satisfies condition 1 by construction, yet it need not satisfy the other two conditions of Theorem 4.3.

In the general case, it is difficult to analyse the complexity of checking the conditions in Theorem 4.3 for a given set of neighbours S as the theorem requires only that there exists a Pauli flow which satisfies them. Yet if the labelled open graph satisfies $|I| = |O|$ as in Observation 4.8, then focused Pauli flow is unique and that difficulty disappears. Suppose this unique Pauli flow on Γ is known in the form of the correction matrix C and the product NC, where N is the order-demand matrix. We will assume M and N are known too (otherwise they can be constructed in time $\mathscr{O}(n^2)$). From S, we can construct in linear time the indicator vectors $\mathbf{x}$, $\mathbf{n}$ and $\mathbf{z}$, and then compute in $\mathscr{O}(n^2)$ the indicator vector of the correction set $\mathbf{k} = C\mathbf{x}$. Then, again in $\mathscr{O}(n^2)$, we can compute the additional row and column for the product $N_{\Gamma'}C_{\Gamma'}$. Checking whether the resulting matrix represents a directed acyclic graph is also in $\mathscr{O}(n^2)$ (see [9, Section 20.4]). Thus the entire process of checking the conditions and updating the flow proceeds in $\mathscr{O}(n^2)$, which is more efficient than finding a flow from scratch (the latter has the complexity of matrix multiplication, i.e. $\mathscr{O}(n^{\log_2 7})$ using Strassen's algorithm). Thus, if the number of vertices to be inserted is less than $\mathscr{O}(n^{\log_2 7 - 2})$, it is more efficient to do this step by step and update the known flow, rather than inserting all the vertices and re-running the flow-finding algorithm.

Another consequence of Observation 4.8 is that, if $|S \cap K| \equiv 1 \bmod 2$ for the unique K that satisfies condition 1, then it is not possible to insert a YZ-measured vertex with neighbourhood S, as u would be in the odd neighbourhood of its own correction set, contradicting (P6). Yet in that case, the right-hand side of (P5) would be satisfied for a vertex with that neighbourhood: so it may be possible to insert an XZ-measured vertex with neighbourhood S instead, assuming the partial order conditions hold. Thus, in the next section, we will consider inserting vertices with measurement labels other than YZ (or Z).

5 Inserting vertices with other measurement labels

XZ-measurements behave similarly to YZ-measurements: the focusing conditions are the same for both; equivalently, they give rise to the same types of rows in the flow-demand matrix. The rows of the order-demand matrix corresponding to an XZ- or a YZ-measured vertex are also nearly the same, the only difference is at the intersection of the column corresponding to the same vertex: a '1' in that position encodes that XZ-measurements are in the odd neighbourhood of their own correction set whereas a '0' in

that position encodes that YZ-measurements are not. The flow preservation theorem for YZ insertions can thus straightforwardly be adapted to XZ-measurements instead. Proofs for this section are in Appendix C.

Definition 5.1. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph. Define $\Gamma' = (G',I,O,\lambda')$ to be the labelled open graph that results from inserting a new XZ-measured vertex u with neighbourhood $S \subseteq V$, where $G' = (V \cup \{z\}, E \cup \{\{z,v\} \mid v \in S\})$, and $\lambda' : (\bar{O} \cup \{z\}) \to \{X,Y,Z,XY,XZ,YZ\}$ is the extension of λ satisfying $\lambda'(z) = XZ$ and $\lambda'(v) = \lambda(v)$ for all $v \in \bar{O}$.

Theorem 5.2. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph and let Γ' be the labelled open graph that results from inserting a new XZ-measured vertex z with neighbourhood $S \subseteq V$ as in Definition 5.1. Then Γ' has Pauli flow if and only if there exists a Pauli flow $(c,\prec)$ on Γ and a set $K \subseteq \bar{I}$ such that:

1. K is focused over $\bar{O} \setminus (S \cap \mathscr{X})$, and moreover $\bar{O} \setminus (S \cap \mathscr{X})$ is the largest set over which K is focused.
2. $|S \cap K| \equiv 1 \bmod 2$.
3. Let $\mathscr{W}_{pred} := \{w \in \bar{O} : |c(w) \cap S| \equiv 1 \bmod 2\}$ and $\mathscr{V}_{succ} := \mathscr{L} \cap (K \cup (\mathsf{Odd}_G(K) \,\Delta\, S))$, then all $w \in \mathscr{W}_{pred}$ and all $v \in \mathscr{V}_{succ}$ satisfy $\neg(v \prec w \vee v = w)$; i.e. no vertex of $\mathscr{V}_{succ}$ is equal to or precedes any vertex of $\mathscr{W}_{pred}$ for the original order $\prec$.

If the properties hold, the focused Pauli flow on $\Gamma'$is given by the extension c' of c to domain $\bar{O} \cup \{z\}$ satisfying $c'(z) = K \cup \{z\}$, and the transitive closure of $\prec \cup \{(w,z) \mid w \in \mathscr{W}_{pred}\} \cup \{(z,v) \mid v \in \mathscr{V}_{succ}\}$.

Proof. The proof is analogous to that of Theorem 4.3 with the only difference being that $N_{\Gamma'} = \left(\begin{smallmatrix} N & \mathbf{z} \\ \mathbf{n} & 1 \end{smallmatrix}\right)$, which corresponds to the changed requirement that $|S \cap K| \equiv 1 \bmod 2$. □

We have now shown how to insert YZ or XZ-measured vertices while preserving flow, and it was already known that Z-insertion preserves the existence Pauli flow (cf. Proposition 2.17). Now, inserting an X,Y or XY measurements into an MBQC pattern must change the interpretation, so it does not make sense to look for insertion rules for those measurement types. Nevertheless, we can use the existing insertion rules in and then apply local complementations or pivots that change the measurement label of the new vertex. This effectively inserts a measurement of a different type into the pattern, at the cost of introducing additional changes elsewhere in the labelled open graph.

Indeed, with just one appeal to standard ZX-calculus rewrite rules to handle phase labels and a new lemma that explicitly shows how flow changes under the pivot operation, we can use YZ-insertions to prove the 'vertex splitting' or 'neighbour unfusion' rule that has previously been used in various forms in the literature about rewriting MBQC and ZX-calculus diagrams [32, 24, 18]. This connection was previously pointed out in [24, p. 215]. The vertex splitting rule always preserves Pauli flow if the new phase-0 vertex is taken to be an X-measurement [24, Corollary 6.1], yet if that vertex is taken to be an XY-measurement, flow is not always preserved [32, Section 5].

Lemma 5.3. Suppose the labelled open graph (G,I,O,λ) has Pauli flow $(c,\prec)$. Let $u,v \in \bar{I} \cap \bar{O}$ such that $\{u,v\} \in E$. Define $\lambda' : \bar{O} \to \{XY,XZ,YZ,X,Y,Z\}$ and $c' : \bar{O} \to 2^{\bar{I}}$ as

$$\lambda'(w) := \begin{cases} Z & \text{if } w \in \{u,v\} \wedge \lambda(w) = X \\ X & \text{if } w \in \{u,v\} \wedge \lambda(w) = Z \\ YZ & \text{if } w \in \{u,v\} \wedge \lambda(w) = XY \\ XY & \text{if } w \in \{u,v\} \wedge \lambda(w) = YZ \\ \lambda(w) & \text{otherwise.} \end{cases} \qquad c'(w) := \begin{cases} c(w) & \text{if } u,v \notin \mathsf{Odd}_G[c(w)] \\ c(w) \,\Delta\, \{u\} & \text{if } u \in \mathsf{Odd}_G[c(w)] \not\ni v \\ c(w) \,\Delta\, \{v\} & \text{if } u \notin \mathsf{Odd}_G[c(w)] \ni v \\ c(w) \,\Delta\, \{u,v\} & \text{if } u,v \in \mathsf{Odd}_G[c(w)], \end{cases}$$

then $(G \wedge uv,I,O,\lambda')$ has Pauli flow $(c',\prec)$. Moreover, for any $w \in \bar{O}$, we have $\mathsf{Odd}_{G \wedge uv}[c'(w)] = \mathsf{Odd}_G[c(w)]$. If $(c,\prec)$ is focused and $\lambda(u),\lambda(v) \neq XZ$, then $(c',\prec)$ is also focused. Finally, if $(c,\prec)$ is a gflow, then $(c',\prec)$ is also a gflow.

Interestingly, this lemma is rather simpler than equivalent ones for local complementations [2, 31], despite pivots arising from the former. We are now ready to consider the vertex splitting operation.

Definition 5.4. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph let $x \in \bar{O} \cap \bar{I}$ be such that $\lambda(x) = XY$, and let $W \subseteq V \setminus \{x\}$. The labelled open graph $\Gamma' = (G',I,O,\lambda')$ that results from splitting x over the set W is defined as follows:

- Let $V' = V \cup \{x',x''\}$, i.e. we add two new vertices x',x'' to the graph.
- Let $E' = E \,\Delta\, \{\{x,x'\},\{x',x''\}\} \,\Delta\, (\{x,x''\} \times W)$, i.e. we connect x to x' and x' to x'', toggle the edges between x and elements of W, and add edges between x'' and elements of W.
- Let λ' be the extension of λ to domain V' which satisfies $\lambda'(x') = \lambda'(x'') = XY$.

This definition is slightly more general than those appearing in the literature, which usually restrict W to be a subset of $N_G(x)$ or even just a single element of $N_G(x)$ [32].

We will explicitly state the condition of the vertex splitting rule for the case $|I| = |O|$ only. This is both because the motivating neighbour unfusion rule is mainly used within diagrams arising from circuits – i.e. where the numbers of inputs and outputs are equal – and because the flow-preservation conditions get even more complicated in the general case where we would need to consider two separate unknown sets of vertices, one for the correction set of x' and the other for x''.

Theorem 5.5. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph such that $|I| = |O|$, let $x \in \bar{O} \cap \bar{I}$ be such that $\lambda(x) = XY$, and let $W \subseteq V \setminus \{x\}$. Suppose Γ' is the labelled open graph that results from splitting x over the set W as in Definition 5.4. Then Γ' has Pauli flow if and only if there exists a focused Pauli flow $(c,\prec)$ on Γ such that:

(V1) If K is the unique set that is both focused over $\bar{O} \setminus (W \cap \mathscr{X})$ and has the property that $\bar{O} \setminus (W \cap \mathscr{X})$ is the largest set over which K is focused, then $|W \cap K| \equiv 0 \bmod 2$.

(V2) If $\mathscr{W}_{pred} := \{w \in \bar{O} : |c(w) \cap W| \equiv 1 \bmod 2\}$ and $\mathscr{V}_{succ} := \mathscr{L} \cap (K \cup (\mathsf{Odd}_G(K) \,\Delta\, W))$, then all $w \in \mathscr{W}_{pred}$ and all $v \in \mathscr{V}_{succ}$ satisfy $\neg(v \prec w \vee v = w)$; i.e. no vertex of $\mathscr{V}_{succ}$ is equal to or precedes any vertex of $\mathscr{W}_{pred}$ for the original order $\prec$.

(V3) For all $u \in V$, we have $x \prec u \implies u \notin \mathscr{W}_{pred}$ and $u \prec x \implies u \notin \mathscr{V}_{succ}$; i.e. vertices in $\mathscr{W}_{pred}$ do not succeed x and vertices in $\mathscr{V}_{succ}$ do not precede x for the original order $\prec$.

(V4) $x \in K$ if and only if $|c(x) \cap W| \equiv 0 \bmod 2$.

If all measurements in Γ are planar, then so are all measurements in Γ', thus the same conditions imply preservation of gflow.

Regarding the restricted version of neighbour unfusion, where W consists of exactly one neighbour of x (as used by Staudacher et al. [32]), we find the following corollary.

Corollary 5.6. Let $\Gamma = (G,I,O,\lambda)$ be a labelled open graph such that $|I| = |O|$, let $a \in \bar{O} \cap \bar{I}$ be such that $\lambda(a) = XY$, and let $b \in N_G(a)$ be such that $\lambda(b) = XY$. Suppose Γ' is the labelled open graph that results from splitting a over the set $W := \{b\}$ as in Definition 5.4. Then Γ' has Pauli flow if and only if there exists a focused Pauli flow $(c,\prec)$ on Γ such that:

- Either $a \in c(b)$ or $b \in c(a)$; i.e. one of the two vertices is in the correction set of the other.
- For all $u \in V$, we have $a \prec u \implies \neg(u \prec b)$ and $u \prec a \implies \neg(b \prec u)$; i.e. there is no third vertex in between a and b in the original order $\prec$.

If all measurements in Γ are planar, then so are all measurements in Γ', thus the same conditions imply preservation of gflow.

This shows that the necessary and sufficient flow-preservation criterion derived by McElvanney for neighbour unfusion in the setting of gflow [22] generalises to Pauli flow as well.

Remark 5.7. The conditions of Corollary 5.6 are symmetric under interchange of a and b. This reflects the fact that, when both new vertices are treated as having planar measurements (as we have done), then splitting a over the set $\{b\}$ results in the same labelled open graph as splitting b over the set $\{a\}$.

If only the first insertion of Lemma C.2 is treated as a planar measurement and the second insertion is treated as a Pauli measurement, then the existence of Pauli flow is preserved for all choices of W [24, Proposition 5.1]. On the other hand, gflow is not preserved since Γ' contains at least one non-planar measurement, and the symmetry between a and b is broken.

6 Conclusions

We have derived conditions under which planar-measured qubits can be inserted into one-way computations without breaking different types of flow. This is a step towards a better theoretical understanding of flow-preserving rewriting for MBQC and for the ZX-calculus. In applications such as the reduction of two-qubit gate counts, it is desirable to remain within the framework of gflow instead of inserting Pauli-measured vertices that have a different extraction procedure [32]: we have characterised when this is possible.

By choosing suitable measurement angles, the insertion of planar measurements into a one-way computation or a ZX-diagram can be performed without changing the semantics, i.e. without changing the linear operator that is implemented; this is desired for example in the above two-qubit gate count reduction procedure. Yet by allowing measurement angles to vary, flow-preserving insertion of one (or more) planar measurements can also increase the range of linear operators that are implementable using a given MBQC pattern or ZX-diagram. This has applications in the generation of ansätze for variational eigensolvers within measurement-based [16] or ZX-based settings [15].

In the common case of working with unitary computations or ZX-diagrams (where the number of inputs is equal to the number of outputs), checking the flow-preservation conditions and then updating the gflow or Pauli flow is more efficient than finding a flow on the new labelled open graph from scratch. The same is not guaranteed if there are more outputs than inputs: in that situation, there are generally different flows on the same labelled open graph, and the insertion may be flow-preserving for some and not others. The question of whether there are efficient ways to modify a known flow to make it compatible with an insertion for a given neighbourhood is left to future work.

Acknowledgments

The authors would like to thank Tommy McElvanney, Piotr Mitosek, and Korbinian Staudacher for interesting and useful conversations during the course of the internship on which this paper is based.

References

[1] Miriam Backens (2014): *The ZX-calculus Is Complete for Stabilizer Quantum Mechanics*. *New Journal of Physics* 16(9), p. 093021, doi:10.1088/1367-2630/16/9/093021. arXiv:1307.7025.

[2] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and Back Again: A Circuit Extraction Tale*. *Quantum* 5, p. 421, doi:10.22331/q-2021-03-25-421. arXiv:2003.01664.

[3] André Bouchet (1988): *Graphic Presentations of Isotropic Systems*. *Journal of Combinatorial Theory, Series B* 45(1), pp. 58–76, doi:10.1016/0095-8956(88)90055-X.

[4] Anne Broadbent, Joseph Fitzsimons & Elham Kashefi (2009): *Universal Blind Quantum Computation*. In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 517–526, doi:10.1109/FOCS.2009.36.

[5] Anne Broadbent & Elham Kashefi (2009): *Parallelizing Quantum Circuits*. *Theoretical Computer Science* 410(26), pp. 2489–2510, doi:10.1016/j.tcs.2008.12.046.

[6] Daniel E. Browne, Elham Kashefi, Mehdi Mhalla & Simon Perdrix (2007): *Generalized Flow and Determinism in Measurement-Based Quantum Computation*. *New Journal of Physics* 9(8), p. 250, doi:10.1088/1367-2630/9/8/250.

[7] Bob Coecke & Ross Duncan (2011): *Interacting Quantum Observables: Categorical Algebra and Diagrammatics*. *New Journal of Physics* 13(4), p. 043016, doi:10.1088/1367-2630/13/4/043016. arXiv:0906.4725.

[8] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, Cambridge, doi:10.1017/9781316219317.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein (2022): *Introduction to Algorithms, Fourth Edition*. MIT Press.

[10] Vincent Danos & Elham Kashefi (2006): *Determinism in the One-Way Model*. *Physical Review A* 74(5), p. 052310, doi:10.1103/PhysRevA.74.052310.

[11] Niel de Beaudrap (2008): *Finding Flows in the One-Way Measurement Model*. *Physical Review A* 77(2), p. 022328, doi:10.1103/PhysRevA.77.022328.

[12] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-Diagrams Can Be #P-Hard*. In: *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 229, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 119:1–119:19, doi:10.4230/LIPIcs.ICALP.2022.119.

[13] Giovanni de Felice, Boldizsár Poór, Lia Yeh & William Cashman (2024): *Fusion and Flow: Formal Protocols to Reliably Build Photonic Graph States*, doi:10.48550/arXiv.2409.13541. arXiv:2409.13541.

[14] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-Theoretic Simplification of Quantum Circuits with the ZX-calculus*. *Quantum* 4, p. 279, doi:10.22331/q-2020-06-04-279. arXiv:1902.03178.

[15] Tom Ewen, Ivica Turkalj, Patrick Holzer & Mark-Oliver Wolf (2025): *Application of ZX-calculus to quantum architecture search*. *Quantum Machine Intelligence* 7(1), doi:10.1007/s42484-025-00264-6.

[16] R. R. Ferguson, L. Dellantonio, A. Al Balushi, K. Jansen, W. Dür & C. A. Muschik (2021): *Measurement-Based Variational Quantum Eigensolver*. *Physical Review Letters* 126(22), p. 220501, doi:10.1103/PhysRevLett.126.220501.

[17] Alexandru Gheorghiu, Theodoros Kapourniotis & Elham Kashefi (2019): *Verification of Quantum Computation: An Overview of Existing Approaches*. *Theory of Computing Systems* 63(4), pp. 715–808, doi:10.1007/s00224-018-9872-3.

[18] Calum Holker (2023): *Causal Flow Preserving Optimisation of Quantum Circuits in the ZX-calculus*, doi:10.48550/arXiv.2312.02793. arXiv:2312.02793.

[19] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, ACM, New York, NY, USA, pp. 559–568, doi:10.1145/3209108.3209131.

[20] Theodoros Kapourniotis, Elham Kashefi, Dominik Leichtle, Luka Music & Harold Ollivier (2024): *Unifying Quantum Verification and Error-Detection: Theory and Tools for Optimisations*. *Quantum Science and Technology* 9(3), p. 035036, doi:10.1088/2058-9565/ad466d.

[21] Elham Kashefi, Damian Markham, Mehdi Mhalla & Simon Perdrix (2009): *Information Flow in Secret Sharing Protocols*. *Electronic Proceedings in Theoretical Computer Science* 9, pp. 87–97, doi:10.4204/EPTCS.9.10. arXiv:0909.4479.

[22] Tommy McElvanney (2025): *Preservation of determinism in MBQC under ZX-calculus rewrites*. Ph.D. thesis, University of Birmingham. To appear.

[23] Tommy McElvanney & Miriam Backens (2023): *Complete Flow-Preserving Rewrite Rules for MBQC Patterns with Pauli Measurements*. *Electronic Proceedings in Theoretical Computer Science* 394, pp. 66–82, doi:10.4204/EPTCS.394.5.

[24] Tommy McElvanney & Miriam Backens (2023): *Flow-Preserving ZX-calculus Rewrite Rules for Optimisation and Obfuscation*. *Electronic Proceedings in Theoretical Computer Science* 384, pp. 203–219, doi:10.4204/EPTCS.384.12. arXiv:2304.08166.

[25] Mehdi Mhalla, Mio Murao, Simon Perdrix, Masato Someya & Peter S. Turner (2014): *Which Graph States Are Useful for Quantum Information Processing?* 6745, pp. 174–187, doi:10.1007/978-3-642-54429-3_12. arXiv:1006.2616.

[26] Mehdi Mhalla & Simon Perdrix (2008): *Finding Optimal Flows Efficiently*. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir & Igor Walukiewicz, editors: *Automata, Languages and Programming*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 857–868, doi:10.1007/978-3-540-70575-8_70.

[27] Mehdi Mhalla, Simon Perdrix & Luc Sanselme (2025): *Shadow Pauli Flow: Characterising Determinism in MBQCs Involving Pauli Measurements*, doi:10.48550/arXiv.2207.09368. arXiv:2207.09368.

[28] Piotr Mitosek & Miriam Backens (2024): *An Algebraic Interpretation of Pauli Flow, Leading to Faster Flow-Finding Algorithms*, doi:10.48550/arXiv.2410.23439. arXiv:2410.23439.

[29] Thomas Perez (2023): *Measurement-Based Quantum Computing : Inserting YZ-measured Qubits in MBQC Patterns While Preserving Flow Conditions*. M1 Internship Report, University of Birmingham.

[30] Robert Raussendorf & Hans J. Briegel (2001): *A One-Way Quantum Computer*. *Physical Review Letters* 86(22), pp. 5188–5191, doi:10.1103/PhysRevLett.86.5188.

[31] Will Simmons (2021): *Relating Measurement Patterns to Circuits via Pauli Flow*. *Electronic Proceedings in Theoretical Computer Science* 343, pp. 50–101, doi:10.4204/EPTCS.343.4. arXiv:2109.05654.

[32] Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann & Wolfgang Gehrke (2023): *Reducing 2-QuBit Gate Count for ZX-Calculus Based Quantum Circuit Optimization*. *EPTCS* 394, pp. 29–45, doi:10.4204/EPTCS.394.3.

[33] S. Takeda & A. Furusawa (2019): *Toward Large-Scale Fault-Tolerant Universal Photonic Quantum Computing*. *APL Photonics* 4(6), p. 060902, doi:10.1063/1.5100160.

[34] John van de Wetering (2020): *ZX-calculus for the Working Quantum Computer Scientist*, doi:10.48550/arXiv.2012.13966. arXiv:2012.13966.

[35] Renaud Vilmart (2019): *A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–10, doi:10.1109/LICS.2019.8785765.

[36] Felix Zilk, Korbinian Staudacher, Tobias Guggemos, Karl Fürlinger, Dieter Kranzlmüller & Philip Walther (2022): *A Compiler for Universal Photonic Quantum Computers*. In: *2022 IEEE/ACM Third International Workshop on Quantum Computing Software (QCS)*, IEEE, Dallas, TX, USA, pp. 57–67, doi:10.1109/QCS56647.2022.00012.

A Proofs for Section 3

Proof of Lemma 3.4. Let $u \in V$. Assume for a contradiction that $c'(u) = z$. As $u \neq z$, we have by condition (C3) that $\lambda(u) = XY$, which implies by condition (C2) that $z \in N_G(u)$. Then, condition (C4) gives $z = c'(u) \prec' u$, which contradicts condition (C1).

Therefore, $c'^{-1}(z) = \{z\}$, and $c'_{|V}$ is well defined.

We have that $c'_{|V}(u) = c'(u) \in V$. Then as $u \prec' c'(u)$ or $u = c'(u)$, we have $u \prec'_{|V\times V} c'_{|V}(u)$ or $u = c'_{|V}(u)$. Condition (C1) is therefore satisfied. Similarly, as the connectivity, labelling and correction are the all induced by Γ', and as nothing changes on V, conditions (C2), (C3) and (C4) are satisfied. □

Proof of Lemma 3.5. Let's first show the direct implication by contraposition.

Let $v, w \in S_1 \times S_2$ such that $w \prec v \vee v = w$. We have $v \in S_1 \Rightarrow v \prec' z$ and $w \in S_2 \Rightarrow z \prec' w$ which gives $v \prec' w$. As we have $w \prec v \vee v = w$, in both cases we get $w \prec' w$, and therefore $\prec'$ is not strict.

Now, for the converse implication, suppose that for all $v \in S_1, w \in S_2. \neg(w \prec v \vee v = w)$. Let us show that $\prec'$ is a strict order.

We directly have that $\prec'$ is transitive as the transitive closure of $\prec''$. Assume for a contradiction that it is not strict.

Let $v \in V \cup \{z\}$ such that $v \prec' v$. Then as $\prec'$ is the transitive closure of $\prec''$, we have

$$\exists n > 1 : \exists (v_1, \dots, v_n) \in V : v = v_1 \prec'' v_2 \prec'' \dots \prec'' v_n = v$$

Denote by $[n]$ the set of all integers between 1 and n (inclusive).

If $n = 2$, we have $v \prec'' v$, which can only be true if $v \prec v$ due to the definition of $\prec''$. This contradicts the strictness of $\prec$.

If $n > 2$, the same argument can be used if we have $v_1 \prec v_2 \prec \dots \prec v_n$.

If not, the construction of $\prec''$ gives that

$$\exists i \in [n] : v_i = z$$

If $v \neq z$, denote

$$i_{min} = \min\{i \in [n-1] \setminus \{1\} \mid v_i = z\}, \quad i_{max} = \max\{i \in [n-1] \setminus \{1\} \mid v_i = z\}.$$

Then if we denote $\preceq = \prec \cup =$, we have that $v \preceq v_{i_{min}-1}$ and $v_{i_{max}+1} \preceq v$. This gives $v_{i_{max}+1} \preceq v \preceq v_{i_{min}-1}$ with $v_{i_{min}-1} \in S_1$, and $v_{i_{max}+1} \in S_2$ giving that either $v_{i_{max}+1} \prec v_{i_{min}-1}$ or $v_{i_{max}+1} = v_{i_{min}-1}$, which contradicts the order hypothesis in both cases.

Now if $v = z$, we have that $v_2 \neq z$. We then have $v_2 \prec' z \prec' v_2$. We can then use the previous argument with $v \leftarrow v_2$ to get a contradiction. □

Proof of Theorem 3.7. Let us begin with the direct implication.

Let's suppose that G' has an extended causal flow $(c', \prec')$. Then let's suppose for a contradiction that

$$\exists v \in S : \exists w \in c'^{-1}(S) : v \prec w$$

As $v \in S$ we have $v \in N_{G'}(z) = N_{G'}(c'(z))$ and therefore $z \prec' v$. Besides, $w \in c'^{-1}(S)$ gives that $z \in N_{G'}(c'(w))$, and therefore $w \prec' z$. This gives $z \prec' z$ which contradicts the strictness of $\prec'$. A similar argument holds if $\exists v \in S : \exists w \in c'^{-1}(S) : v = w$.

Therefore, $(c', \prec')$ is such that $\forall v \in S, \forall w \in c'^{-1}(S), \neg(v \prec' w)$. Using Lemma 3.4 to remove z, we get an extended causal flow $(c, \prec)$ for (G, I, O, λ) with the same property, implying $\forall v \in S, \forall w \in c^{-1}(S), \neg(v \prec w \vee v = w)$.

Let us now show the converse implication. Let $(c, \prec)$ be an extended causal flow for (G, I, O, λ) such that $\forall v \in S, \forall w \in f^{-1}(S), \neg(v \prec w \vee v = w)$. Let $(c', \prec')$ be as in Definition 3.6. Using Lemma 3.5 with $S_1 \leftarrow c^{-1}(S)$ and $S_2 \leftarrow S$ we get that $\prec'$ is a strict order.

Let us now show that $(c', \prec')$ is an extended causal flow. One can easily check that $(c', \prec')$ satisfies conditions (C1), (C2), (C3) of Definition 3.1 by construction. Let's elaborate on why it satisfies condition (C4).

Let $v \in V \cup \{z\}$ and $w \in N_{G'}(c(v))$ such that $v \neq w$. Let's show that $v \prec' w$:

If $v = z$, then $N_{G'}(c'(v)) = S$, thus $w \in S$, and by construction $v \prec' w$.

Otherwise, $v \neq z$, and then $c'(v) = c(v) \neq z$. Recall that $E' = E \cup \{\{z,v\} \mid v \in S\}$. Then as we have $w \in N_{G'}(c(v))$, either $w \in N_G(c(v))$ or $\{w, c(v)\} \in \{\{z,v\} \mid v \in S\}$.

If $w \in N_G(c(v))$, as $v \neq w$ we get that $v \prec w$ from condition (C4) of $(c, \prec)$ being an extended causal flow. By construction we get $v \prec' w$.

Otherwise, $w = z$ and $c(v) \in S$, as the case $c(v) = z$ would imply $v = z$. Then $v \in c^{-1}(S)$. Therefore by construction we have $v \prec' w$.

Therefore $v \prec' w$. Therefore Γ' has extended causal flow.

□

Proof of Corollary 3.9. If all measurements are XY, the causal flow function c defines a path covering of the labelled open graph, in which each path is given by a finite sequence $v, c(v), c(c(v)), \ldots$ for some $v \in V$, with the final element of the sequence being an output. Thus the number of paths is equal to the number of outputs [11, Section III]. By Lemma 3.4, this generalises to extended causal flow, where the paths cover only the XY-measured vertices.

Suppose for a contradiction that $|S| > |O|$, then by the pigeonhole principle there exist two distinct vertices $s_1, s_2 \in S$ which belong to the same path. Without loss of generality assume s_2 is the one that is closer to the output, i.e. $s_2 = c^j(s_1)$ for some positive integer j. But then $c^{j-1}(s_1) = c^{-1}(s_2)$ and thus $s_1 = c^{-1}(s_2) \vee s_1 \prec c^{-1}(s_2)$ by repeated application of (C1). This contradicts the flow-preservation condition of Theorem 3.7. □

B Proofs for Section 4

Proof of Lemma 4.1. Lemma 2.11 implies that for any $u \in \bar{O}$, $(M\mathbf{a})_u = 0$ if and only if

$$\begin{aligned} \lambda(u) \in \{X, XY\} \quad &\text{and} \quad u \notin \mathsf{Odd}(\mathscr{A}), \text{or} \\ \lambda(u) \in \{XZ, YZ, Z\} \quad &\text{and} \quad u \notin \mathscr{A}, \text{or} \\ \lambda(u) = Y \quad &\text{and} \quad u \notin \mathsf{Odd}[\mathscr{A}]. \end{aligned}$$

First, assume $\mathscr{A}$ is focused and consider each focusing condition in turn.

- (F1) states $w \in S \cap \mathscr{A} \implies \lambda(w) \in \{XY, X, Y\}$, so if $w \in S$ satisfies $\lambda(w) \in \{XZ, YZ, Z\}$, then $w \notin \mathscr{A}$, which by the above implies $(M\mathbf{a})_w = 0$.
- (F2) states $w \in S \cap \mathsf{Odd}(\mathscr{A}) \implies \lambda(w) \in \{XZ, YZ, Y, Z\}$, so if $w \in S$ satisfies $\lambda(w) \in \{X, XY\}$, then $w \notin \mathsf{Odd}(\mathscr{A})$, which again implies $(M\mathbf{a})_w = 0$.
- (F3) states $w \in S$ and $\lambda(w) = Y \Rightarrow w \notin \mathsf{Odd}[\mathscr{A}]$, thus once more $(M\mathbf{a})_w = 0$.

The vertices in S are non-outputs and – between the three conditions – we have considered all the measurement labels, so we may conclude $(M\mathbf{a})_w = 0$ for all $w \in S$ as desired.

Conversely, assume $(M\mathbf{a})_u = 0$ for all $u \in S$. Distinguish cases according to the measurement label of u:

- If $\lambda(u) \in \{X, XY\}$, then $(M\mathbf{a})_u = 0$ implies $u \notin \mathsf{Odd}(\mathscr{A})$, thus (F2) holds for u.
- If $\lambda(u) \in \{XZ, YZ, Z\}$, then $(M\mathbf{a})_u = 0$ implies $u \notin \mathscr{A}$, thus (F1) holds for u.
- If $\lambda(u) = Y$, then $(M\mathbf{a})_u = 0$ implies $u \notin \mathsf{Odd}[\mathscr{A}]$, thus (F3) holds for u.

In each case, the other two focusing conditions hold trivially for u. Hence $\mathscr{A}$ is focused, as desired. □

Several observations follow straightforwardly from Lemma 4.1.

Observation B.1. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph. Then:

- If $\mathscr{A} \subseteq \bar{I}$ is focused over $\mathscr{D} \subseteq \bar{O}$ and $\mathscr{D}'$ is a subset of $\mathscr{D}$, then $\mathscr{A}$ is focused over $\mathscr{D}'$.
- If $\mathscr{A}_1, \ldots, \mathscr{A}_n \subseteq \bar{I}$ are all focused over $\mathscr{D} \subseteq \bar{O}$, then $\Delta_{k=1..n} \mathscr{A}_k$ is focused over $\mathscr{D}$. This follows by linearity from noting that the indicator vector of $\Delta_{k=1..n} \mathscr{A}_k$ is the element-wise sum (modulo 2) of the indicator vectors of the individual sets.
- If $\mathscr{A}_1, \mathscr{A}_2 \subseteq \bar{I}$, then $\mathscr{A}_1 \Delta \mathscr{A}_2$ is focused over the singleton set $\{v\} \subseteq \bar{O}$ if and only if either both sets are focused over $\{v\}$, or neither is. (This corresponds to [31, Lemmas B.2 and B.3].)
- If a set $\mathscr{A} \subseteq \bar{I}$ is focused over sets $\mathscr{D}_1 \subseteq \bar{O}$ and $\mathscr{D}_2 \subseteq \bar{O}$, then $\mathscr{A}$ is also focused over $\mathscr{D}_1 \cup \mathscr{D}_2$.

Proof of Theorem 4.3. First, suppose Γ has a Pauli flow $(c, \prec)$ and there exists a set $K \subseteq \bar{I}$ which satisfies the three conditions. Let C be the correction matrix representing c, and let M, N be the flow-demand and order demand-matrices for Γ. Then by Theorem 2.13 we know that $MC = Id$ and that NC forms a directed acyclic graph.

The flow-demand and order-demand matrices for Γ' can be written in block-matrix form where rows and columns labelled by the original vertices come first and those labelled by z come last. Then, with M and N the respective matrices for Γ, we can write:

$$M_{\Gamma'} = \begin{pmatrix} M & \mathbf{x} \\ \mathbf{0} & 1 \end{pmatrix} \qquad \text{and} \qquad N_{\Gamma'} = \begin{pmatrix} N & \mathbf{z} \\ \mathbf{n} & 0 \end{pmatrix},$$

where

- $\mathbf{x}$ is the indicator vector among the non-outputs for the set of neighbours of u which are measured X, Y, or XY, i.e. $x_w = 1 \Leftrightarrow w \in S \cap \mathscr{X}$.
- $\mathbf{z}$ is the indicator vector among the non-outputs for the set of neighbours of u which are measured YZ or XZ, i.e. $z_w = 1 \Leftrightarrow w \in S \cap \mathscr{Z} \cap \mathscr{L}$.
- $\mathbf{n}$ is the indicator vector among the non-inputs for the set of all neighbours of u, i.e. $n_w = 1 \Leftrightarrow w \in S \setminus I$.

Let $\mathbf{k}$ be the indicator vector for the set K among the non-inputs and let $C'_{\Gamma'} = \left(\begin{smallmatrix} C & \mathbf{k} \\ \mathbf{0} & 1 \end{smallmatrix}\right)$, then

$$\begin{pmatrix} M & \mathbf{x} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} C & \mathbf{k} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} MC & M\mathbf{k} + \mathbf{x} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} Id & M\mathbf{k} + \mathbf{x} \\ \mathbf{0} & 1 \end{pmatrix}$$

Consider the v-th element of $M\mathbf{k} + \mathbf{x}$ for some $v \in \bar{O}$. If $v \in \bar{O} \setminus (S \cap \mathscr{X})$, then $x_v = 0$ and assumption 1 together with Lemma 4.1 implies that $(M\mathbf{k})_v = 0$. If $v \in S \cap \mathscr{X}$, then $x_v = 1$. By assumption 1, K would not be focused over $(\bar{O} \setminus (S \cap \mathscr{X})) \cup \{v\}$, so by Lemma 4.1 we must have $(M\mathbf{k})_v = 1$. Since $(M\mathbf{k})_v = x_v$ in either case, we find that $M\mathbf{k} + \mathbf{x} = \mathbf{0}$. Hence $M_{\Gamma'} C' = Id$, i.e. the new correction matrix satisfies the first of the algebraic Pauli flow properties.

For the product of order-demand and correction matrix, we have:

$$\begin{pmatrix} N & \mathbf{z} \\ \mathbf{n} & 0 \end{pmatrix} \begin{pmatrix} C & \mathbf{k} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} NC & N\mathbf{k} + \mathbf{z} \\ \mathbf{n}C & \mathbf{n} \cdot \mathbf{k} \end{pmatrix}$$

Now, each non-zero term in the sum $\mathbf{n} \cdot \mathbf{k} = \sum_{w \in \bar{I}} n_w k_w$ arises from a vertex $w \in S \cap K$, so $\mathbf{n} \cdot \mathbf{k} \equiv |S \cap K| \bmod 2$. Thus, assumption 2 implies that $\mathbf{n} \cdot \mathbf{k} = 0$.

Since $M_{\Gamma'} C_{\Gamma'} = Id$, by Lemma 4.1 applied to each column of $C_{\Gamma'}$ in turn, the matrix $C_{\Gamma'}$ represents a focused correction function on Γ'. Thus, by Observation 2.16, we have

- $(\mathbf{n}C)_w = 1$ for $w \in \bar{O}$ if and only if $z \in \mathsf{Odd}_{G'}(c(w))$, which is equivalent to $|c(w) \cap S| \equiv 1 \bmod 2$, and
- $(N\mathbf{k} + \mathbf{z})_v = 1$ for $v \in \bar{I}$ if and only if either $\lambda(v) = XY$ and $v \in K$, or $\lambda(v) \in \{XZ, YZ\}$ and $v \in \mathsf{Odd}_{G'}(K \cup \{z\})$. As $z \notin K$ and $z \notin \mathsf{Odd}_{G'}(K)$, we have:

$$\mathsf{Odd}_{G'}(K \cup \{z\}) = \mathsf{Odd}_{G'}(K \Delta \{z\}) = \mathsf{Odd}_{G'}(K) \Delta \mathsf{Odd}_{G'}(\{z\}) = \mathsf{Odd}_{G}(K) \Delta S.$$

  By focusing, XY-measured vertices don't appear in the odd neighbourhood of $K \cup \{z\}$ and other planar-measured vertices do not appear in K. Thus, equivalently $(N\mathbf{k} + \mathbf{z})_v = 1$ for $v \in \bar{I}$ if and only if $v \in \mathscr{L} \cap (K \cup (\mathsf{Odd}_G(K) \Delta S))$.

Since we know NC is a DAG and thus extends to a strict partial order $\prec$, Lemma 3.5 together with assumption 3 implies that the product $N_{\Gamma'}C_{\Gamma'}$ also extends to a strict partial order $\prec'$ and hence is a DAG. Then, as both $M_{\Gamma'}C_{\Gamma'}$ and $N_{\Gamma'}C_{\Gamma'}$ satisfy the properties of Theorem 2.13, the labelled open graph Γ' has Pauli flow.

For the other direction, assume Γ' has Pauli flow, i.e. there exists a matrix $C'_{\Gamma'}$ such that $M_{\Gamma'}C'_{\Gamma'} = Id$ and $N_{\Gamma'}C'_{\Gamma'}$ is a DAG. Express $C'_{\Gamma'}$ in block matrix form (separating out the row and column labelled by z, like before) as $\left(\begin{smallmatrix} C' & \mathbf{k}' \\ \mathbf{h} & d \end{smallmatrix}\right)$, then

$$Id = M_{\Gamma'}C'_{\Gamma'} = \begin{pmatrix} M & \mathbf{x} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} C' & \mathbf{k}' \\ \mathbf{h} & d \end{pmatrix} = \begin{pmatrix} MC' + \mathbf{x} \otimes \mathbf{h} & M\mathbf{k}' + \mathbf{x} \\ \mathbf{h} & d \end{pmatrix}$$

which implies $M\mathbf{k}' + \mathbf{x} = \mathbf{0}$, $\mathbf{h} = \mathbf{0}$ and $d = 1$. Then setting $\mathbf{h} = \mathbf{0}$ yields $MC' = Id$ by the properties of block matrices. Moreover, NC' is a DAG since a subgraph of a DAG must itself be a DAG, so we immediately have that Γ also has Pauli flow. It remains to show there exists a set K with the desired properties

Let $K' \subseteq \bar{I}$ be the set whose indicator vector is $\mathbf{k}'$. Consider first the property $M\mathbf{k}' + \mathbf{x} = \mathbf{0}$. Then for all $v \notin S \cap \mathscr{X}$, we have $x_v = 0$ and thus $(M\mathbf{k}')_v = 0$. By Lemma 4.1, this means K' is focused over $\bar{O} \setminus (S \cap \mathscr{X})$, i.e. the first part of property 1 holds for K'. Now, if $v \in S \cap \mathscr{X}$, then $x_v = 1$ and thus $(M\mathbf{k}')_v = 1$, so by Lemma 4.1, K' is not focused over any set that contains v. This establishes the second part of property 1.

Furthermore,

$$N_{\Gamma'}C_{\Gamma'} = \begin{pmatrix} N & \mathbf{z} \\ \mathbf{n} & 0 \end{pmatrix} \begin{pmatrix} C' & \mathbf{k}' \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} NC' & N\mathbf{k}' + \mathbf{z} \\ \mathbf{n}C' & \mathbf{n} \cdot \mathbf{k}' \end{pmatrix}$$

being a DAG implies that $\mathbf{n} \cdot \mathbf{k}' = 0$, which is equivalent to $|S \cap K'| \equiv 0 \bmod 2$: this is property 2. Finally, if $|c(w) \cap S| \equiv 1 \bmod 2$, then $u \in \mathsf{Odd}_{G'}(c(w))$ for some $w \in \bar{O}$, so $w \prec z$ by (P2). Similarly, by the definition of K' and the properties of a focused Pauli flow, if $v \in \mathscr{L}$ satisfies $v \in K' \cup (\mathsf{Odd}_G(K') \Delta S)$, then $v \in K'$ or $v \in \mathsf{Odd}_{G'}(K' \cup \{z\})$. Then by (P1) and (P2), $z \prec v$. Thus for all such v, w we have $w \prec v$ by transitivity of $\prec$, which implies $\neg(v \prec w \vee v = w)$, i.e. property 3. This completes the proof of the reverse direction. □

Corollary B.2. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph and let $S \subseteq O$, i.e. all neighbours of the new vertex are outputs. Define Γ' to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood S as in Definition 4.2. Then if Γ has Pauli flow, Γ' also has Pauli flow.

Proof. Suppose Γ has Pauli flow. Take $K = \emptyset$, then K is trivially focused over $\bar{O} \setminus S = \bar{O}$ and there is no larger set for which focusing is defined, so condition 1 holds. Moreover, condition 2 is trivially true. Finally, condition 3 is also satisfied since the set $\{v \in \mathscr{L} \mid v \in K \cup \mathsf{Odd}_{G'}(\{z\})\}$ is empty. Thus by Theorem 4.3, Γ' has Pauli flow. □

Corollary B.3. Let $\Gamma = (G, I, O, \lambda)$ be a labelled open graph and let $S \subseteq I$, i.e. all neighbours of the new vertex are inputs. Define Γ' to be the labelled open graph that results from inserting a new YZ-measured vertex z with neighbourhood S as in Definition 4.2. Then if Γ has Pauli flow, Γ' also has Pauli flow.

Proof. Assume Γ has focused Pauli flow $(c, \prec)$, let $K = \Delta_{s \in S} c(s)$ and recall Observation B.1. From the definition of a focused Pauli flow, for each $s \in S$, the correction set $c(s)$ is focused over $\bar{O} \setminus \{s\}$, and thus $c(s)$ is also focused over $\bar{O} \setminus S$. Therefore, K is focused over $\bar{O} \setminus S$. Moreover, for each $v \in S$ all but one of the components of K are focused over $\{v\}$ and thus K as a whole is not focused over $\{v\}$. This establishes condition 1. Condition 2 is immediate as $K \subseteq \bar{I}$ by the definition of correction sets. By the same reasoning, $\mathscr{W}_{pred} = \emptyset$ and therefore condition 3 holds. Thus by Theorem 4.3, Γ' has Pauli flow. □

Proof of Corollary 4.6. Given the focused Pauli flow $(c, \prec)$ on Γ, take $K := c(x)$. Firstly, K is focused over $\bar{O} \setminus \{x\}$ since c is a focused correction function, so the first part of condition 1 holds. Furthermore, $x \in \mathsf{Odd}_G(c(x))$, so K is not focused over any set containing x, which establishes the second part of condition 1

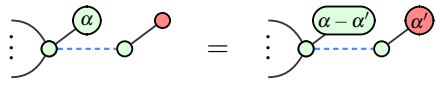
We have $x \notin c(x)$ by (P4), so $|S \cap K| = |\{x\} \cap c(x)| = 0$ and condition 2 holds.

Note that $\mathscr{W}_{pred} := \{w \in \bar{O} : |c(w) \cap \{x\}| \equiv 1 \bmod 2\} = \{w \in \bar{O} \mid x \in c(w)\}$, so $w \in \mathscr{W}_{pred}$ implies $w \prec x$ by (P1). Similarly $\mathscr{V}_{succ} := \mathscr{L} \cap (c(x) \cup (\mathsf{Odd}_G(c(x)) \Delta \{x\}))$ has the property that $v \in \mathscr{V}_{succ}$ implies $x \prec v$ by (P1) and (P2). Hence for all $w \in \mathscr{W}_{pred}$ and for all $v \in \mathscr{V}_{succ}$ we have $w \prec x \prec v$, which implies $\neg(v \prec w \vee v = w)$ since $\prec$ is strict. Therefore condition 3 holds.

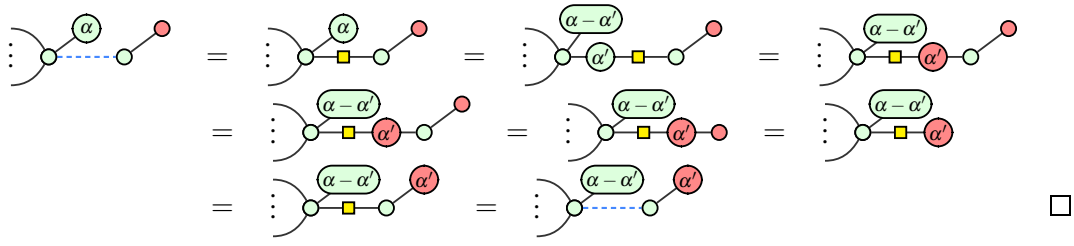
Thus, Γ' has the given Pauli flow by Theorem 4.3. □

C Proofs for Section 5

Observation C.1 (Phase shifting)**.** If a YZ-measured vertex z is connected to a single XY-measured vertex x, then shifting part from the phase of x to z is sound.



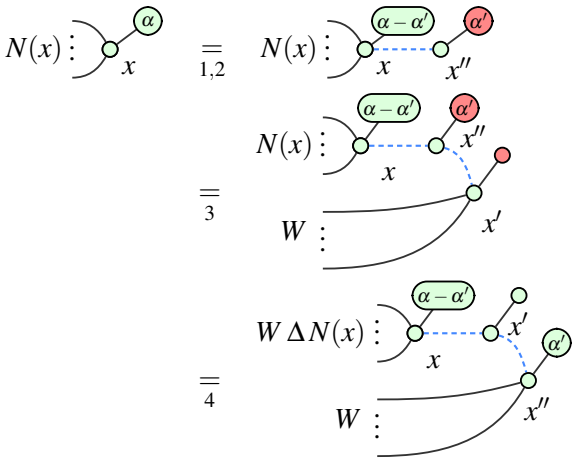
Proof.



□

Lemma C.2. The vertex splitting operation of Definition 5.4 can be effected via the following sequence of steps:

1. Insert a YZ-measured vertex x'' with neighbourhood $\{x\}$ according to Definition 4.2.
2. If desired, split measurement angles between x and x''.
3. Insert a YZ-measured vertex x' with neighbourhood $W \cup \{x''\}$ according to Definition 4.2.
4. Pivot on the edge $\{x', x''\}$.



The sequence can also be reversed.

Proof. Let $\Gamma_0 := \Gamma$ be the initial labelled open graph, with $G = (V,E)$, and denote by $\Gamma_k = (G_k, I, O, \lambda_k)$ the labelled open graph after step k in the above list. Then the labelled open graphs after each step have the following graphs and measurement labellings:

1. $G_1 = (V \cup \{x''\}, E \cup \{\{x,x''\}\})$ and λ_1 is the extension of λ that satisfies $\lambda_1(x'') = YZ$.
2. $\Gamma_2 = \Gamma_1$ as changing the measurement angles of planar-measured vertices does not affect the underlying labelled open graph.
3. $G_3 = (V_3, E_3)$ where $V_3 = V \cup \{x', x''\}$ and $E_3 = E \cup \{\{x,x''\}, \{x',x''\}\} \cup (\{x'\} \times W)$, with λ_3 being the extension of λ_1 that satisfies $\lambda_3(x') = YZ$. Note that all the new edges involve at least one vertex that did not appear in the original graph and hence are definitely not contained in E. Thus we may equivalently write $E_3 = E \,\Delta\, \{\{x,x''\}, \{x',x''\}\} \,\Delta\, (\{x'\} \times W)$.
4. By Lemma C.4 applied to singleton sets (so that the odd neighbourhood reduces to the usual neighbourhood), we have

$$N_{G_3 \wedge x'x''}(v) = \begin{cases} N_{G_3}(v) & \text{if } x', x'' \notin N_{G_3}[v] \\ N_{G_3}(v) \,\Delta\, N_{G_3}[x''] & \text{if } x' \in N_{G_3}[v] \wedge x'' \notin N_{G_3}[v] \\ N_{G_3}(v) \,\Delta\, N_{G_3}[x'] & \text{if } x' \notin N_{G_3}[v] \wedge x'' \in N_{G_3}[v] \\ N_{G_3}(v) \,\Delta\, N_{G_3}[x'] \,\Delta\, N_{G_3}[x''] & \text{if } x', x'' \in N_{G_3}[v] \end{cases}$$

or, by noting that $u \in N_{G_3}[v] \Leftrightarrow v \in N_{G_3}[u]$ and $N_{G_3}[x'] = \{x',x''\} \cup W$ and $N_{G_3}[x''] = \{x,x',x''\}$:

$$N_{G_4}(v) = N_{G_3 \wedge x'x''}(v) = \begin{cases} N_{G_3}(v) & \text{if } v \notin \{x,x',x''\} \cup W \\ N_{G_3}(v) \,\Delta\, \{x,x',x''\} & \text{if } v \in W \\ N_{G_3}(v) \,\Delta\, \{x',x''\} \,\Delta\, W & \text{if } v = x \\ N_{G_3}(v) \,\Delta\, \{x\} \,\Delta\, W & \text{if } v \in \{x',x''\} \end{cases} \tag{2}$$

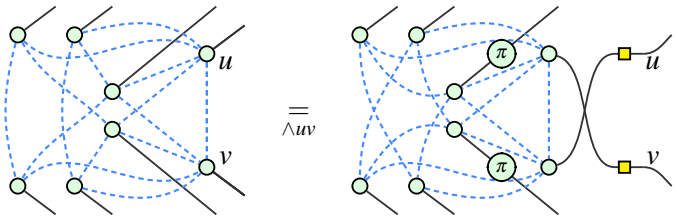
This implies we toggle edges between W and $\{x,x',x''\}$ as well as edges between x and $\{x',x''\}$. (Note that each toggled edge appears twice in (2), once for each endpoint.) Thus, $G_4 = (V_4, E_4)$ where $V_4 = V_3 = V \cup \{x',x''\}$ and:

$$\begin{aligned} E_4 &= E_3 \,\Delta\, (\{x,x',x''\} \times W) \,\Delta\, \{\{x,x'\}, \{x,x''\}\} \\ &= E \,\Delta\, \{\{x,x''\}, \{x',x''\}\} \,\Delta\, (\{x'\} \times W) \,\Delta\, (\{x,x',x''\} \times W) \,\Delta\, \{\{x,x'\}, \{x,x''\}\} \\ &= E \,\Delta\, \{\{x,x'\}, \{x',x''\}\} \,\Delta\, (\{x,x''\} \times W) \end{aligned}$$

Furthermore, λ_4 satisfies $\lambda_4(x') = \lambda_4(x'') = XY$ and $\lambda_4(v) = \lambda(v)$ for all other non-outputs. It is straightforward to check that Γ_4 is the same as the labelled open graph Γ' of Definition 5.4.

For the reverse sequence, note that pivoting is involutive and that YZ-measured vertices can be deleted without affecting the interpretation of the underlying diagram if their measurement angle is 0. □

Remark C.3. It is well-known [14] that in ZX-calculus, pivoting on edge $\{u,v\}$ reads:



What happens to the measurement effects when pivoting on an edge $\{u,v\}$ can be derived from this and rewrite rules. For instance, as  , if u and v were YZ-measured before the pivot, they will be XY-measured afterwards.

Lemma C.4 ([14, Lemma B.9])**.** Given a graph $G = (V,E)$, $A \subseteq V$, $u \in V$, and $v \in N_G(u)$,

$$\mathsf{Odd}_{G\wedge uv}(A) = \begin{cases} \mathsf{Odd}_G(A) & \text{if } u,v \notin \mathsf{Odd}_G[A] \\ \mathsf{Odd}_G(A)\,\Delta\, N_G[v] & \text{if } u \in \mathsf{Odd}_G[A], v \notin \mathsf{Odd}_G[A] \\ \mathsf{Odd}_G(A)\,\Delta\, N_G[u] & \text{if } u \notin \mathsf{Odd}_G[A], v \in \mathsf{Odd}_G[A] \\ \mathsf{Odd}_G(A)\,\Delta\, N_G[u]\,\Delta\, N_G[v] & \text{if } u,v \in \mathsf{Odd}_G[A]. \end{cases}$$

Proof of Lemma 5.3. First, note (e.g. via Remark C.3) that $u,v \in N_G[u]$, that symmetrically $u,v \in N_G[v]$, and that

$$\mathsf{Odd}_G[A\,\Delta\,\{u\}] = \mathsf{Odd}_G[A]\,\Delta\, N_G[u].$$

Thus, for any set $A \subseteq V$, we have $u \in \mathsf{Odd}_G[A\,\Delta\,\{u\}] \Leftrightarrow u \notin \mathsf{Odd}_G[A]$ and similarly $v \in \mathsf{Odd}_G[A\,\Delta\,\{u\}] \Leftrightarrow v \notin \mathsf{Odd}_G[A]$.

By Lemma C.4, in the four cases (depending on which of u,v are in $\mathsf{Odd}_G[c(w)]$), we have:

- If $u,v \notin \mathsf{Odd}_G[c(w)]$, then

$$\mathsf{Odd}_{G\wedge uv}(c'(w)) = \mathsf{Odd}_{G\wedge uv}(c(w)) = \mathsf{Odd}_G(c(w))$$

- If $u \in \mathsf{Odd}_G[c(w)]$ and $v \notin \mathsf{Odd}_G[c(w)]$, then $u \notin \mathsf{Odd}_G[c(w)\,\Delta\,\{u\}]$ and $v \in \mathsf{Odd}_G[c(w)\,\Delta\,\{u\}]$; thus

$$\begin{aligned} \mathsf{Odd}_{G\wedge uv}(c'(w)) &= \mathsf{Odd}_{G\wedge uv}(c(w)\,\Delta\,\{u\}) \\ &= \mathsf{Odd}_G(c(w)\,\Delta\,\{u\})\,\Delta\, N_G[u] \\ &= \mathsf{Odd}_G(c(w))\,\Delta\, N_G(u)\,\Delta\, N_G[u] \\ &= \mathsf{Odd}_G(c(w))\,\Delta\,\{u\} \end{aligned}$$

- If $u \notin \mathsf{Odd}_G[c(w)], v \in \mathsf{Odd}_G[c(w)]$, then symmetrically to the previous case,

$$\mathsf{Odd}_{G\wedge uv}(c'(w)) = \mathsf{Odd}_{G\wedge uv}(c(w)\,\Delta\,\{v\}) = \mathsf{Odd}_G(c(w))\,\Delta\,\{v\}$$

Case	$c'(w)$	$\mathsf{Odd}_{G\wedge uv}(c'(w))$
$u,v \notin \mathsf{Odd}_G[c(w)]$	$c(w)$	$\mathsf{Odd}_G(c(w))$
$u \in \mathsf{Odd}_G[c(w)], v \notin \mathsf{Odd}_G[c(w)]$	$c(w)\,\Delta\,\{u\}$	$\mathsf{Odd}_G(c(w))\,\Delta\,\{u\}$
$u \notin \mathsf{Odd}_G[c(w)], v \in \mathsf{Odd}_G[c(w)]$	$c(w)\,\Delta\,\{v\}$	$\mathsf{Odd}_G(c(w))\,\Delta\,\{v\}$
$u,v \in \mathsf{Odd}_G[c(w)]$	$c(w)\,\Delta\,\{u,v\}$	$\mathsf{Odd}_G(c(w))\,\Delta\,\{u,v\}$

Table 1: Correction sets and odd neighbourhoods after pivoting in Lemma 5.3, depending on the relationship of u and v to the original closed odd neighbourhood.

- If $u,v \in \mathsf{Odd}_G[c(w)]$, then $u,v \in \mathsf{Odd}_G[c(w)\,\Delta\,\{u,v\}]$ and thus

$$\begin{aligned}\mathsf{Odd}_{G\wedge uv}\left(c'(w)\right) &= \mathsf{Odd}_{G\wedge uv}(c(w)\,\Delta\,\{u,v\})\\ &= \mathsf{Odd}_G(c(w)\,\Delta\,\{u,v\})\,\Delta N_G[u]\,\Delta N_G[v]\\ &= \mathsf{Odd}_G(c(w))\,\Delta N_G(u)\,\Delta N_G(v)\,\Delta N_G[u]\,\Delta N_G[v]\\ &= \mathsf{Odd}_G(c(w))\,\Delta\,\{u,v\}\end{aligned}$$

These results are summarised in Table 1. They straightforwardly imply $\mathsf{Odd}_{G\wedge uv}[c'(w)] = \mathsf{Odd}_G[c(w)]$ for all $w \in \bar{O}$: in each row, the changes to correction set and odd neighbourhood are the same, so they cancel out when computing the closed odd neighbourhood.

Note that the only changes to correction sets and odd neighbourhoods are symmetric differences with some subset of $\{u,v\}$, so Pauli flow and (where applicable) focusing conditions involving only vertices other than u,v remain satisfied.

Additionally, whether a symmetric difference is taken with u (and what effect this may have) is independent of what is done regarding v. By symmetry, it thus suffices to consider only u. First, consider the correction set for u itself.

- If $\lambda(u) = XY$, then $u \notin c(u) \wedge u \in \mathsf{Odd}_G(c(u))$ by (P4), so $u \in \mathsf{Odd}_G[c(u)]$. Thus we take symmetric differences with u, which means $u \in c'(u) \wedge u \notin \mathsf{Odd}_{G\wedge uv}(c'(u))$. Yet note that $\lambda'(u) = YZ$ and the above is exactly what is needed to satisfy (P6).
- The case $\lambda(u) = YZ$ is symmetric to $\lambda(u) = XY$.
- If $\lambda(u) = XZ$, then $u \in c(u) \wedge u \in \mathsf{Odd}_G(c(u))$, so $u \notin \mathsf{Odd}_G[c(u)]$. Thus the membership of u in those sets does not change and (P5) remains satisfied.
- If $\lambda(u) = X$, then $u \in \mathsf{Odd}_G(c(u))$ by (P7), and $\lambda'(u) = Z$. There are two subcases:
  - If furthermore $u \in c(u)$, then $u \notin \mathsf{Odd}_G[c(u)]$ and nothing changes, meaning $u \in c'(u)$ and $u \in \mathsf{Odd}_{G\wedge uv}(c'(u))$. Hence (P9) is satisfied.
  - If instead $u \notin c(u)$, then $u \in \mathsf{Odd}_G[c(u)]$. Thus symmetric differences with u are taken and we have $u \in c'(u)$ and $u \notin \mathsf{Odd}_{G\wedge uv}(c(u'))$. Again, (P9) is satisfied.
- The case $\lambda(u) = Z$ is symmetric to $\lambda(u) = X$.
- If $\lambda(u) = Y$, then there are two possibilities but in either case $u \in \mathsf{Odd}_G[c(u)]$ by (P8). Thus symmetric differences are taken, and afterwards $u \in \mathsf{Odd}_{G\wedge uv}[c'(u)]$ again. The measurement label remains $\lambda'(u) = Y$ and (P8) remains satisfied.

Now suppose $w \in \bar{O} \setminus \{u\}$. Note that by the first part of the proof,

$$u \in c'(w) \Leftrightarrow u \in \mathsf{Odd}_G(c(w)) \quad \text{and} \quad u \in \mathsf{Odd}_{G\wedge uv}\left(c'(w)\right) \Leftrightarrow u \in c(w). \tag{3}$$

If $u \notin c(w)$ and $u \notin \mathsf{Odd}_G(c(w))$, then analogous properties will hold after pivoting, so there is nothing to prove. We will therefore only consider cases where u is in the correction set, the odd neighbourhood, or both.

- If $\lambda(u) = XY$, and u appears in $c(w)$ and/or $\mathsf{Odd}_G(c(w))$, then $w \prec u$ by (P1) and (P2). So, whenever u appears in $c'(w)$ and/or $\mathsf{Odd}_{G\wedge uv}(c'(w))$, we have $w \prec u$ and thus (P1) and (P2) remain satisfied for $c'(w)$.
- The case $\lambda(u) = YZ$ is analogous to $\lambda(u) = XY$.
- The case $\lambda(u) = XZ$ is also analogous to $\lambda(u) = XY$.
- If $\lambda(u) = X$, there are two subcases.
  - Suppose $u \in \mathsf{Odd}_G(c(w))$, then $w \prec u$. Then $u \in c'(w)$ but (P1) remains satisfied after the change of label to $\lambda'(u) = Z$ since we still have $w \prec u$.
  - Suppose $u \in c(w)$ and $u \notin \mathsf{Odd}_G(c(w))$. In this case, nothing is implied about the order of u and w. After the pivot, we have $\lambda'(u) = Z$, $u \notin c'(w)$ and $u \in \mathsf{Odd}_{G\wedge uv}(c'(w))$. Thus, (P2) is satisfied.
- The case $\lambda(u) = Z$ is symmetric to $\lambda(u) = X$.
- If $\lambda(u) = Y$, there are again two cases.
  - If $u \notin \mathsf{Odd}_G[c(w)]$, then we know nothing about the order of u and w because of (P3), but nothing changes regarding the correction set or the label of u, so (P3) remains satisfied.
  - If $u \in \mathsf{Odd}_G[c(w)]$, then by (P3), we have $w \prec u$. In this case u moves from the correction set to the odd neighbourhood, or conversely, but (P3) remains satisfied.

The arguments regarding v are independent of and analogous to the ones for u, so this completes the proof that $(p', \prec)$ is a Pauli flow.

For focusing, it suffices to consider the correction sets of vertices other than u (as focusing does not talk about the relationship between a vertex and its own correction set or the latter's odd neighbourhood). So suppose $w \in \bar{O} \setminus \{u\}$.

- If $\lambda(u) = Y$ then $(p, \prec)$ being focused implies $u \notin \mathsf{Odd}_G[c(w)]$ via (F3). But then nothing changes during the pivot, not even the measurement label of u, so (F3) remains satisfied.
- If $\lambda(u) = XY$ or $\lambda(u) = X$, then $(p, \prec)$ being focused implies $u \notin \mathsf{Odd}_G(c(w))$ via (F2). After the pivot, $\lambda'(u) \in \{YZ, Z\}$ and $u \notin c'(w)$ by (3), so (F1) is satisfied.
- If $\lambda(u) = YZ$ or $\lambda(u) = Z$, the argument is symmetric to the previous case.
- The case $\lambda(u) = XZ$ is excluded.

Thus $(p, \prec)$ being focused implies that $(p', \prec)$ is focused, assuming the vertices incident on the pivot edge are not XZ-measured.

Finally, since the gflow conditions are a subset of the Pauli flow conditions and pivoting maps planar measurements to planar measurements, if $(p, \prec)$ is a gflow, then $(p', \prec')$ is also a gflow. □

Remark C.5. In Lemma 5.3, focusing may break if one of the vertices incident on the pivot edge is XZ-measured, say $\lambda(u) = XZ$: assume $(p, \prec)$ is focused, then we know $u \notin c(w)$ for any $w \neq u$ but we may have $u \in \mathsf{Odd}_G(c(w))$. In the latter case, we would have $u \in c'(w)$ after the pivot but $\lambda(u) = XZ$ still, so the flow is no longer focused. Of course it can be re-focused by replacing $c'(w)$ with $c'(w)\,\Delta\,c'(u)$ for all w such that $u \in \mathsf{Odd}_G(c(w))$, but the straightforward modification of Lemma 5.3 is no longer sufficient.

Proof of Theorem 5.5. By Lemma C.2, the vertex splitting operation of Definition 5.4 can be expressed as a sequence of two YZ-insertions and a pivot, as well as an operation on measurement angles that does not affect the labelled open graphs. As $|I| = |O|$, the focused Pauli flow on each labelled open graph is unique. Using the same numbering as in the above lemma, Γ_1 has Pauli flow if and only if Γ has Pauli flow by Corollary 4.6. Similarly, by Lemma 5.3 and the property that pivoting is involutive, Γ_4 has Pauli flow if and only if Γ_3 has Pauli flow. In each case, there are no conditions on the flows. Thus it remains

only to show that step 3 – the second YZ-insertion – preserves the existence of Pauli flow under the given conditions.

First, suppose Γ has focused Pauli flow $(c, \prec)$ where without loss of generality $\prec$ is the induced partial order. Then, by Corollary 4.6, the unique focused Pauli flow on $\Gamma_2 = \Gamma_1$ satisfies

$$\prec_2 = \prec \cup \{(w, x'') \mid w \prec x\} \cup \{(x'', v) \mid x \prec v\} \quad \text{and} \quad c_2(v) = \begin{cases} c(x) \cup \{x''\} & \text{if } v = x'' \\ c(v) & \text{otherwise.} \end{cases}$$

Furthermore, by Theorem 4.3, Γ_3 has focused Pauli flow if and only if there exists $K \subseteq V_2 = V \cup \{x''\}$ such that three conditions hold with respect to the neighbourhood $S = W \cup \{x''\}$ of the new vertex. We now show those conditions follow from (V1)–(V4).

- The only difference between Γ_2 and Γ is the insertion of the YZ-measured vertex x'' with a single edge to x. Thus K being focused over $\bar{O} \setminus (W \cap \mathscr{X})$ in Γ by (V1) implies K being focused over $(V \setminus O) \setminus (W \cap \mathscr{X}_2)$ in Γ_2.
  Note that $x'' \notin K$ by assumption, so K is focused over $\{x''\}$. Therefore by Observation B.1, K is focused over $(V \cup \{x''\} \setminus O) \setminus (W \cap \mathscr{X}_2)$ in Γ_2. We can also add x'' to W since $x'' \notin \mathscr{X}_2$. Hence K is focused over $\bar{O}_2 \setminus ((W \cup \{x''\}) \cap \mathscr{X}_2)$, which is the first part of condition 1. The second part follows similarly from (V1).
- We have $|W \cap K| \equiv 0 \bmod 2$ by (V1) and $x'' \notin K$. Thus $|(W \cup \{x''\}) \cap K| \equiv 0 \bmod 2$, which is condition 2.
- Denote by $\mathscr{W}^{(2)}_{pred}$ and $\mathscr{V}^{(2)}_{succ}$ the sets from condition 3, i.e.

$$\mathscr{W}^{(2)}_{pred} = \left\{w \in \bar{O}_2 : \left|c(w) \cap (W \cup \{x''\})\right| \equiv 1 \bmod 2\right\} \tag{4}$$

$$\mathscr{V}^{(2)}_{succ} = \mathscr{L}_2 \cap (K \cup (\mathsf{Odd}_{G_2}(K) \,\Delta\, (W \cup \{x''\}))) \tag{5}$$

  Note that x'' does not appear in any correction set other than its own because of focusing, so the first definition reduces to

$$\mathscr{W}^{(2)}_{pred} = \begin{cases} \mathscr{W}_{pred} & \text{if } |c(x) \cap W| \equiv 1 \bmod 2 \\ \mathscr{W}_{pred} \cup \{x''\} & \text{if } |c(x) \cap W| \equiv 0 \bmod 2 \end{cases} \tag{6}$$

  Similarly, since $x'' \notin K$, the set $\mathsf{Odd}_{G_2}(K)$ is equal to $\mathsf{Odd}_G(K)$ if $x \notin K$ and to $\mathsf{Odd}_G(K) \cup \{x''\}$ if $x \in K$. Thus:

$$\mathscr{V}^{(2)}_{succ} = \begin{cases} \mathscr{V}_{succ} & \text{if } x \in K \\ \mathscr{V}_{succ} \cup \{x''\} & \text{if } x \notin K \end{cases} \tag{7}$$

  We distinguish cases depending on the parity of $|c(x) \cap W|$ and on whether x is in K.
  - Suppose $|c(x) \cap W| \equiv 1 \bmod 2$ and $x \in K$. This is forbidden by (V4) so can be ignored.
  - Suppose $|c(x) \cap W| \equiv 0 \bmod 2$ and $x \in K$. Take $w \in \mathscr{W}^{(2)}_{pred}$.
    If $w = x''$, then for all $u \in V$ such that $u \prec_2 x''$ we also have $u \prec x$ by the definition of $\prec_2$, and thus by (V3) $u \notin \mathscr{V}^{(2)}_{succ}$. Moreover, $x'' \notin \mathscr{V}^{(2)}_{succ}$ since $x \in K$. Thus $\neg(v \prec_2 w \vee v = w)$ for all $v \in \mathscr{V}^{(2)}_{succ}$.
    If $w \neq x''$, the result follows from (V2), noting that for all $u, v \in V$ we have $u \prec_2 v$ if and only if $u \prec v$.
  - Suppose $|c(x) \cap W| \equiv 1 \bmod 2$ and $x \notin K$. This is again forbidden by (V4).

– Suppose $|c(x) \cap W| \equiv 0 \bmod 2$ and $x \notin K$. Take $v \in \mathscr{V}^{(2)}_{succ}$.
If $v = x''$, then for all $u \in V$ such that $x'' \prec_2 u$ we also have $x \prec u$ by the definition of $\prec_2$, and thus by (V3) $u \notin \mathscr{W}^{(2)}_{pred}$. Moreover, $x'' \notin \mathscr{W}^{(2)}_{pred}$ since $|c(x) \cap W| \equiv 0 \bmod 2$. Thus $\neg(v \prec_2 w \vee v = w)$ for all $w \in \mathscr{W}^{(2)}_{pred}$.
If $v \neq x''$, the result again follows from (V2) and noting that for all $u, w \in V$ we have $u \prec_2 w$ if and only if $u \prec w$.

Thus condition 3 follows from (V2)–(V4).

We have shown that Γ' has Pauli flow if Γ has a focused Pauli flow and we can find a set K such that (V1)–(V4) hold.

For the reverse direction, assume Γ' has focused Pauli flow $(c_4, \prec_4)$, then by Lemma 5.3, Γ_3 has focused Pauli flow $(c_3, \prec_3)$, where $\prec_3 = \prec_4$ and for all $w \in V_4 = V \cup \{x', x''\}$:

$$c_3(w) := \begin{cases} c_4(w) & \text{if } x', x'' \notin \mathsf{Odd}_{G_4}[c_4(w)] \\ c_4(w) \Delta \{x'\} & \text{if } x' \in \mathsf{Odd}_{G_4}[c_4(w)], x'' \notin \mathsf{Odd}_{G_4}[c_4(w)] \\ c_4(w) \Delta \{x''\} & \text{if } x' \notin \mathsf{Odd}_{G_4}[c_4(w)], x'' \in \mathsf{Odd}_{G_4}[c_4(w)] \\ c_4(w) \Delta \{x', x''\} & \text{if } x', x'' \in \mathsf{Odd}_{G_4}[c_4(w)] \end{cases}$$

Take $K' = c_3(x') \setminus \{x'\}$, i.e. $K' = c_4(x') \setminus \{x''\}$ if $x'' \in c_4(x')$ and $K' = c_4(x')$ otherwise. By Theorem 4.3, we know that Γ_2 (and thus iteratively Γ) has Pauli flow, and conditions 1–3 hold for Γ_2.

1. K' being focused over $\bar{O}_2 \setminus ((W \cup \{x''\}) \cap \mathscr{X}_2)$ and no larger set implies K' is focused over $(\bar{O} \cup \{x''\}) \setminus (W \cap \mathscr{X})$ since $\lambda_2(x'') = YZ$. Now $x'' \notin K'$ by definition, so (F1)–(F3) hold with respect to x'' and we may limit attention to the remaining vertices. This yields the condition of K' being focused over $\bar{O} \setminus (W \cap \mathscr{X})$ and no larger set, which is exactly the definition of K. Moreover, $|(W \cup \{x''\}) \cap K'| \equiv 0 \bmod 2$ implies $|W \cap K'| \equiv 0 \bmod 2$ since $x'' \notin K'$ by the definition above. Thus (V1) holds.
2. Let $\mathscr{W}^{(2)}_{pred}$ and $\mathscr{V}^{(2)}_{succ}$ as defined in (4) and(5). Then all $w \in \mathscr{W}^{(2)}_{pred}$ and all $v \in \mathscr{V}^{(2)}_{succ}$ satisfying $\neg(v \prec_2 w \vee v = w)$ immediately implies (V2) via (6) and (7), as well as noting that for vertices $u, u' \in V$, we have $u \prec_2 u' \Leftrightarrow u \prec u'$.
Then from (6), we see that $x'' \in \mathscr{W}^{(2)}_{pred}$ if and only if $|c(x) \cap W| \equiv 0 \bmod 2$ and $x \in \mathscr{W}^{(2)}_{pred}$ if and only if $|c(x) \cap W| \equiv 1 \bmod 2$. Similarly, from (7), we deduce that $x'' \in \mathscr{V}^{(2)}_{succ}$ if and only if $x \notin K$ and $x \in \mathscr{V}^{(2)}_{succ}$ if and only if $x \in K$. Thus the requirement that $\mathscr{W}^{(2)}_{pred}$ and $\mathscr{V}^{(2)}_{succ}$ do not share elements implies either $|c(x) \cap W| \equiv 0 \bmod 2$ and $x \in K$, or $|c(x) \cap W| \equiv 1 \bmod 2$ and $x \notin K$: i.e. (V4) holds. It also means that exactly one of x, x'' is in $\mathscr{W}^{(2)}_{pred}$ and the other is in $\mathscr{V}^{(2)}_{succ}$.
Finally, by the uniqueness of focused Pauli flow on labelled open graphs with $|I| = |O|$ and by Corollary 4.6, we know that x and x'' must have the same partial order relationships in $\prec_2$. Suppose $x \in \mathscr{W}^{(2)}_{pred}$ and $x'' \in \mathscr{V}^{(2)}_{succ}$ (the other case is analogous). Then for all $v \in \mathscr{V}^{(2)}_{succ}$, condition 3 implies $\neg(v \prec_2 x)$. So if $u \in V$ satisfies $u \prec x$ and thus $u \prec_2 x$, then $u \notin \mathscr{V}^{(2)}_{succ}$. Similarly, for all $w \in \mathscr{W}^{(2)}_{pred}$, condition 3 implies $\neg(v \prec_2 x'')$. So if $u \in V$ satisfies $x \prec u$ and thus $x'' \prec_2 u$, then $u \notin \mathscr{W}^{(2)}_{pred}$. Thus (V3) holds.

This establishes all six conditions hold if Γ' has Pauli flow. □

Proof of Corollary 5.6. Suppose Γ has a focused Pauli flow $(c, \prec)$ which satisfies the two conditions and where without loss of generality $\prec$ is the induced partial order. Take $K := c(b)$ in Theorem 5.5. Then

focusing of $(c,\prec)$ implies K is focused over $\bar{O}\setminus\{b\}$ and (P4) implies K is not focused over $\{b\}$ as well as $|\{b\}\cap c(b)| = 0 \bmod 2 \Leftrightarrow b \notin c(b)$. Together these imply (V1). We also have

$$\mathscr{W}_{pred} := \{w \in \bar{O} : |c(w)\cap\{b\}| \equiv 1 \bmod 2\} = \{w \in \bar{O} \mid b \in c(w)\} = \{w \in \bar{O} \mid w \lhd_c b\}$$

and

$$\mathscr{V}_{succ} := \mathscr{L} \cap (c(b) \cup (\mathsf{Odd}_G(c(b)) \,\Delta\, \{b\})) = \{v \in \bar{O} \mid b \lhd_c v\}.$$

Thus (V2) follows directly from $\prec$ being the induced partial order (cf. Definition 2.14). Next, (V3) reduces to $a \prec u \implies \neg(u \lhd_c b)$ and $u \prec a \implies \neg(b \lhd_c u)$ for all $u \in V$, which follows from the second bullet point. Finally, (V4) becomes $a \in c(b) \Leftrightarrow b \notin c(a)$, which is equivalent to the first bullet point. Thus all the conditions of Theorem 5.5 are satisfied and the operation preserves the existence of Pauli flow.

For the other direction, suppose Γ' has Pauli flow. Then by Theorem 5.5, Γ has a focused Pauli flow $(c,\prec)$ such that conditions (V1)–(V4) hold. We may assume without loss of generality that $\prec$ is the induced partial order (cf. Section 2.3), since removing relationships from the partial order will not break any of the four conditions. Then, from the above, (V4) implies the first bullet point. Now suppose for a contradiction $u \in V$ satisfies $b \prec u \prec a$. Then there exists a sequence $u_1,\ldots,u_n$ such that $b \lhd_c u_1 \lhd_c \ldots \lhd_c u_n \lhd_c u$ since $\prec$ is the induced partial order. Thus there exists $u_1 \in \mathscr{V}_{succ}$ such that $u_1 \prec a$, which contradicts condition (V3) so this cannot happen and we have $u \prec a \implies \neg(b \prec u)$. A similar argument works for the case $a \prec u \prec b$. Thus the second bullet point holds. □

ZX-calculus is Complete for Finite-Dimensional Hilbert Spaces

Boldizsár Poór[1,2] Razin A. Shaikh[1,2] Quanlong Wang[1]

[1]Quantinuum, 17 Beaumont Street, Oxford, OX1 2NA, United Kingdom
[2]University of Oxford, United Kingdom

The ZX-calculus is a graphical language for reasoning about quantum computing and quantum information theory. As a complete graphical language, it incorporates a set of axioms rich enough to derive any equation of the underlying formalism. While completeness of the ZX-calculus has been established for qubits and the Clifford fragment of prime-dimensional qudits, universal completeness beyond two-level systems has remained unproven until now. In this paper, we present a proof establishing the completeness of finite-dimensional ZX-calculus, incorporating only the mixed-dimensional Z-spider and the qudit X-spider as generators. Our approach builds on the completeness of another graphical language, the finite-dimensional ZW-calculus, with direct translations between these two calculi. By proving its completeness, we lay a solid foundation for the ZX-calculus as a versatile tool not only for quantum computation but also for various fields within finite-dimensional quantum theory.

1 Introduction

The ZX-calculus [14, 15] is a graphical language for reasoning about quantum computation and quantum information theory. Thanks to its model-independent representation of quantum computation and set of intuitive graphical rewrite rules, the ZX-calculus has found applications across various domains of quantum technologies [58]. These domains include quantum error correction [24, 39, 36, 56, 23, 50], quantum circuit optimization [30, 6, 40, 60], measurement-based quantum computing [31, 4, 43], fusion-based quantum computing [7, 45, 33], compilation [55], classical simulation [13, 10], and education [19, 18, 32].

While the ZX-calculus has primarily been studied for reasoning about qubit quantum computation, various extensions or modifications allow it to address different aspects of quantum computation and theory. From the perspective of the ZX-calculus, there are two approaches to modifying the language: changing the set of generators and/or giving them an alternative interpretation. Languages with a different set of generators include the ZW-calculus [20, 35], ZH-calculus [3, 42], and ZXW-calculus [54, 65]. Alternatively, a different interpretation may involve extending the language to higher-dimensional systems such as qutrits [62, 57, 59], quopits [8, 47, 21, 9], qudits [51, 48, 22], or the finite-dimensional setting [63, 64, 28].

While languages like the ZXW- and ZW-calculus enable complete reasoning for both qudits and finite-dimensional Hilbert spaces, defining the ZX-calculus with the same generality can open avenues for interesting applications, as different generator sets exhibit distinct strengths and weaknesses. For instance, the ZW-calculus [17] provides valuable insights for studying multi-partite entanglement [17, 35] and linear optical quantum computing [26, 27]. However, it is less effective for understanding circuit-based quantum protocols. Similarly, while the ZXW-calculus is a powerful analytical tool capable of

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 127–158, doi:10.4204/EPTCS.426.5

expressing Hamiltonians [54, 27], performing differentiation and integration of ZX-diagrams [65], classical simulation [41], and interactions in quantum field theory [53], extracting circuits expressed by these methods remains a hard problem [25]. By contrast, with the ZX-calculus we can keep diagrams easily extractable for quantum circuit optimization [30]. Therefore, the finite-dimensional ZX-calculus has the potential to yield results directly applicable to quantum circuits.

When designing graphical languages, there are three crucial properties that ensure the language can be effectively used. These properties are (1) *soundness*, which ensures that the rules of the calculus are correct, (2) *universality*, ensuring that the language can express any element of the underlying formalism, and (3) *completeness*, enabling the derivation of any equality within the underlying formalism. Of these, proving completeness poses the most difficulty; nevertheless, it ensures that the language can be effectively used to graphically derive equalities and proofs.

Indeed, there is a rich history of results investigating the completeness and incompleteness of various calculi. While the qubit ZX-calculus was first formulated in 2007 [14], completeness for qubit quantum computing was not proved until 2017 [44, 38]. This unfolded in several stages, progressively increasing the fragment for which completeness had been proved. The first completeness proof was for the Clifford fragment [1], followed by a proof of incompleteness for the universal fragment [52]. Moving forward, completeness for single qubit Clifford+T quantum mechanics was established in [2], and the completeness of the ZW-calculus was presented in [34] for qubits with integer coefficients and for the universal fragment in [35]. Further study of the ZX-calculus revealed the necessity of the 'supplementarity' rule [46]. Transferring the completeness proof of the ZW-calculus [34, 35] led to a surge of completeness results. First, for the Clifford+T fragment in [37], and then for the universal fragment in [44, 38]. Further improvements to the qubit ZX-calculus focused on minimizing its set of axioms, first in [5] and then in [61]. This process was highly non-trivial, but now each rule is concise and intuitive.

Further to qubit calculi, the completeness of qudit calculi has also been extensively studied over the years. The first such completeness result was that of the stabilizer fragment for qutrit ZX-calculus [62]. The stabilizer fragment of the ZX-calculus for all odd prime dimensions was shown to be complete in [8], and its rule-set has since been further reduced in [47]. The first universal completeness of a graphical language beyond qubits was established in [48] for the arbitrary finite dimensions of ZXW-calculus. Recently, the completeness of the qudit ZW-calculus has also been demonstrated in [28], with a significantly reduced set of axioms. Alongside the qudit ZW-calculus, the paper also introduces and proves the completeness of the finite-dimensional ZW-calculus [28]. This result came shortly after the establishment of the qufinite ZXW-calculus and its completeness for finite-dimensional Hilbert spaces [64]. However, the universal completeness of any higher-dimensional ZX-calculus has remained unproven.

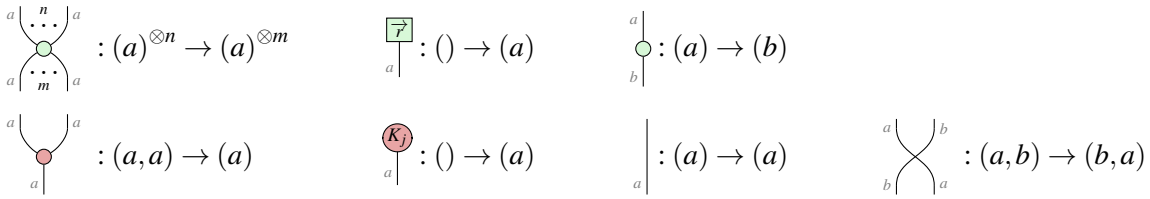
In this paper, we define the finite-dimensional ZX-calculus and prove its completeness. We begin by defining the generators of the calculus as the mixed-dimensional Z-spider [64] and the qudit X-spider, and then presenting its axioms. Next, we recapitulate the axioms and definition of the finite-dimensional ZW-calculus [28]. We then move on to proving the main result of the paper — the completeness of the finite-dimensional ZX-calculus. It is achieved by translating the generators of our language to the finite-dimensional ZW-calculus [28] and proving the invertibility of this translation. Similar techniques of proving completeness have been employed before, first in [37], and later in [44, 38]. By establishing completeness, we lay a solid foundation for the ZX-calculus as a versatile tool not only for quantum computation but also for various fields within finite-dimensional quantum theory.

2 Finite-Dimensional ZX-calculus

In this section, we define our calculus starting with the generators, their interpretation, and lastly, the axiomatization that we later show to be complete. Since any 1-dimensional diagram is just the empty diagram, this paper only considers dimensions strictly bigger than 1.

2.1 Generators

We define the symmetric monoidal category $\mathbf{ZX_f}$ with objects as lists of dimensions $(d_i)_{i=1}^n$ where $d_i \in \mathbb{N}$ for all $0 < i \leq n$, such that $d_i \geq 2$, and morphisms generated by the following diagrams, for any $a,b,n,m \in \mathbb{N}$, $0 \leq j < a$, and $\overrightarrow{r} \in \mathbb{C}^a$ such that $r_0 = 1$:

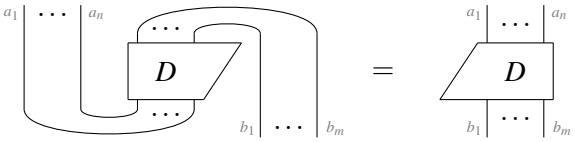


Diagrams are to be read top-to-bottom, as implied by the interpretation below. Diagrams can be composed in two ways: sequentially, by connecting input and output wires, and in parallel, by placing them side-by-side.

We extend our language with the standard *qudit Z-spider* and the *mixed-dimensional Z-spider* [64], given by the following compositions, respectively:

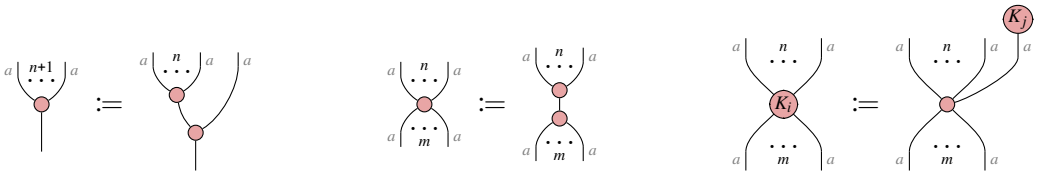
where $a = \min_{i=0}^{n+m} a_i$ is the minimal dimension. Furthermore, the phase-free Z-spider with arbitrary legs can express both the cap and cup as follows:

Caps and cups then allow us to construct the transposition of any diagram:



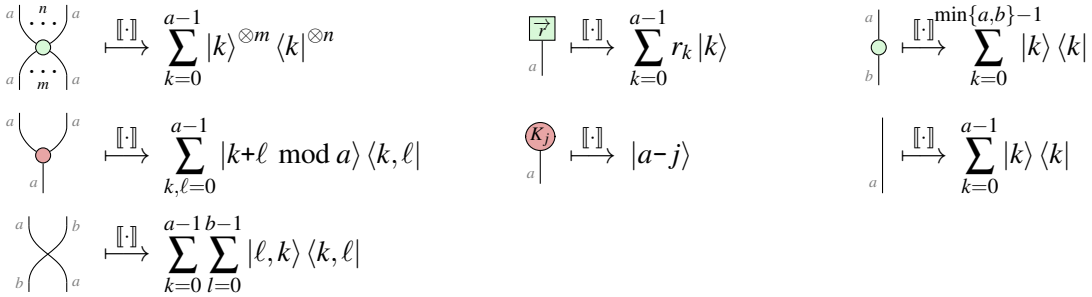
In particular, we obtain the transposition of all the generators.

With these diagrams, we can now define the *X-spider* inductively, for any $m,n \in \mathbb{N}$:

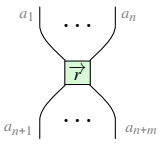


2.2 Interpretation

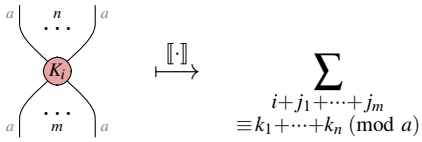
The standard interpretation of a diagram in $\mathbf{ZX_f}$ is a symmetric monoidal functor $[\![\cdot]\!] : \mathbf{ZX_f} \to \mathbf{FHilb}$ where **FHilb** is the category of finite-dimensional Hilbert spaces. On objects, it is defined as $[\![(a_i)_{i=1}^n]\!] = \mathbb{C}^A$ where $A = \prod_{i=1}^n a_i$, and on generators, it is given as follows:



where $\overrightarrow{r} \in \mathbb{C}^a$ and $r_0 := 1$. Any other morphisms can be defined compositionally: $[\![D_1 \otimes D_2]\!] = [\![D_1]\!] \otimes [\![D_2]\!]$, and $[\![D_1 \circ D_2]\!] = [\![D_1]\!] \circ [\![D_2]\!]$. By functoriality of the standard interpretation, the general spiders have the following correspondence in **FHilb**:



where $a = \min_{i=0}^{n+m} a_i$, $\overrightarrow{r} = (r_1, \cdots, r_{a-1})$, and $r_0 = 1$; moreover, for $0 \le j_p, k_q < a$,

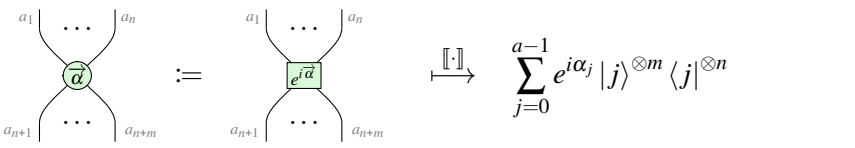


The mixed-dimensional Z-spider can have legs of varying dimensions which needs some further explanation: In the qudit setting, a Z-spider behaves as the generalized Kronecker delta — it ensures that the same basis state is present on each of its legs. In our mixed-dimensional calculus, this behaviour is preserved by selecting the k-th standard basis on each leg for any k less than the minimal dimension. As such behaviour cannot be defined for the remaining basis states, we set their coefficients to zero. In other words, these basis states are not included in the sum.

2.3 Notations

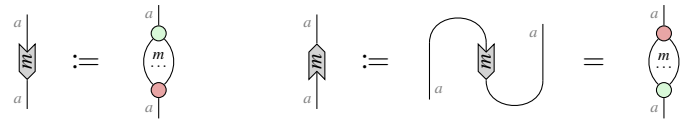
Here, we define some useful notations that we use throughout the paper.

- The original green circle spider [16, 49] can be defined using the Z box:



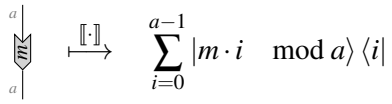
where $a = \min_{i=0}^{n+m} a_i$, $\overrightarrow{\alpha} = (\alpha_1, \cdots, \alpha_{a-1})$, $\alpha_0 := 0$, $e^{i\overrightarrow{\alpha}} = (e^{i\alpha_1}, \cdots, e^{i\alpha_{a-1}})$, and $\alpha_i \in [0, 2\pi)$.

- A *multiplier* [12] labelled by m corresponds to a Z- and X-spider connected with m wires. Unlike in the qubit case, a green and a red spider can be connected with more than one wire. In fact, the Hopf law generalizes to a connections in dimension a (Lemma 16), implying that the multiplier may be labelled modulo a.



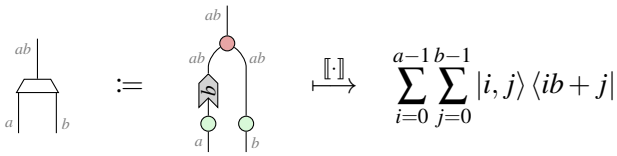
(MU)

Then, the interpretation of the multiplier is given as follows:



$$\sum_{i=0}^{a-1} |m \cdot i \mod a\rangle \langle i|$$

- We can define the *dimension splitter* of the qufinite ZX-calculus [63] as follows:



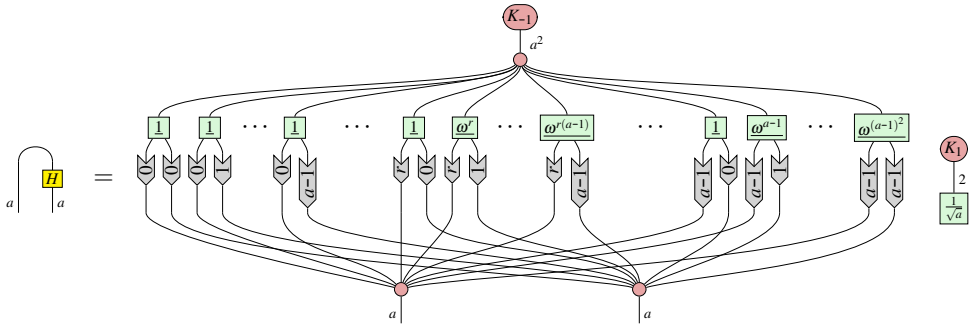
$$\sum_{i=0}^{a-1}\sum_{j=0}^{b-1} |i,j\rangle \langle ib+j|$$

(DD)

Note that this definition matches that of [64, Axiom (DD)].

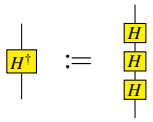
- Throughout this paper, we extensively use the vector $N = (1, \cdots, a-1)$ (where a is the dimension), and elementwise functions on this vector. For example, $\sqrt{N!}$ corresponds to the vector $(\sqrt{1!}, \cdots, \sqrt{(a-1)!})$ and r^N refers to $(r^1, \cdots, r^{a-1})$.
- When two vectors are multiplied or added in the parameter of a Z-spider, we refer to elementwise multiplication or addition of the vectors.
- When the only non-zero element of the parameter in a Z-box is the first or last, we use the following shorthands:

- We use the notations $\overrightarrow{1_k} = (\overbrace{1, \cdots, 1}^{k-1}, 0, \cdots, 0)$, $K_j = \left(j\frac{2\pi}{d}, 2j\frac{2\pi}{d}, \cdots, (d-1)j\frac{2\pi}{d}\right)$ in dimension d, and $\overrightarrow{1} = (1, \cdots, 1)$.
- The qudit generalization of the yellow Hadamard box is defined as follows, for $\omega = e^{i\frac{2\pi}{a}}$:



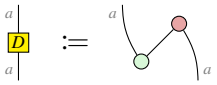
(HD)

- The inverse of the Hadamard box, the yellow $H^\dagger$ box, is defined as follows:



(H†)

- We use a yellow D box to denote the *dualiser* as defined in [16]:

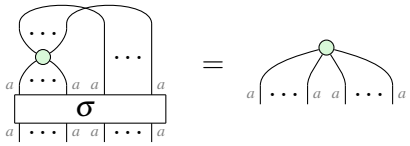


$$\sum_{i=0}^{a-1} |i\rangle \langle a-i \pmod a| . \tag{Du}$$

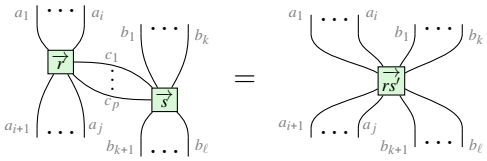
- The Z-spider state with phase $\overrightarrow{1}$ and the X-spider state with phase K_0 are denoted with empty spiders:

2.4 Axiomatization

In this section, we give a set of graphical rewrite rules. First, an equation we assume for the Z-spider is called flexsymmetry [11], enabling us to freely swap the legs of a Z-spider. That is, for an arbitrary permutation of wires σ, the following holds:



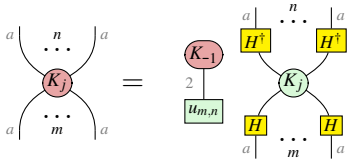
We write $\mathbf{ZX_f} \vdash D_1 = D_2$, if we can use the rules of the calculus to turn D_1 into D_2. Now, let us present the set of axioms for the calculus to perform purely diagrammatic reasoning:



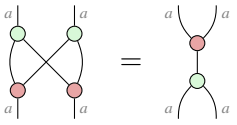
(S1)

where $0 \le j < a$ and $u_{m,n} = a^{\frac{m+n-2}{2}}$

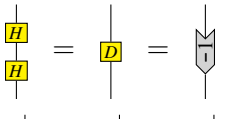
where $A = \{a_t\}_{t=0}^{j}$, $B = \{b_t\}_{t=0}^{\ell}$, $C = \{c_t\}_{t=0}^{p}$, $M = \min(A \cup B \cup C)$, $m = \min(A \cup C)$, $n = \min(B \cup C)$, $v = \min(A \cup B)$, $w = \min(C)$, $\overrightarrow{r} = (r_1, \ldots, r_{m-1})$, $\overrightarrow{s} = (s_1, \ldots, s_{n-1})$, and $\overrightarrow{rs'} = (r_1 s_1, \ldots, r_{M-1} s_{M-1}, \underbrace{0, \ldots, 0}_{\max(v-w,0)})$.



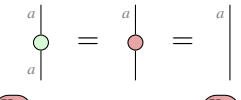
(HZ)



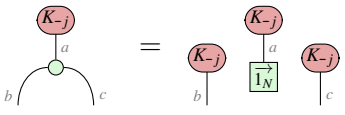
(B2)



(P1)

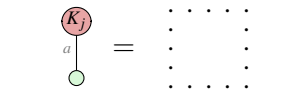


(S2)

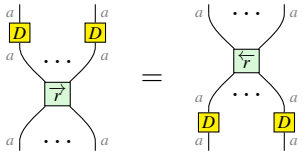


(K0)

where $N = \min\{a,b,c\}$.

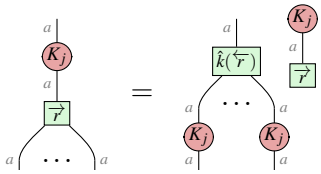


(EPT)



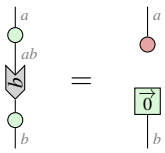
(D1)

where $\overleftarrow{r} = (r_{a-1}, \cdots, r_1)$

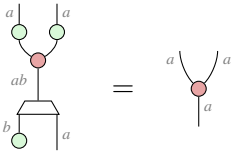


(K2)

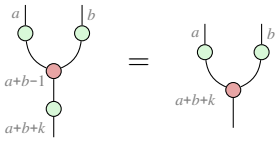
where $\hat{k}(\overleftarrow{r}) = \left(\frac{r_{1-j}}{r_{a-j}}, \ldots, \frac{r_{a-1-j}}{r_{a-j}}\right)$



(HP)

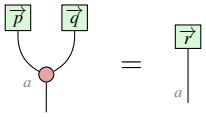


(XM)



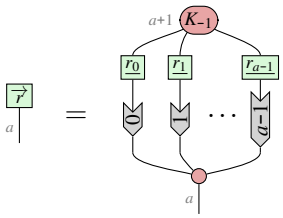
(DA)

where $k \in \mathbb{N}$.

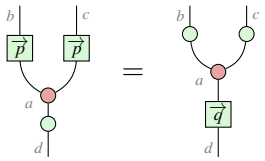


(PA)

where $r_k = \sum_{i=0}^{a-1} p_i q_{k-i \bmod a}$ is the k^{th} entry of $\overrightarrow{r}$.

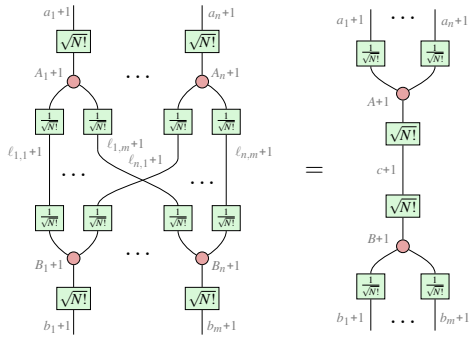


(ZNF)



(PC)

where $\overrightarrow{q}$ is a vector such that for all $0 \le i < b$ and $0 \le j < c$ we have $p_i p_j = q_{i+j \pmod a}$.



(WW)

where $c \ge \min(\sum a_i, \sum b_i)$, $\ell_{ij} = \min(a_i, b_j)$, $A_i = \sum_k \ell_{i,k}$, $B_i = \sum_k \ell_{k,i}$, $A = \sum_k a_k$, and $B = \sum_k b_k$.

2.4.1 Explanation of Axioms

While many of the axioms have already been presented in other papers, some appear here for the first time; therefore, this section provides a brief explanation of the newly introduced axioms.

- (S1) is the generalization of the qudit fusion rule for mixed-dimensional Z-spiders. The phases are multiplied but the vector needs to be cut off at the minimal dimension.
- (D1) generalizes [48, (D1)] for an arbitrary number of legs, making [48, (H1)] redundant.
- (K2) generalizes [48, (K2)] and [48, (K1)].

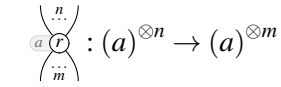
- (K0) is the mixed-dimensional copy rule. The scalar is 0, if the basis state $|j\rangle$ exceeds the capacity of the output dimension, and 1 otherwise.
- (PA), standing for *Phase Addition*, expresses how an X-spider sums up two Z-box states.
- (PC), standing for *Phase Commute*, commutes a class of Z-boxes through an X-spider.
- (HP) is the translation of the (h) rule of the finite-dimensional ZW-calculus.
- The diagrams in (DA) implement the scalar-less W-node, with the rule stating that we can change the internal dimension as long as it is sufficiently high.
- (ZNF) maps an arbitrary Z-spider into its normal form.
- (XM) asserts that an X-spider can also be implemented as a scalar-less W-node composed with a modulo box; see Section 4.1 for further explanation.
- (WW) directly translates the (b_2) rule of the finite-dimensional ZW-calculus.

3 Finite-Dimensional ZW-calculus [28]

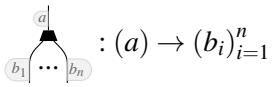
This section recapitulates the finite-dimensional ZW-calculus, in accordance with [28], discussing its generators with their interpretations and presenting the complete axiomatization of the calculus.[1]

3.1 Generators

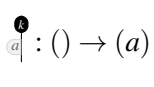
We define the symmetric monoidal category $\mathbf{ZW_f}$ with objects as lists of dimensions $(d_i)_{i=1}^n$ where $d_i \in \mathbb{N}$ for all $0 < i \leq n$, and morphisms generated by the following diagrams, for any $a, b, b_j, n, m \in \mathbb{N}$, $0 < j \leq n$, $0 < k \leq a$, and $r \in \mathbb{C}$:



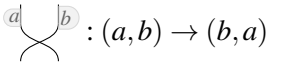
$: (a)^{\otimes n} \to (a)^{\otimes m}$



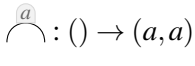
$: (a) \to (b_i)_{i=1}^n$



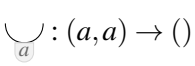
$: () \to (a)$



$: (a,b) \to (b,a)$

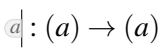


$: () \to (a,a)$



$: (a,a) \to ()$

$r : () \to ()$



$: (a) \to (a)$

Compositions are given in the usual way.

3.2 Interpretation

The interpretation of a diagram in $\mathbf{ZW_f}$ is a symmetric monoidal functor $[\![\cdot]\!] : \mathbf{ZW_f} \to \mathbf{FHilb}$. On objects, it is defined as $[\![(a_i)_{i=1}^n]\!] = \mathbb{C}^A$ where $A = \prod_{i=1}^n (a_i + 1)$. The interpretation of the Z-spider is as follows, for any $r \in \mathbb{C}$:

$\overset{[\![\cdot]\!]}{\longmapsto} \quad \sum_{k=0}^{a} r^k \sqrt{k!}^{n+m-2} |k^m\rangle\langle k^n|$

[1] This paper builds on the equational theory presented in Ref. [28]; however, the referenced paper has since been updated and published, with a slightly modified set of rewrite rules; see Ref. [29].

The W-node and its interpretation are given as follows:

$$\llbracket\cdot\rrbracket \longmapsto \sum_{\substack{0\le k_i\le b_i \\ k_1+\ldots+k_n\le a}} \sqrt{\binom{k_1+\ldots+k_n}{k_1,\ldots,k_n}}\,|k_1,\ldots,k_n\rangle\langle k_1+\ldots+k_n|$$

where $a \geq \max_{i=1}^{n} b_i$. The interpretations of the remaining generators are as follows:

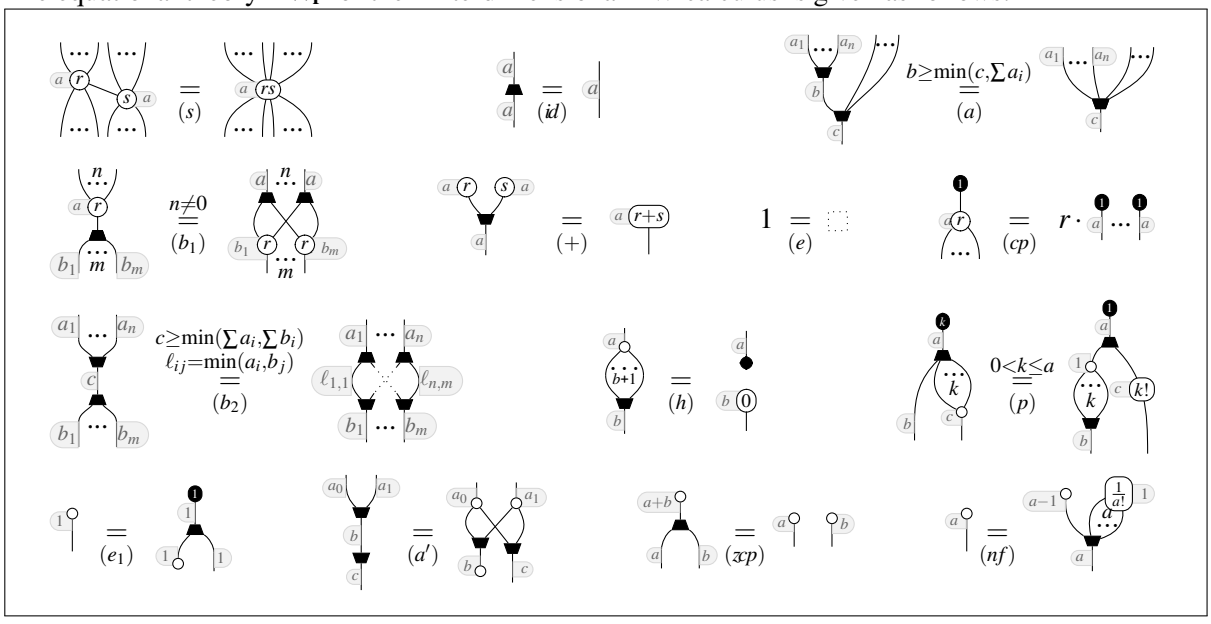
$$\llbracket\cdot\rrbracket \longmapsto \begin{cases} \sqrt{k!}\,|k\rangle & \text{if } 0<k\le a \\ \vec{0} & \text{otherwise} \end{cases} \qquad r \longmapsto r \qquad \llbracket\cdot\rrbracket \longmapsto \sum_{k=0}^{a} |k\rangle\langle k|$$

$$\llbracket\cdot\rrbracket \longmapsto \sum_{k=0}^{a}\sum_{l=0}^{b} |\ell,k\rangle\langle k,\ell| \qquad \llbracket\cdot\rrbracket \longmapsto \sum_{k=0}^{a} |k,k\rangle \qquad \llbracket\cdot\rrbracket \longmapsto \sum_{k=0}^{a} \langle k,k|$$

It is worth pointing out that the dimensions of the wire in $\mathbf{ZX_f}$ and $\mathbf{ZW_f}$ are not the same. In the ZX-calculus d is used to indicate that the wire carries a d-dimensional qudit while in the case of the ZW-calculus it means a $(d+1)$-dimensional qudit. On ZX-diagrams, dimension is marked with grey text by the wire, whereas text in light grey bubble next to the wire indicates the dimension of the ZW-calculus.

3.3 Axioms

The equational theory $\mathbf{ZW_f}$ for the finite-dimensional ZW-calculus is given as follows:



Proposition 1 (Completeness of ZW-calculus)**.** *For any two* $\mathbf{ZW_f}$*-diagrams of the same type* $D_1 : A \to B$ *and* $D_2 : A \to B$*,*

$$\llbracket D_1\rrbracket = \llbracket D_2\rrbracket \iff \mathbf{ZW_f} \vdash D_1 = D_2$$

The proof is the content of [28, Theorem 3].

Remark 1. *While in* $\mathbf{ZX_f}$ *all wire dimensions must be greater than or equal to* 2*, the finite-dimensional ZW-calculus allows wires of dimension* 1 *(labelled by* 0*). Therefore, we require the completeness of the finite-dimensional ZW-calculus restricted to having no* 0*-labeled wires. This variant of the ZW completeness is still complete as the proof does not use* 0*-labeled wires.*

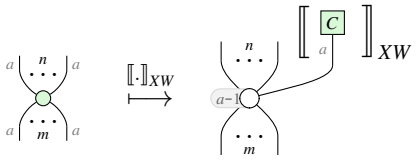
4 Completeness from Translation

In this section, we prove the completeness of the finite-dimensional ZX-calculus. Our proof strategy is based on the one outlined in [37, 44]. The idea of the proof is to define a translation from and to a complete language — the finite-dimensional ZW-calculus in our case. Then, if one can prove certain properties of these translation functors, such as soundness and invertibility, then we obtain completeness.

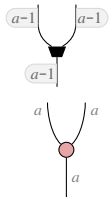
4.1 ZX-to-ZW Translation

Here, we define the *ZX-to-ZW translation functor*, $[\![.]\!]_{XW} : \mathbf{ZX_f} \to \mathbf{ZW_f}$. On objects, we have $[\![(a_i)_{i=1}^n]\!]_{XW} = (a_i - 1)_{i=1}^n$. On morphisms, we show how it maps the generators of $\mathbf{ZX_f}$ to the generators of $\mathbf{ZW_f}$; the other morphisms can be defined compositionally: $[\![D_1 \otimes D_2]\!]_{XW} = [\![D_1]\!]_{XW} \otimes [\![D_2]\!]_{XW}$, and $[\![D_1 \circ D_2]\!]_{XW} = [\![D_1]\!]_{XW} \circ [\![D_2]\!]_{XW}$. We first show how the generators defining a general mixed-dimensional Z-spider are translated, starting with the Z-spider with a single output and the embedding:

The former interpretation is based on the normal form of the qudit ZW-calculus [28, Definition 3], and the embedding follows from comparing the interpretations. Since the Z-spiders of the two calculi match up to a $C = \left(\frac{1}{\sqrt{N!}}\right)^{m+n-1}$ vector parameter, its translation is as follows:



Now, we focus on expressing generators that include X-spiders. The translation of computational basis states follows from the interpretations. To express the X-spider in the finite-dimensional ZW-calculus, we first point out that the interpretation of the W- and X-spider are closely related. Other than the scalar factors, the only difference is that the W-spider results in sums of the input states that are smaller than the output dimension while the X-spider sums elements modulo a:



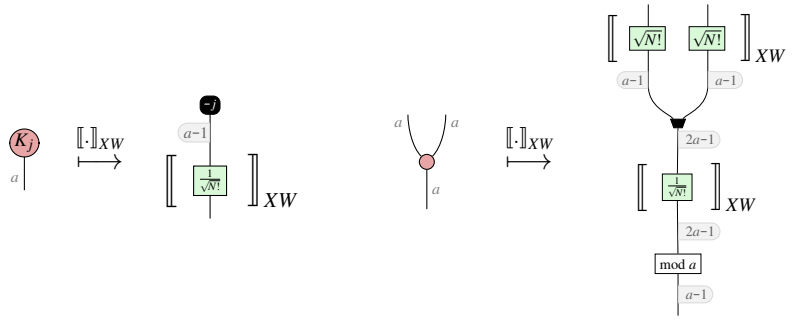
$$\overset{[\![\cdot]\!]}{\longmapsto} \quad |k+\ell\rangle\langle k,\ell| \quad \text{for} \quad k,\ell < a \text{ and } k+\ell < a,$$

$$\overset{[\![\cdot]\!]}{\longmapsto} \quad |k+\ell \mod a\rangle\langle k,\ell| \quad \text{for} \quad k,\ell < a$$

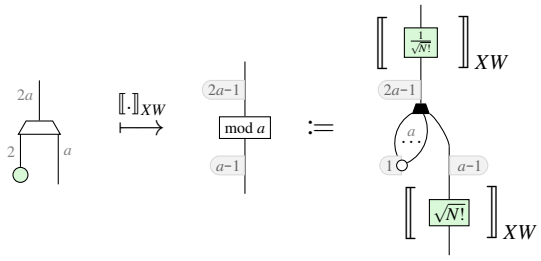
Setting the output dimension of the W-spider to be $2a$, and applying a modulo a gate, we obtain the translation of the X spider, that is,

$$|j \mod a\rangle\langle j|k+\ell\rangle\langle k,\ell| \text{ for } k,\ell < a \wedge k+\ell < 2a, \iff |k+\ell \mod a\rangle\langle k,\ell| \text{ for } k,\ell < a,$$

Diagrammatically, these translations are given as follows:



where the modulo a gadget is given as follows:



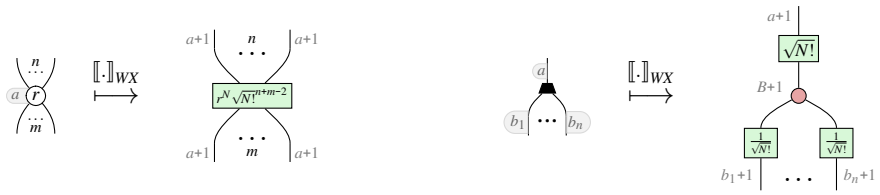
The translations of the remaining generators, the identity and the swap, are trivial.

Lemma 1. *The ZX-to-ZW translation functor $[\![.]\!]_{XW}$ preserves the semantics, that is, $[\![[\![D]\!]_{XW}]\!] = [\![D]\!]$ for any ZX-diagram D.*

To verify this lemma, it suffices to apply the computational basis states to both the original ZX-diagram D and its translated ZW-diagram $[\![D]\!]_{XW}$. By evaluating these diagrams under all possible computational basis inputs, we can verify their equivalence under interpretations.

4.2 ZW-to-ZX Translation

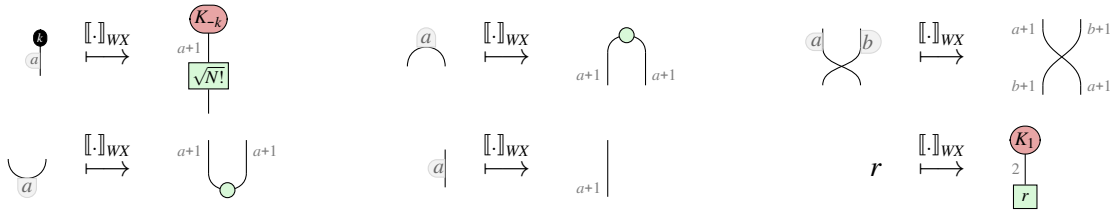
Here, we define the *ZW-to-ZX translation functor*, $[\![.]\!]_{WX} : \mathbf{ZW_f} \to \mathbf{ZX_f}$. On objects, we have $[\![(a_i)_{i=1}^n]\!]_{WX} = (a_i+1)_{i=1}^n$. On morphisms, we show how it maps the generators of $\mathbf{ZW_f}$ to the generators of $\mathbf{ZX_f}$; the other morphisms can be defined compositionally: $[\![D_1 \otimes D_2]\!]_{WX} = [\![D_1]\!]_{WX} \otimes [\![D_2]\!]_{WX}$, and $[\![D_1 \circ D_2]\!]_{WX} = [\![D_1]\!]_{WX} \circ [\![D_2]\!]_{WX}$. For $B = \sum_{i=1}^n b_i$,



The translation of the Z-spider follows directly from its interpretation. The intuition behind the translation of the W-node is similar to that used for translating the X-spider from ZX to ZW. Up to scalars, both nodes sum up basis states, but the X-spider also takes the result modulo the dimension. Embedding the X-spider in a sufficiently high dimension, we can ensure that the sum of the basis states is always less than the dimension and thus the modulo is never taken. Once we have this, the mixed-dimensional

Z-spider can remove sums that are larger than the output dimension of the W-node resulting in the same interpretation.

The remaining translations follow directly from the interpretation:



Lemma 2. *The ZW-to-ZX translation functor preserves the semantics, that is,* $[\![[\![D]\!]_{WX}]\!] = [\![D]\!]$ *for any ZW-diagram D.*

To verify this lemma, it suffices to apply the computational basis states to both the original ZW-diagram D and its translated ZX-diagram $[\![D]\!]_{WX}$. By evaluating these diagrams under all possible computational basis inputs, we can confirm that their interpretations yield identical results.

4.3 Proof of Completeness

In this section, we prove the completeness of $\mathbf{ZX_f}$, the finite-dimensional ZX-calculus.

Lemma 3. *For an arbitrary ZX-diagram D,* $\mathbf{ZX_f} \vdash [\![[\![D]\!]_{XW}]\!]_{WX} = D$.

Due to the functoriality of the monoidal functors $[\![.]\!]_{XW}$ and $[\![.]\!]_{WX}$, it suffices to show that the above lemma holds for all generators of $\mathbf{ZX_f}$. Appendix A.2 discusses these lemmas.

Proposition 2. *If* $\mathbf{ZW_f} \vdash D_1 = D_2$ *then* $\mathbf{ZX_f} \vdash [\![D_1]\!]_{WX} = [\![D_2]\!]_{WX}$.

By the functoriality of $[\![.]\!]_{WX}$, we only need to show that the translations of all axioms of $\mathbf{ZW_f}$ are derivable in $\mathbf{ZX_f}$. These proofs are the content of Appendix A.3.

Theorem 1 (Completeness)**.** *For finite-dimensional Hilbert spaces, the ZX-calculus is universally complete: For any two ZX-diagrams of the same type* $D_1 : A \to B$ *and* $D_2 : A \to B$*, if* $[\![D_1]\!] = [\![D_2]\!]$*, then* $\mathbf{ZX_f} \vdash D_1 = D_2$.

Proof. Suppose $D_1, D_2 \in \mathbf{ZX_f}$ such that $[\![D_1]\!] = [\![D_2]\!]$ and they have the same type. By Lemma 1, $[\![[\![D_1]\!]_{XW}]\!] = [\![D_1]\!] = [\![D_2]\!] = [\![[\![D_2]\!]_{XW}]\!]$. By the completeness of ZW-calculus, $\mathbf{ZW_f} \vdash [\![D_1]\!]_{XW} = [\![D_2]\!]_{XW}$. Now by Proposition 2, $\mathbf{ZX_f} \vdash [\![[\![D_1]\!]_{XW}]\!]_{WX} = [\![[\![D_2]\!]_{XW}]\!]_{WX}$. Finally, by Lemma 3, $\mathbf{ZX_f} \vdash [\![[\![D_1]\!]_{XW}]\!]_{WX} = D_1$, $\vdash [\![[\![D_2]\!]_{XW}]\!]_{WX} = D_2$, therefore, $\mathbf{ZX_f} \vdash D_1 = D_2$. □

5 Conclusion and Further Work

In this paper, we presented the finite-dimensional ZX-calculus, which generalizes the qudit ZX-calculus with mixed-dimensional Z-spiders. We proved the completeness of the language by translating to and from the finite-dimensional ZW-calculus and showing the translation is invertible.

While the ZW-calculus is close to minimal, we did not consider the minimality of the ZX-calculus. It would be interesting to investigate the smallest set of rules needed for the ZX-calculus to be complete for finite-dimensional Hilbert spaces.

Secondly, our calculus generalizes the Z-spider to mixed dimensions but leaves the X-spider unchanged from qudits. Figuring out a nice mixed-dimensional generalization of the X-spider would allow us more flexibility in doing mixed-dimensional reasoning.

Next, we used the ZW-calculus to prove the completeness of the ZX-calculus. By translating the ZW normal form, we easily obtain a normal form for the ZX-calculus. This normal form can be used to prove the completeness of the ZX-calculus directly, by showing that every diagram can be reduced to the normal form. We leave proving this direct completeness without relying on the result of a different calculus for future work.

Another idea to explore could be obtaining completeness through translation to and from the qufinite ZXW-calculus. As it is a superset of finite-dimensional ZX-calculus, one side of the translation is trivial. However, we suspect that given the close-minimality of the finite-dimensional ZW-calculus, the translation presented in this paper results in a more minimal set of rules.

Finally, we note that the ZX-calculus presented here allows for Z-boxes labelled with arbitrary complex numbers, in contrast with the original ZX-calculus where spider coefficients are restricted to be phases. Further work could be done to probe whether the ZX-calculus with only phase coefficients is complete for finite-dimensional Hilbert spaces.

Acknowledgements

We would like to thank Alexander Cowtan and Lia Yeh for their detailed feedback and several suggestions for improvement. We thank the anonymous reviewers at QPL 2024 for their valuable feedback and pointing out a mistake in a translation of the previous version of this paper. We would also like to thank the reviewers at QPL 2025 for their very detailed reviews and helpful suggestions. BP is supported by the Engineering and Physical Sciences Research Council grant number EP/Z002230/1, '(De)constructing quantum software (DeQS)'. RS is supported by the Clarendon Fund Scholarship.

References

[1] Miriam Backens (2014): *The ZX-calculus Is Complete for Stabilizer Quantum Mechanics*. *New Journal of Physics* 16(9), p. 093021, doi:10.1088/1367-2630/16/9/093021.

[2] Miriam Backens (2014): *The ZX-calculus Is Complete for the Single-Qubit Clifford+T Group*. *Electronic Proceedings in Theoretical Computer Science* 172, pp. 293–303, doi:10.4204/EPTCS.172.21.

[3] Miriam Backens & Aleks Kissinger (2019): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-Linearity*. In Peter Selinger & Giulio Chiribella, editors: *Proceedings of the 15th International Conference on Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 287, Open Publishing Association, Halifax, Canada, pp. 23–42, doi:10.4204/EPTCS.287.2.

[4] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and Back Again: A Circuit Extraction Tale*. *Quantum* 5, p. 421, doi:10.22331/q-2021-03-25-421.

[5] Miriam Backens, Simon Perdrix & Quanlong Wang (2020): *Towards a Minimal Stabilizer ZX-calculus*. *Logical Methods in Computer Science* 16(4), doi:10.23638/LMCS-16(4:19)2020.

[6] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Fast and effective techniques for t-count reduction via spider nest identities*. In Steven T. Flammia, editor: *15th conference on the theory of quantum computation, communication and cryptography (TQC 2020)*, *Leibniz international proceedings in informatics (lipics)* 158, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, p. 11:1–11:23, doi:10.4230/LIPIcs.TQC.2020.11.

[7] Hector Bombin, Daniel Litinski, Naomi Nickerson, Fernando Pastawski & Sam Roberts (2024): *Unifying Flavors of Fault Tolerance with the ZX Calculus*. *Quantum* 8, p. 1379, doi:10.22331/q-2024-06-18-1379.

[8] Robert I. Booth & Titouan Carette (2022): *Complete ZX-calculi for the Stabiliser Fragment in Odd Prime Dimensions*. In Stefan Szeider, Robert Ganian & Alexandra Silva, editors: *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, *Leibniz International Proceedings in Informatics (Lipics)* 241, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 24:1–24:15, doi:10.4230/LIPIcs.MFCS.2022.24. Available at https://drops.dagstuhl.de/opus/volltexte/2022/16822.

[9] Robert I. Booth, Titouan Carette & Cole Comfort (2024): *Graphical Symplectic Algebra*. arXiv:2401.07914.

[10] Tristan Cam & Simon Martiel (2023): *Speeding up Quantum Circuits Simulation Using ZX-Calculus*. arXiv:2305.02669.

[11] Titouan Carette (2021): *Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations*. Ph.D. thesis, Loria, Université de Lorraine. Available at https://hal.archives-ouvertes.fr/tel-03468027.

[12] Titouan Carette, Dominic Horsman & Simon Perdrix (2019): *SZX-Calculus: Scalable Graphical Quantum Reasoning*. In Peter Rossmanith, Pinar Heggernes & Joost-Pieter Katoen, editors: *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 138, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 55:1–55:15, doi:10.4230/LIPIcs.MFCS.2019.55.

[13] Julien Codsi & John van de Wetering (2023): *Classically Simulating Quantum Supremacy IQP Circuits through a Random Graph Approach*. arXiv:2212.08609.

[14] B. Coecke & R. Duncan (2007): *Interacting Quantum Observables*. Available at www.cs.ox.ac.uk/people/bob.coecke/GreenRed.pdf.

[15] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir & Igor Walukiewicz, editors: *Automata, Languages and Programming*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 298–310, doi:10.1007/978-3-540-70583-3_25. Available at http://personal.strath.ac.uk/ross.duncan/papers/iqo-icalp.pdf.

[16] Bob Coecke & Ross Duncan (2011): *Interacting Quantum Observables: Categorical Algebra and Diagrammatics*. *New Journal of Physics* 13(4), p. 043016, doi:10.1088/1367-2630/13/4/043016.

[17] Bob Coecke & Bill Edwards (2011): *Three Qubit Entanglement within Graphical Z/X-calculus*. *Electronic Proceedings in Theoretical Computer Science* 52, pp. 22–33, doi:10.4204/EPTCS.52.3.

[18] Bob Coecke & Stefano Gogioso (2022): *Quantum in Pictures*. Quantinuum.

[19] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes*. Cambridge University Press, doi:10.1017/9781316219317.

[20] Bob Coecke, Aleks Kissinger, Alex Merry & Shibdas Roy (2011): *The GHZ/W-calculus Contains Rational Arithmetic*. *Electronic Proceedings in Theoretical Computer Science* 52, pp. 34–48, doi:10.4204/EPTCS.52.4.

[21] Cole Comfort (2023): *The Algebra for Stabilizer Codes*. arXiv:2304.10584.

[22] Alexander Cowtan (2022): *Qudit Lattice Surgery*. arXiv:2204.13228.

[23] Alexander Cowtan & Simon Burton (2024): *CSS Code Surgery as a Universal Construction*. *Quantum* 8, p. 1344, doi:10.22331/q-2024-05-14-1344.

[24] Niel de Beaudrap & Dominic Horsman (2020): *The ZX Calculus Is a Language for Surface Code Lattice Surgery*. *Quantum* 4, p. 218, doi:10.22331/q-2020-01-09-218.

[25] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-Diagrams Can Be #P-Hard*. In Mikołaj Bojańczyk, Emanuela Merelli & David P. Woodruff, editors: *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 229, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 119:1–119:19, doi:10.4230/LIPIcs.ICALP.2022.119.

[26] Giovanni de Felice & Bob Coecke (2023): *Quantum Linear Optics via String Diagrams*. In Stefano Gogioso & Matty Hoban, editors: *Proceedings 19th International Conference on Quantum Physics and Logic, Wolfson College, Oxford, UK, 27 June - 1 July 2022*, *Electronic Proceedings in Theoretical Computer Science* 394, Open Publishing Association, pp. 83–100, doi:10.4204/EPTCS.394.6.

[27] Giovanni de Felice, Razin A. Shaikh, Boldizsár Poór, Lia Yeh, Quanlong Wang & Bob Coecke (2023): *Light-Matter Interaction in the ZXW Calculus*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 20–46, doi:10.4204/EPTCS.384.2.

[28] Marc de Visme & Renaud Vilmart (2024): *Minimality in Finite-Dimensional ZW-Calculi*. arXiv:2401.16225.

[29] Marc de Visme & Renaud Vilmart (2025): *Minimality in Finite-Dimensional ZW-Calculi*. In Jörg Endrullis & Sylvain Schmitz, editors: *33rd EACSL Annual Conference on Computer Science Logic (CSL 2025)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 326, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 49:1–49:22, doi:10.4230/LIPIcs.CSL.2025.49. Available at https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2025.49.

[30] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-Theoretic Simplification of Quantum Circuits with the ZX-calculus*. *Quantum* 4, p. 279, doi:10.22331/q-2020-06-04-279.

[31] Ross Duncan & Simon Perdrix (2010): *Rewriting Measurement-Based Quantum Computations with Generalised Flow*. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide & Paul G. Spirakis, editors: *Automata, Languages and Programming*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 285–296, doi:10.1007/978-3-642-14162-1_24. Available at http://personal.strath.ac.uk/ross.duncan/papers/gflow.pdf.

[32] Selma Dündar-Coecke, Lia Yeh, Caterina Puca, Sieglinde M.-L. Pfaendler, Muhammad Hamza Waseem, Thomas Cervoni, Aleks Kissinger, Stefano Gogioso & Bob Coecke (2023): *Quantum Picturalism: Learning Quantum Theory in High School*. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 03, pp. 21–32, doi:10.1109/QCE57702.2023.20321. arXiv:2312.03653.

[33] Giovanni de Felice, Boldizsár Poór, Lia Yeh & William Cashman (2024): *Fusion and Flow: Formal Protocols to Reliably Build Photonic Graph States*. arXiv:2409.13541.

[34] Amar Hadzihasanovic (2015): *A Diagrammatic Axiomatisation for Qubit Entanglement*. In: *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '15, IEEE Computer Society, USA, pp. 573–584, doi:10.1109/LICS.2015.59. arXiv:1501.07082.

[35] Amar Hadzihasanovic (2017): *The Algebra of Entanglement and the Geometry of Composition*. Ph.D. thesis, University of Oxford. arXiv:1709.08086.

[36] Jiaxin Huang, Sarah Meng Li, Lia Yeh, Aleks Kissinger, Michele Mosca & Michael Vasmer (2023): *Graphical CSS Code Transformation Using ZX Calculus*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 1–19, doi:10.4204/EPTCS.384.1.

[37] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, pp. 559–568, doi:10.1145/3209108.3209131. arXiv:1705.11151.

[38] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, pp. 569–578, doi:10.1145/3209108.3209139. arXiv:1801.10142.

[39] Aleks Kissinger (2022): *Phase-free ZX diagrams are CSS codes (...or how to graphically grok the surface code)*. arXiv:2204.14038.

[40] Aleks Kissinger & John van de Wetering (2020): *Reducing T-count with the ZX-calculus*. *Physical Review A* 102(2), p. 022406, doi:10.1103/PhysRevA.102.022406. arXiv:1903.10477.

[41] Mark Koch, Richie Yeung & Quanlong Wang (2024): *Contraction of ZX Diagrams with Triangles via Stabiliser Decompositions*. *Physica Scripta* 99(10), p. 105122, doi:10.1088/1402-4896/ad6fd8. arXiv:2307.01803.

[42] Tuomas Laakkonen, Konstantinos Meichanetzidis & John van de Wetering (2023): *Picturing Counting Reductions with the ZH-Calculus*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, Paris, France, pp. 89–113, doi:10.4204/EPTCS.384.6.

[43] Tommy McElvanney & Miriam Backens (2023): *Flow-Preserving ZX-calculus Rewrite Rules for Optimisation and Obfuscation*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 203–219, doi:10.4204/EPTCS.384.12.

[44] Kang Feng Ng & Quanlong Wang (2017): *A Universal Completion of the ZX-calculus*. arXiv:1706.09877.

[45] Brendan Pankovich, Alex Neville, Angus Kan, Srikrishna Omkar, Kwok Ho Wan & Kamil Brádler (2023): *Flexible Entangled State Generation in Linear Optics*. arXiv:2310.06832.

[46] Simon Perdrix & Quanlong Wang (2016): *Supplementarity Is Necessary for Quantum Diagram Reasoning*. In: *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 58, Krakow, Poland, pp. 76:1–76:14, doi:10.4230/LIPIcs.MFCS.2016.76.

[47] Boldizsár Poór, Robert I. Booth, Titouan Carette, John van de Wetering & Lia Yeh (2023): *The Qupit Stabiliser ZX-travaganza: Simplified Axioms, Normal Forms and Graph-Theoretic Simplification*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 220–264, doi:10.4204/EPTCS.384.13.

[48] Boldizsár Poór, Quanlong Wang, Razin A. Shaikh, Lia Yeh, Richie Yeung & Bob Coecke (2023): *Completeness for Arbitrary Finite Dimensions of ZXW-calculus, a Unifying Calculus*. In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, Boston, MA, USA, pp. 1–14, doi:10.1109/LICS56636.2023.10175672. arXiv:2302.12135.

[49] André Ranchin (2014): *Depicting Qudit Quantum Mechanics and Mutually Unbiased Qudit Theories*. *Electronic Proceedings in Theoretical Computer Science* 172, pp. 68–91, doi:10.4204/EPTCS.172.6.

[50] Benjamin Rodatz, Boldizsár Poór & Aleks Kissinger (2024): *Floquetifying Stabiliser Codes with Distance-Preserving Rewrites*. arXiv:2410.17240.

[51] Patrick Roy, John van de Wetering & Lia Yeh (2023): *The Qudit ZH-Calculus: Generalised Toffoli+Hadamard and Universality*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 142–170, doi:10.4204/EPTCS.384.9.

[52] Christian Schröder de Witt & Vladimir Zamdzhiev (2014): *The ZX-calculus Is Incomplete for Quantum Mechanics*. In Bob Coecke, Ichiro Hasuo & Prakash Panangaden, editors: *Proceedings of the 11th Workshop on Quantum Physics and Logic, Kyoto, Japan*, *Electronic Proceedings in Theoretical Computer Science* 172, Open Publishing Association, pp. 285–292, doi:10.4204/EPTCS.172.20.

[53] Razin A. Shaikh & Stefano Gogioso (2022): *Categorical Semantics for Feynman Diagrams*. arXiv:2205.00466.

[54] Razin A. Shaikh, Quanlong Wang & Richie Yeung (2023): *How to Sum and Exponentiate Hamiltonians in ZXW Calculus*. In Stefano Gogioso & Matty Hoban, editors: *Proceedings 19th International Conference on*

Quantum Physics and Logic, Wolfson College, Oxford, UK, 27 June - 1 July 2022, *Electronic Proceedings in Theoretical Computer Science* 394, Open Publishing Association, pp. 236–261, doi:10.4204/EPTCS.394.14.

[55] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *T|ket⟩: A Retargetable Compiler for NISQ Devices*. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92. arXiv:2003.10611.

[56] Alex Townsend-Teague, Julio Magdalena de la Fuente & Markus Kesselring (2023): *Floquetifying the Colour Code*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 265–303, doi:10.4204/EPTCS.384.14.

[57] Alex Townsend-Teague & Konstantinos Meichanetzidis (2022): *Simplification Strategies for the Qutrit ZX-Calculus*. arXiv:2103.06914.

[58] John van de Wetering (2020): *ZX-calculus for the Working Quantum Computer Scientist*. arXiv:2012.13966.

[59] John van de Wetering & Lia Yeh (2023): *Building Qutrit Diagonal Gates from Phase Gadgets*. In Stefano Gogioso & Matty Hoban, editors: *Proceedings 19th International Conference on Quantum Physics and Logic, Wolfson College, Oxford, UK, 27 June - 1 July 2022*, *Electronic Proceedings in Theoretical Computer Science* 394, Open Publishing Association, pp. 46–65, doi:10.4204/EPTCS.394.4.

[60] John van de Wetering, Richie Yeung, Tuomas Laakkonen & Aleks Kissinger (2024): *Optimal Compilation of Parametrised Quantum Circuits*. arXiv:2401.12877.

[61] Renaud Vilmart (2019): *A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–10, doi:10.1109/LICS.2019.8785765. arXiv:1812.09114.

[62] Quanlong Wang (2018): *Qutrit ZX-calculus Is Complete for Stabilizer Quantum Mechanics*. In Bob Coecke & Aleks Kissinger, editors: *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, the Netherlands*, *Electronic Proceedings in Theoretical Computer Science* 266, Open Publishing Association, pp. 58–70, doi:10.4204/EPTCS.266.3.

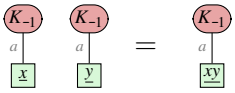
[63] Quanlong Wang (2022): *Qufinite ZX-calculus: A Unified Framework of Qudit ZX-calculi*. arXiv:2104.06429.

[64] Quanlong Wang, Boldizsár Poór & Razin A. Shaikh (2024): *Completeness of Qufinite ZXW Calculus, a Graphical Language for Finite-Dimensional Quantum Theory*. arXiv:2309.13014.

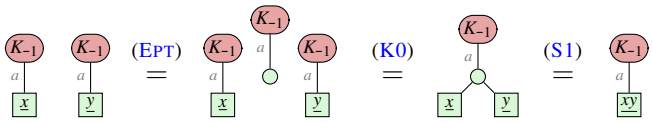
[65] Quanlong Wang, Richie Yeung & Mark Koch (2024): *Differentiating and Integrating ZX Diagrams with Applications to Quantum Machine Learning*. *Quantum* 8, p. 1491, doi:10.22331/q-2024-10-04-1491.

A Lemmas

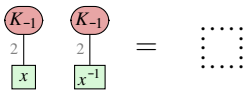
A.1 General Derivations

Lemma 4. 

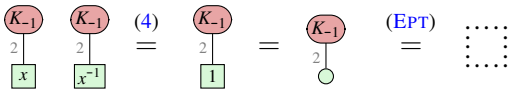
Proof.



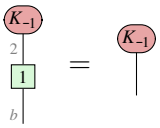
□

Lemma 5. *Suppose $x \neq 0$. Then* 

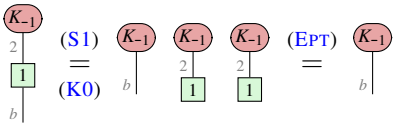
Proof.



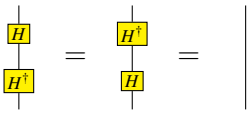
□

Lemma 6. 

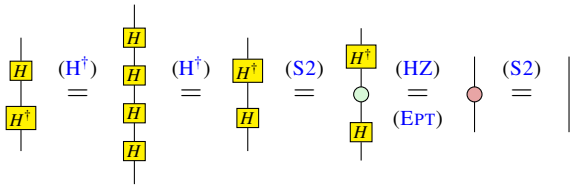
Proof.



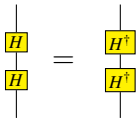
□

Lemma 7. *[63]* 

Proof.

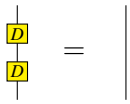


□

Lemma 8. 

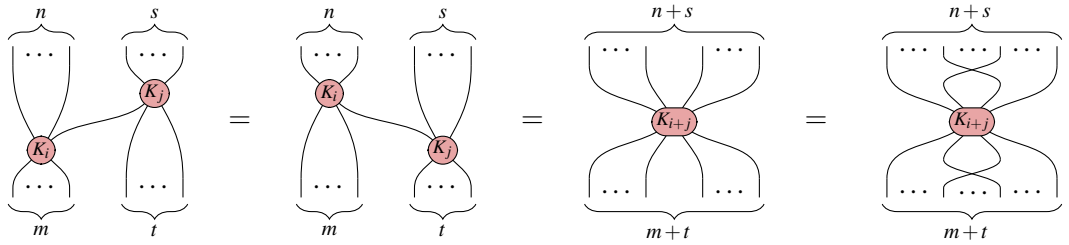
Proof. Same as [48, Lemma 12]. □

Lemma 9. *[63]*



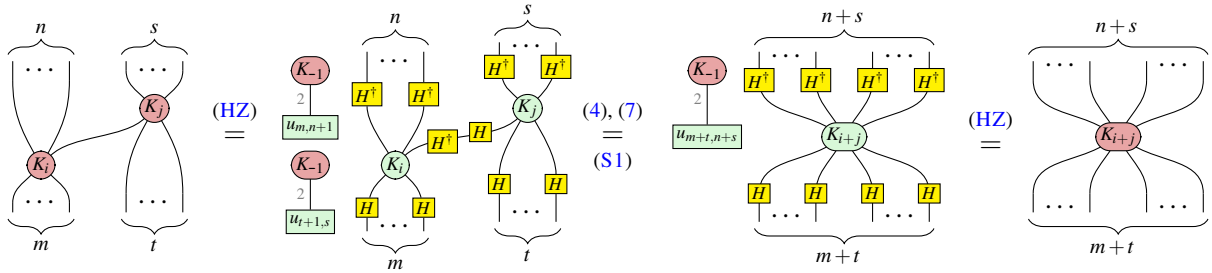
Proof. Same as [48, Lemma 16]. □

Lemma 10. *[63]*



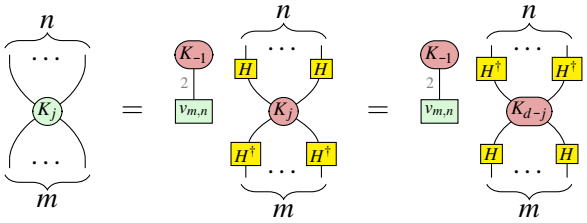
(S4)

Proof.



The other equalities can be proved similarly using the same rules. □

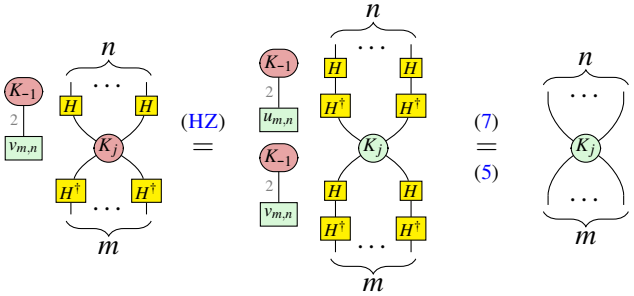
Lemma 11. *[63]*

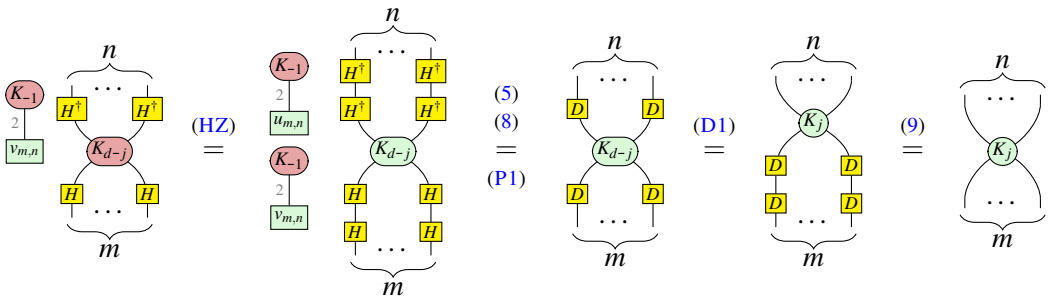


(HX)

where $v_{m,n} = d^{\frac{-m-n+2}{2}}$, $j \in \{0,1,\cdots,d-1\}$. *We also call this equality* (HX).

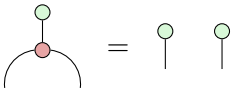
Proof.





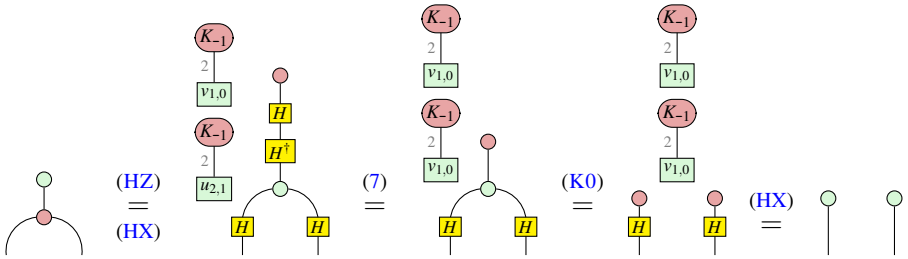
where $u_{m,n} = d^{\frac{m+n-2}{2}}$, $v_{m,n} = d^{\frac{-m-n+2}{2}}$, $j \in \{0,1,\cdots,d-1\}$. □

Lemma 12.



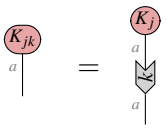
(B3)

Proof.

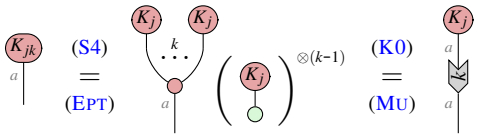


where $u_{2,1} = d^{\frac{1}{2}}$, $v_{1,0} = d^{\frac{1}{2}}$. □

Lemma 13.

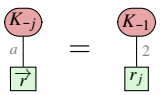


Proof.



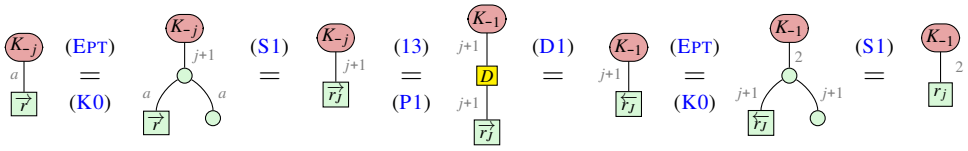
□

Lemma 14. *[63]*



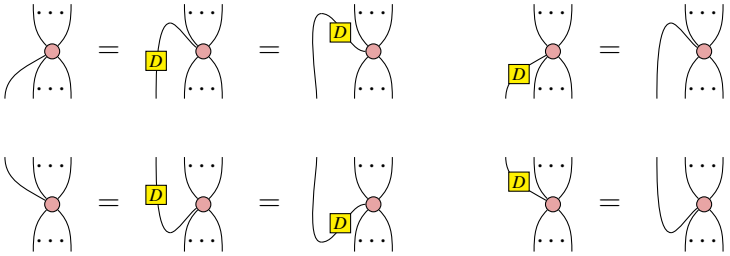
where $\overrightarrow{r} = (r_1,\cdots,r_{a-1})$.

Proof.



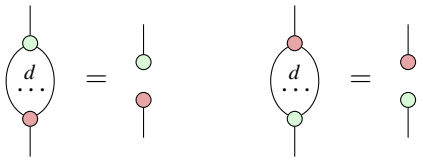
where $\overrightarrow{r_j} = (r_1,\cdots,r_j)$ and $\overleftarrow{r_j} = (r_j,\cdots,r_1)$. □

Lemma 15.



Proof. Same as [48, Lemma 25]. □

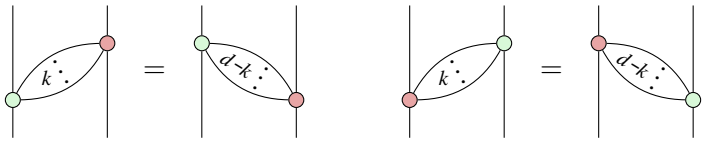
Lemma 16. *[63]*



(Hopf)

Proof. Same as [48, Lemma 29]. □

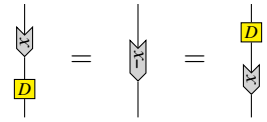
Lemma 17. *[63]*



where $1 \leq k \leq d-1$.

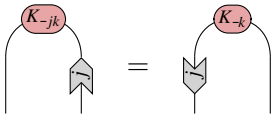
Proof. Same as [48, Lemma 30]. □

Lemma 18. *Suppose* $x \in \{0,\dots,d-1\}$. *Then*

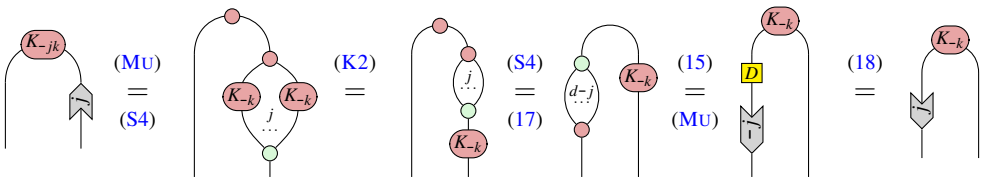


Proof. Same as [48, Lemma 34]. □

Lemma 19.



Proof.

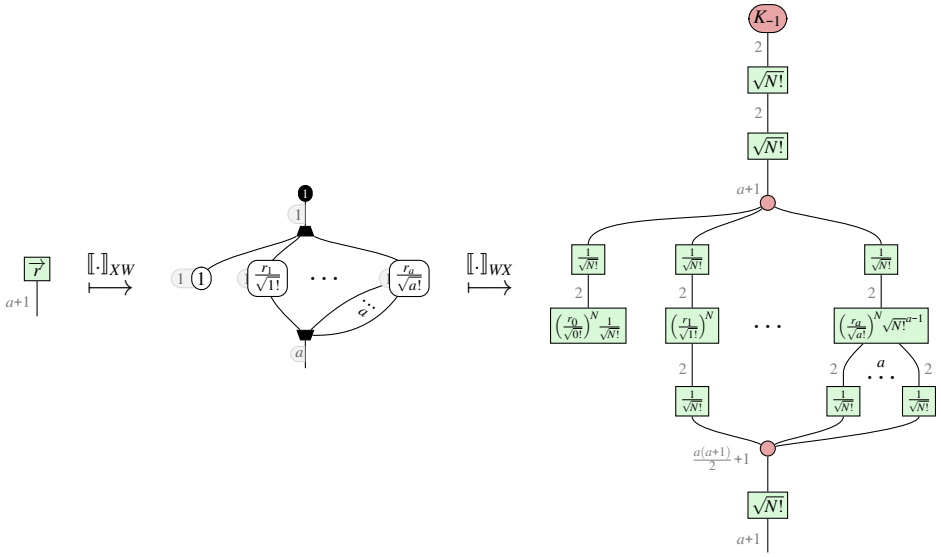


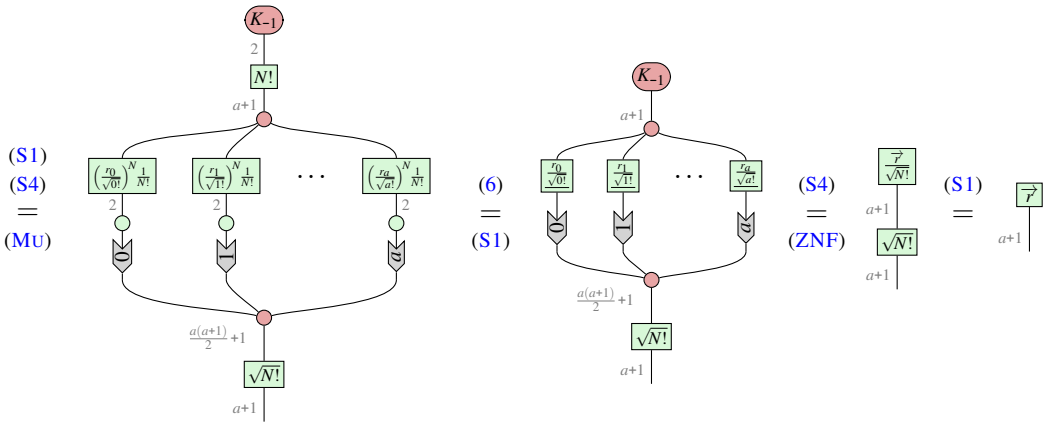
□

A.2 Recovering Generators

Lemma 20.

Proof.

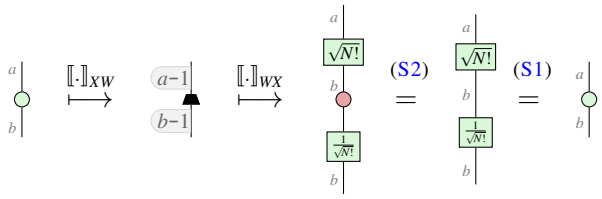




□

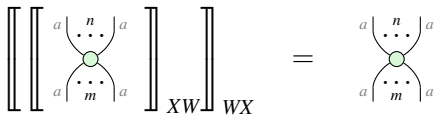
Lemma 21.

Proof. Suppose $a \geq b$. Then

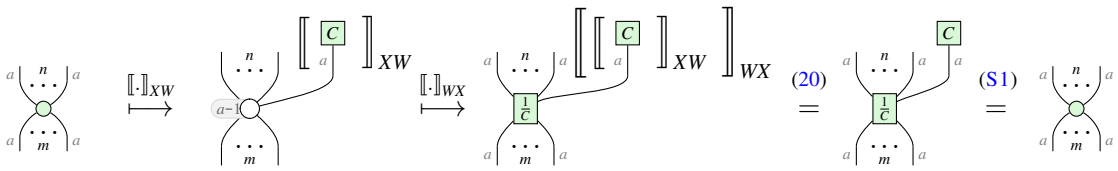


The other case can be proved similarly. □

Lemma 22.

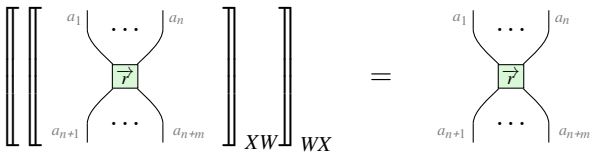


Proof. For $C = \left(\frac{1}{\sqrt{N!}}\right)^{m+n-1}$,



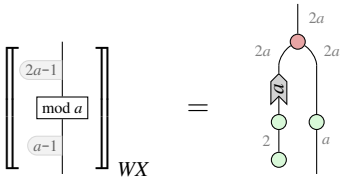
□

Lemma 23.

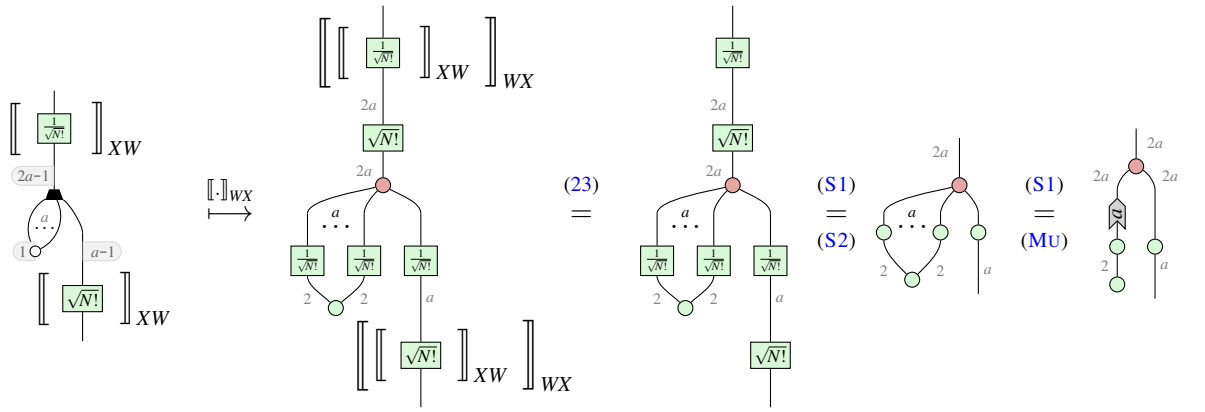


Proof. This follows from Lemmas 20 to 22. □

Lemma 24.

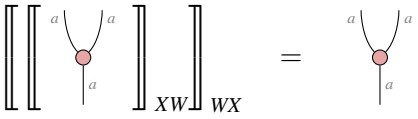


Proof. Expanding the definition of the modulo a box,

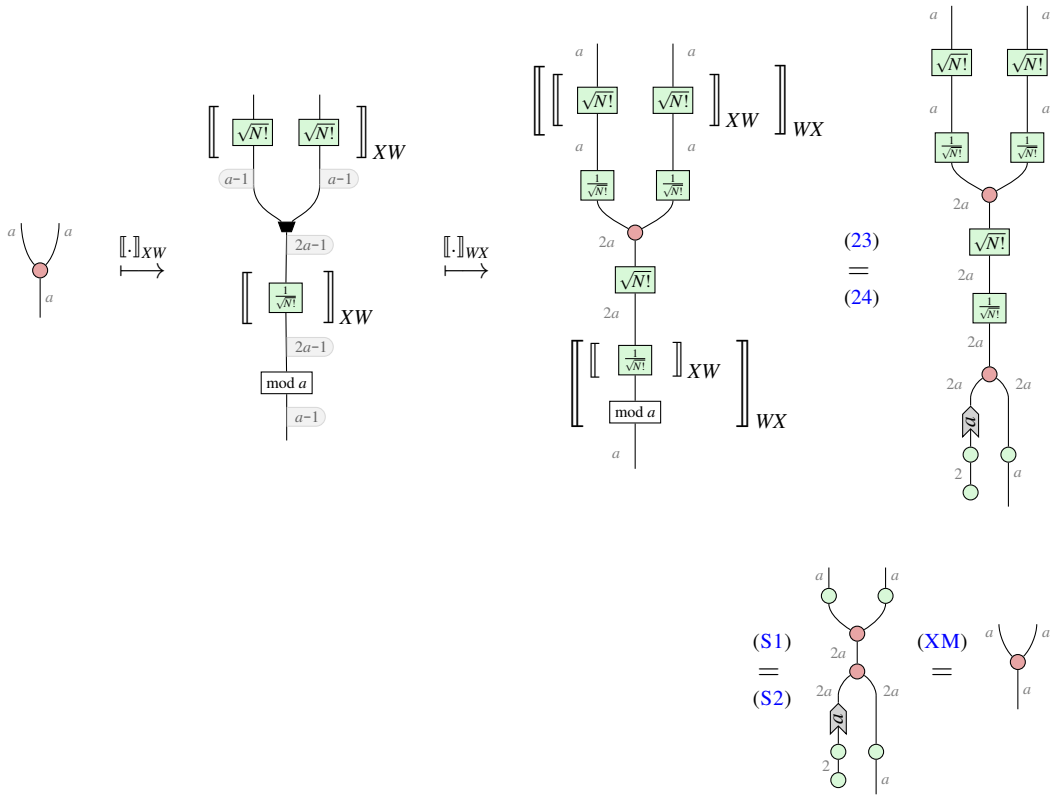


□

Lemma 25.

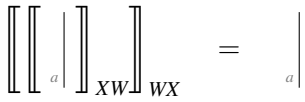


Proof.



□

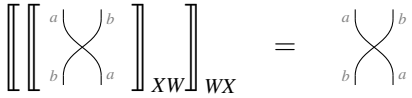
Lemma 26.



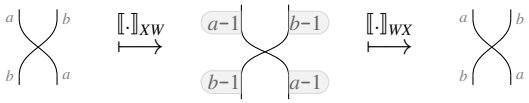
Proof.

□

Lemma 27.



Proof.



□

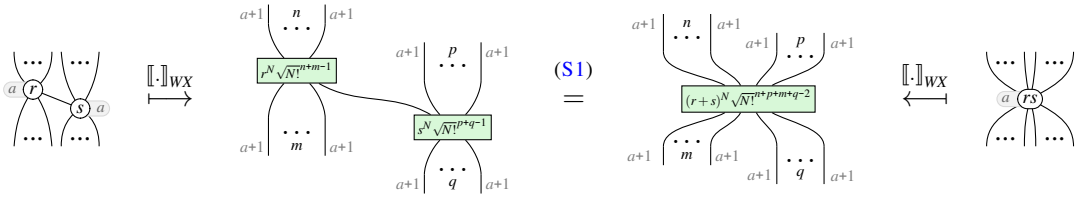
Lemma 3. *For an arbitrary ZX-diagram D, $\mathbf{ZX_f} \vdash [\![[\![D]\!]_{XW}]\!]_{WX} = D$.*

Proof. Due to the functoriality of the monoidal functors $[\![.]\!]_{WX}$ and $[\![.]\!]_{XW}$, it suffices to show that the above lemma holds for all generators of $\mathbf{ZX_f}$. These generators are presented in Section 2.1, and the proofs are shown in Lemmas 20 to 27. □

A.3 Proving Axioms of ZW

Lemma 28. *The translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in $\mathbf{ZX_f}$:*

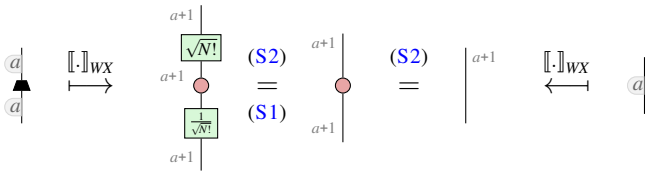
Proof.



□

Lemma 29. *The translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in $\mathbf{ZX_f}$:*

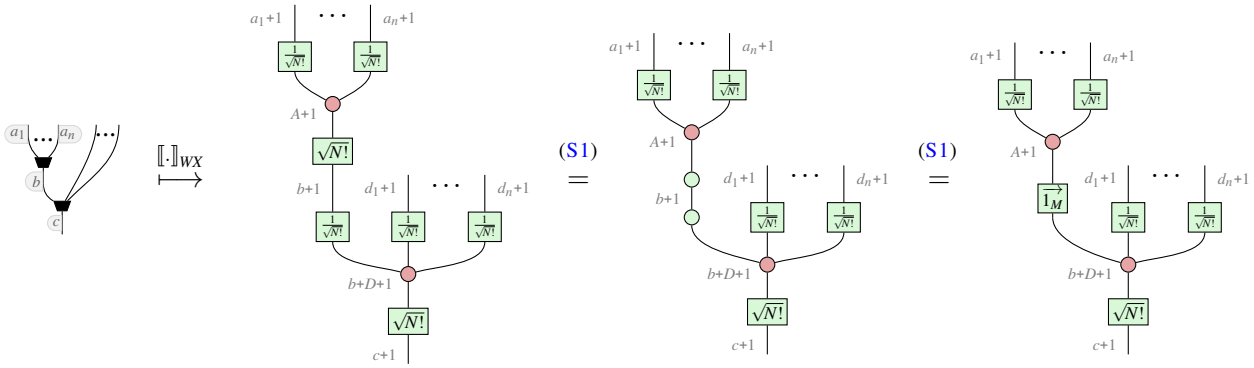
Proof.



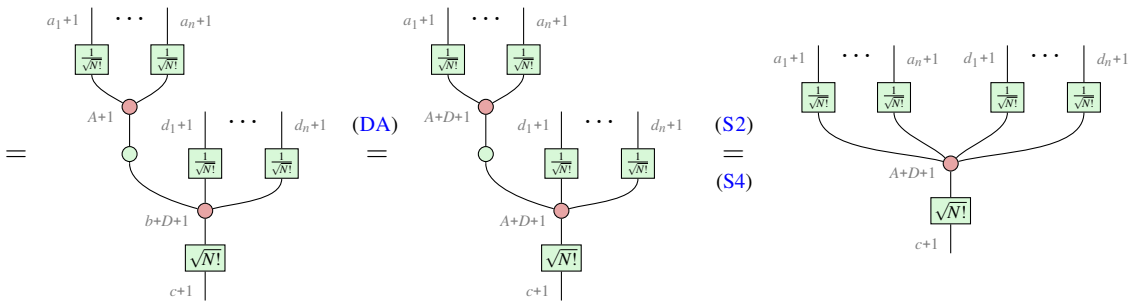
□

Lemma 30. *For $b \geq \min(c, \sum a_i)$, the translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in* $\mathbf{ZX_f}$*:*

Proof. For $A := \sum a_i$, $D := \sum d_i$, $M = \min\{A, b\}$:

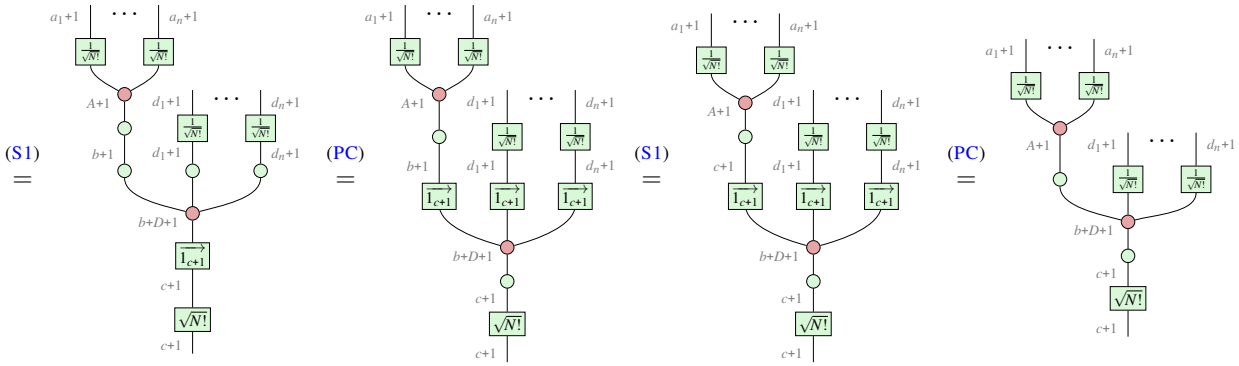


Here, we split the proof based on the value of M. If $A \leq b$, then $M = \min\{A, b\} = A$ and we have:



Otherwise, we have $A > b$, and thus $M = \min\{A, b\} = b$. Combining this with the assumption $b \geq$

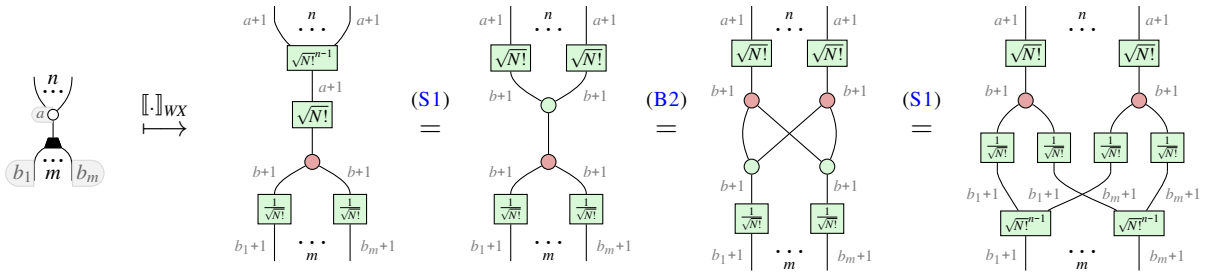
$\min(c, \sum a_i) = \min(c, A)$, we conclude that $b \geq c$. Therefore,



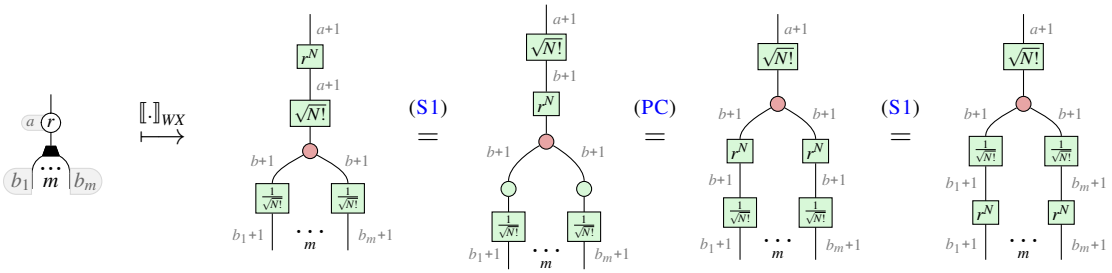
Then, the remaining steps are the same as in the $A \leq b$ case. □

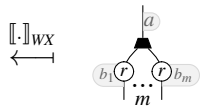
Lemma 31. *For $n \neq 0$, the translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in* $\mathbf{ZX}_\mathbf{f}$*:*

Proof. Let $b := \sum b_i$. We first show the following:



Now, we only have to show that the W-node copies phases:

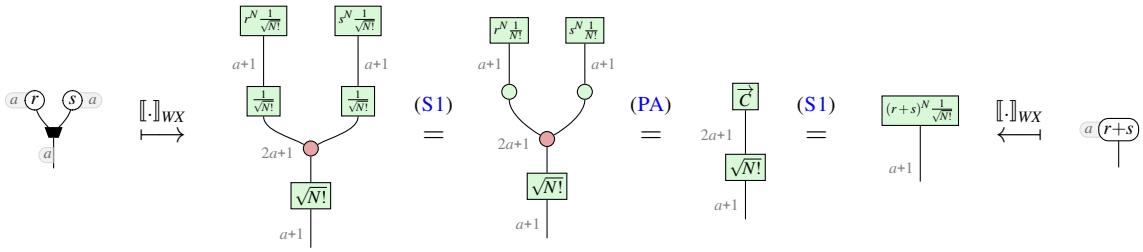




□

Lemma 32. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

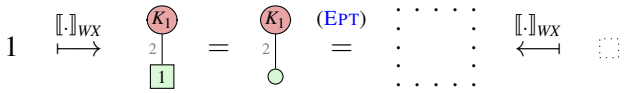
Proof.



where the k-th element of $\overrightarrow{C}$ for $k \leq a$ is given by $c_k = \sum_{i=0}^{k} r^i \frac{1}{i!} s^{k-i} \frac{1}{(k-i)!} = \frac{(r+s)^k}{k!}$ and for $k > a$ it is $c_k = \sum_{i=k-a}^{a} r^i \frac{1}{i!} s^{k-i} \frac{1}{(k-i)!}$; therefore, $\frac{(r+s)^N}{N!}$ defines the first $a+1$ element of $\overrightarrow{C}$. □

Lemma 33. *The translation of the global scalar* 1 *and the empty diagram* ⬚ *under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*.*

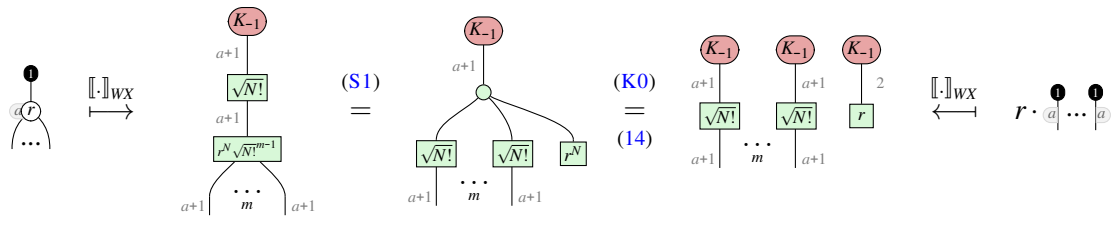
Proof.



□

Lemma 34. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

Proof.



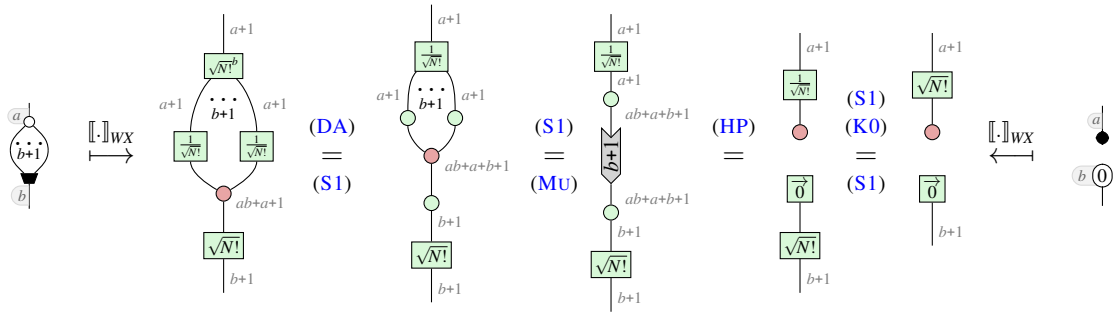
□

Lemma 35. *For $c \geq \min(\sum a_i, \sum b_i)$, $\ell_{ij} = \min(a_i, b_j)$, the translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in* $\mathbf{ZX_f}$*:*

Proof. This rule follows from Axiom (WW). □

Lemma 36. *The translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in* $\mathbf{ZX_f}$*:*

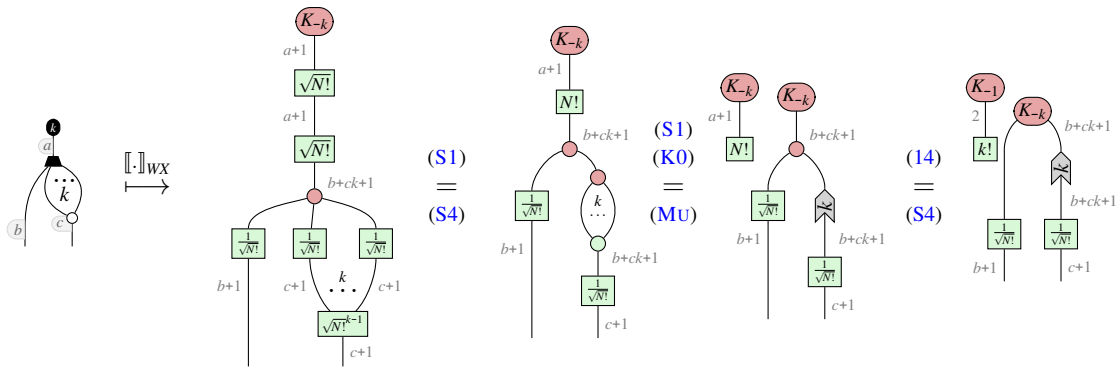
Proof.

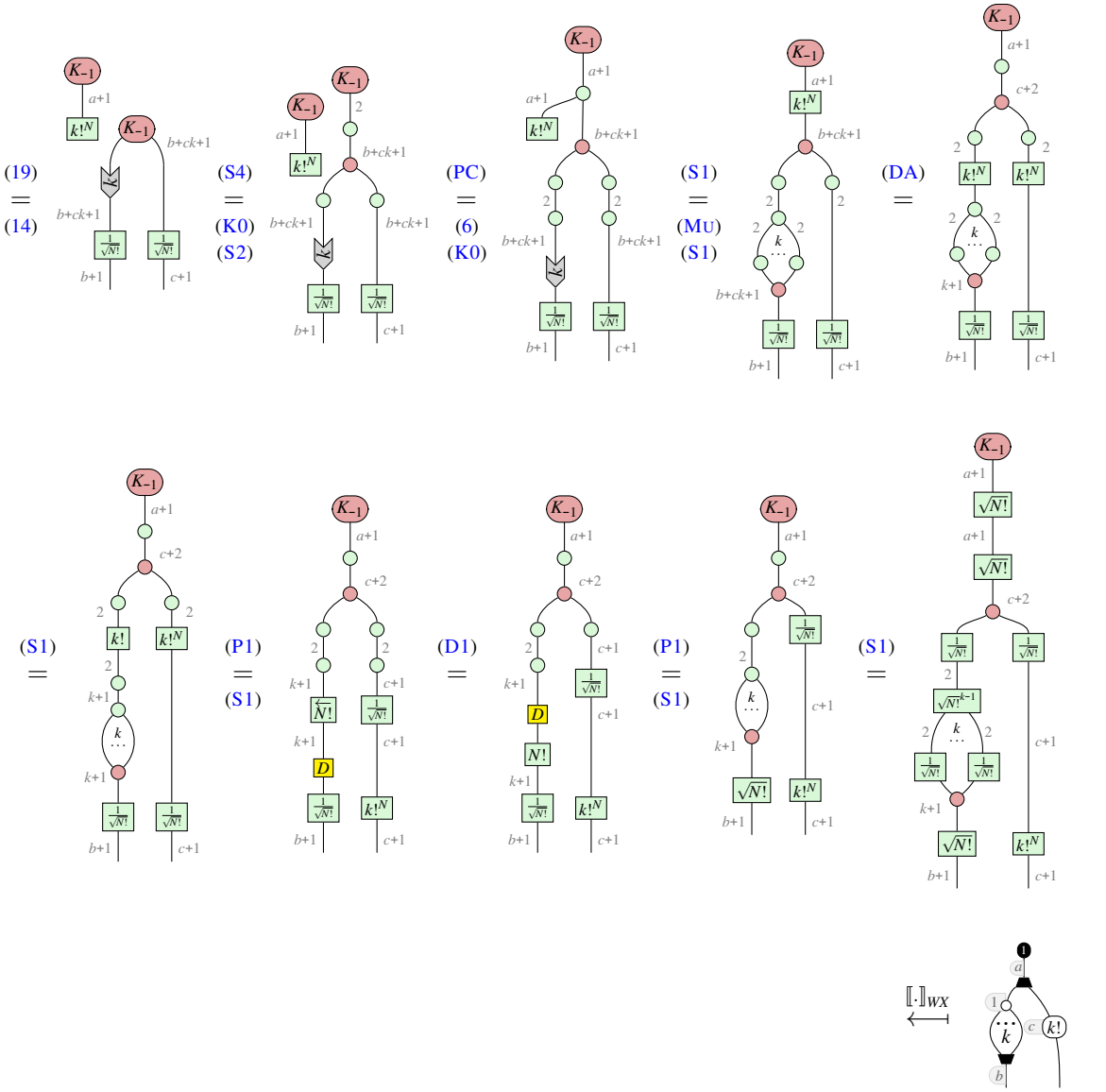


□

Lemma 37. *For $0 < k \leq a$, the translation of the following ZW diagrams under $[\![.]\!]_{WX}$ equal in* $\mathbf{ZX_f}$*:*

Proof.

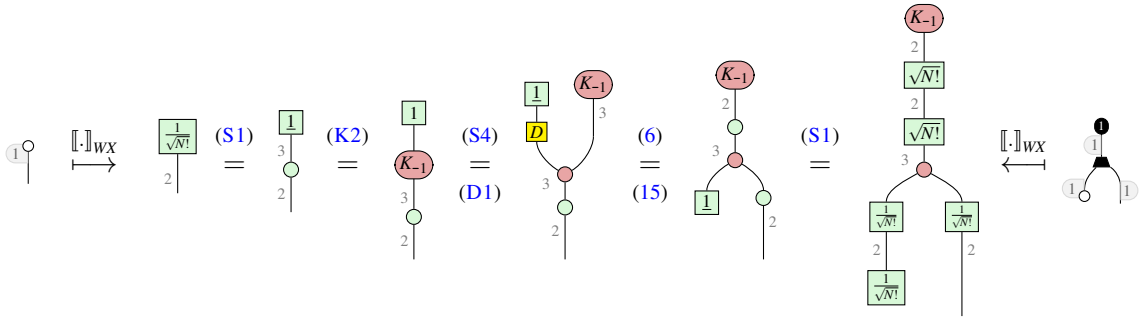




□

Lemma 38. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

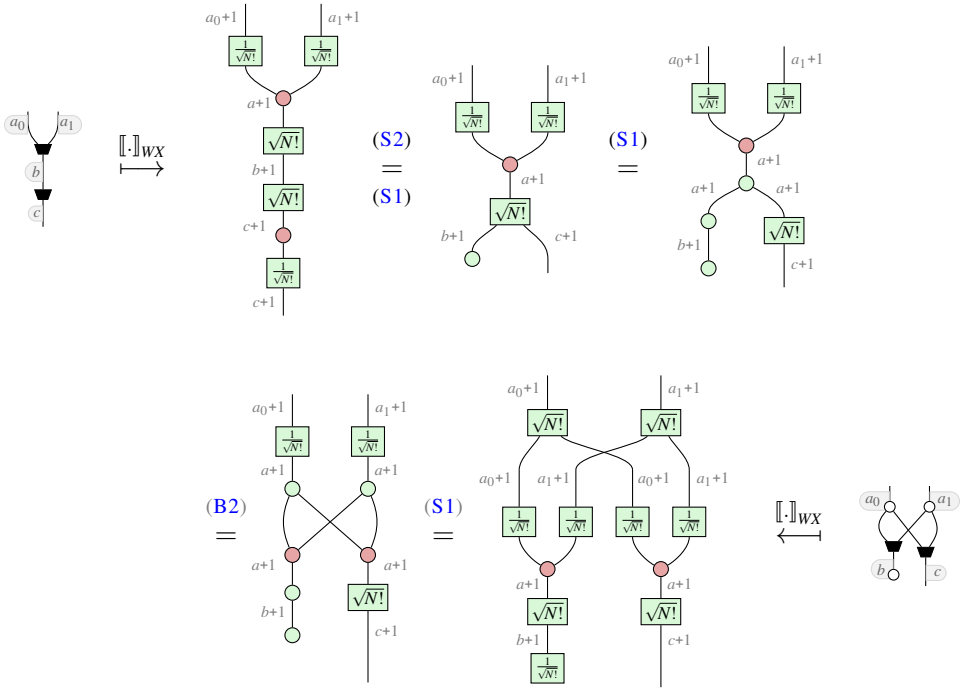
Proof.



□

Lemma 39. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

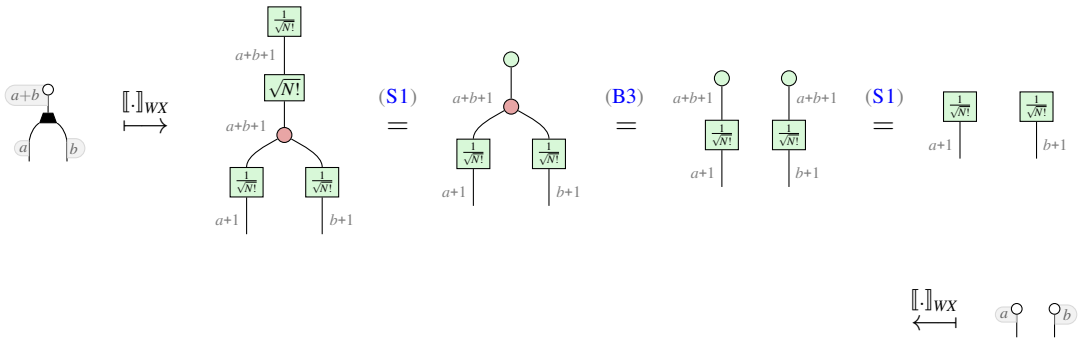
Proof. Let $a := a_0 + a_1$, then



□

Lemma 40. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

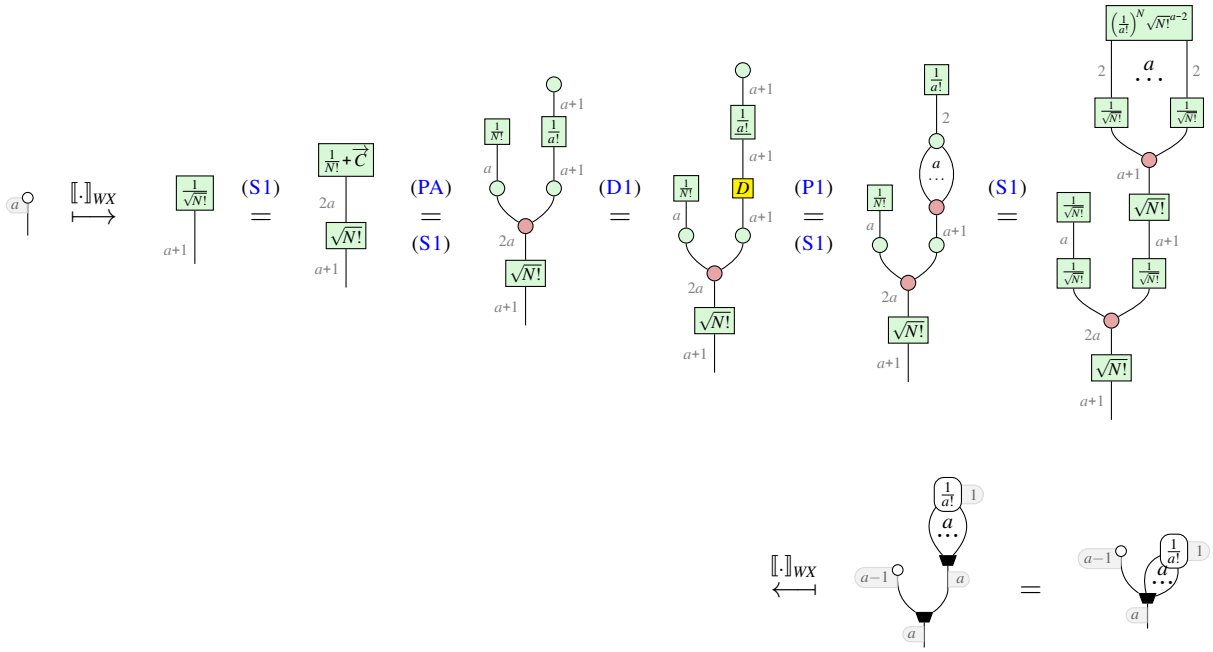
Proof. Let $a = a_0 + a_1$, then



□

Lemma 41. *The translation of the following ZW diagrams under* $[\![.]\!]_{WX}$ *equal in* $\mathbf{ZX_f}$*:*

Proof. For $\overrightarrow{C} = (\underbrace{0,\cdots,0}_{a+1}, \frac{1}{1!a!}, \frac{1}{2!a!}, \cdots, \frac{1}{(a-1)!a!})$,



Note that we can derive the last equality of the ZW diagrams using Lemma 30. □

Proposition 2. *If* $\mathbf{ZW_f} \vdash D_1 = D_2$ *then* $\mathbf{ZX_f} \vdash [\![D_1]\!]_{WX} = [\![D_2]\!]_{WX}$.

Proof. By the functoriality of $[\![.]\!]_{WX}$, we only need to show that all the rewrite rules of $\mathbf{ZW_f}$ are derivable in $\mathbf{ZX_f}$. The rewrite rules are given in Section 3.3, and the derivation of each axiom is given in Lemmas 28 to 41. □

String Diagrams for Defect-Based Surface Code Computing

Mateusz Kupper*
Dept. Informatics
University of Sussex
m.kupper@sussex.ac.uk

Dominic Horsman
Dept. Computer Science
University of Oxford
dom.horsman@gmail.com

Chris Heunen
School of Informatics
University of Edinburgh
chris.heunen@ed.ac.uk

Niel de Beaudrap
Dept. Informatics
University of Sussex
jrd27@sussex.ac.uk

Surface codes are a popular choice for implementing fault-tolerant quantum computing. Two-qubit gates may be realised in these codes using only nearest-neighbour interactions, either by lattice surgery [27] or by braiding defects around each other [43]. The effect of lattice surgery operations may be simply described [3] using the ZX-calculus: a graphical language that has proven effective for program design and optimisation (see e.g. [23]). In this work, we formalise a similar description via the ZX-calculus of defect braiding, as it is conventionally described. We define a graphical calculus **KNOT**, denoting the logical effects (in the absence of byproduct operations) of defect braiding in surface codes: we show how these effects may be described via a fragment of ZX-calculus which we call the $(0, \pi)$-fragment. We then use a 'doubling' construction to define a subtheory of **KNOT**, more specialised to standard encoding techniques in the defect braiding literature. Within this subtheory, we encompass standard braiding techniques by families of 'ribbon-like' and 'tangle-like' diagrams, each with semantics distinct from **KNOT**, in terms of the $(0, \pi)$-fragment of ZX diagrams (again in the absence of byproducts). These subtheories may be used interoperably, and are each sound and complete for the $(0, \pi)$-fragment of ZX diagrams. This provides a starting point to use the formal diagrammatics to analyse the operational effects of defect braiding procedures.

1 Introduction

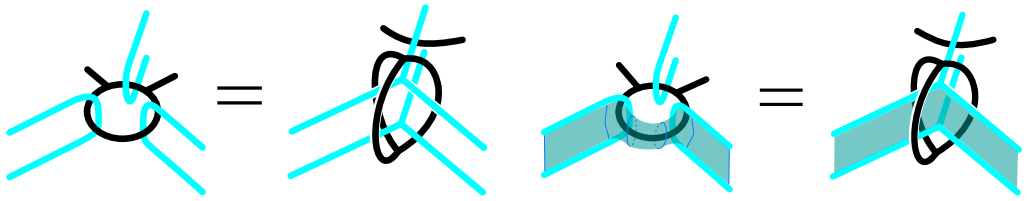
Topological quantum error correcting codes store quantum information in global properties of a system rather than in local degrees of freedom. The logical operators and the encoded space are determined by the topological features of an underlying lattice with boundary conditions. Surface codes are among the most widely studied quantum error correction codes due to their high threshold (tolerance to hardware error) and compatibility with multiple physical quantum computing paradigms [42].

Surface codes can be either planar, where individual 'patches' of physical qubits stabilised by the nearest-neighbour surface code operations support a single logical qubit each [13], or defect-based [42]. In the latter, logical qubits are supported by (usually) pairs of holes in the lattice of physical qubits, with multiple logical qubits sharing the same lattice 'patch'. Two-qubit entangling operations are performed using nearest-neighbour-only operations either by lattice surgery [27] for planar codes, or defect braiding for defect-based codes [43, 20]. Lattice surgery has lower resource requirements in general than defect-based surface codes (e.g. [23]). However, braiding remains an important potential tool for future surface code architectures and for the development of novel procedures using the code.

Logical information in the surface code is defined by logical operators: chains or rings of physical qubits whose joint state is the state of the logical operator. Extended surfaces of logical operators over time are known as 'correlation surfaces' (see e.g. [18, §3]). In the planar code, logical operators are strings between two opposite boundaries of the code patch, with the type of operator (X or Z) dependent on the type of boundary ('rough' or 'smooth'). In the defect-based code, logical operators are supported

*Corresponding author; author order is reverse-alphabetical, with the purpose of sowing confusion.

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 159–196, doi:10.4204/EPTCS.426.6



(a) (b)

Figure 1: Informal diagrammatic representations of braiding procedures using double defects. Different colours denote primal and dual; (b) shows also correlation surfaces. From Ref. [43].

between the boundaries created in the lattice by the pair of 'holes' (qubits removed from the stabilisation procedure). These also come in two flavours, 'primal' and 'dual', depending on the qubits removed.

Defect-based codes can be implemented as either 2D or 3D codes, homologically equivalent [43]. In 2D codes, physical qubits persist and are operated on over time; in 3D, they are prepared in an entangled state and then layers are measured out to create 'clock ticks' of the code. Intuitively, one can think of the 3D code as simply the 2D code represented in spacetime. Another homological equivalence [42] is between double and single defect codes. In single defects, the role of the second defect is taken by the boundary of the entire surface (in practice limiting to two logical qubits per surface). Figure 3 shows how the logical operators for the planar, single defect, and double defect codes are supported across their respective surfaces. Planar and defect-based qubits can even be interconverted, by judiciously increasing or measuring out the stabilised surface [27, §5].

Two-qubit operations perform interactions of logical operators and correlation surfaces. In lattice surgery, planar code X or Z operators are merged or split by straightforward operations that cut or combine operators from different patches. Braiding, however, interacts logical operators by moving defects around each other (either moving single defects around each other, or one of a double-defect pair moving through the space between the pair of another logical qubit). This causes the correlation surfaces to interact, performing a unitary two-qubit entangling gate (usually CNOT) provided the qubits are supported by pairs of opposite-type defects (primal and dual).

Braiding is clearly a more challenging operation to visualise than lattice surgery, not least being inherently 3D. Diagrams such as Figure 2 show a simple subroutine of six CNOT operations (similar diagrams also appear in [18, 19, 21]). Larger procedures quickly become impractical to represent visually, both in terms of human readability and computing power needed to render 3D diagrams. Such diagrams are informal: they are not supported by a standardised semantics and compositional formal language capable of equational reasoning, but are instead validated by checking correlation surfaces [43] or using a set of ad-hoc rewrite rules [37]. This is made more difficult by the use of a combination of 2D and 3D

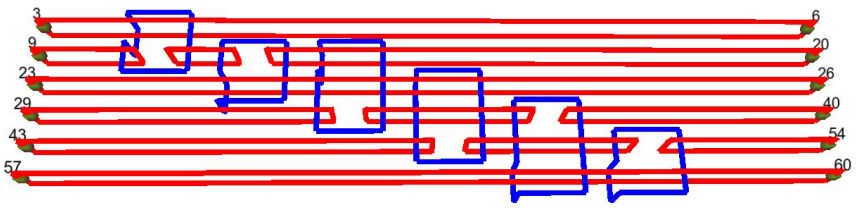
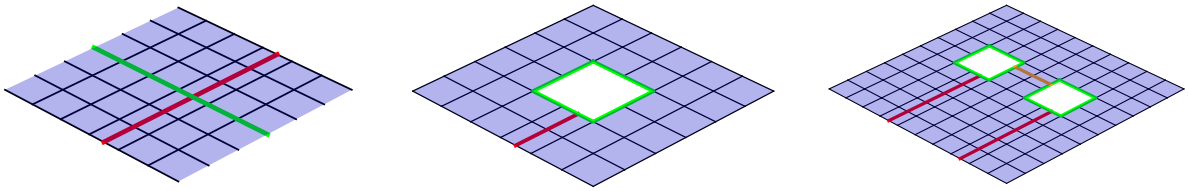


Figure 2: A three-dimensional representation of defect braiding. The red and blue structures represent dual and primal defects whose braiding implements a T gate. From Ref. [40].



(a) Surface code patch with two boundary types (rough - primal and smooth - dual) and two logical operators represented by red and green lines.

(b) Surface code with one boundary type with a defect with the same boundary type. One of the logical operators encircles the defect.

(c) Surface code with one boundary type with two defects with the same boundary types. The red logical operators are together equivalent to the orange operator.

Figure 3: We can introduce defects and modify boundaries in a surface code to change the observables encoded in the code. The observables (or also called logical operators or errors) are supported on the boundaries and defects. The lattices and logical operators in each figure encode a single logical qubit.

diagrams and the use of informal syntactic sugar [43] (also see Figure 1).

By contrast, lattice surgery benefits from a representation of its transformations in terms of ZX diagrams [3]. This representation provides a formal, systematic means of reasoning about lattice surgery procedures [22, 24]. Diagrammatic reasoning (especially ZX-calculus) has been used in quantum error correction for the analysis of more general stabiliser codes (e.g. [44, 34, 48, 4]) and surface codes [31]. There has been some limited work applying it to braided surface codes, where the correctness of *some* braiding patterns has been proven using ZX-calculus [26]. An informal mapping between different braiding patterns and ZX-calculus has also appeared in Ref. [24]. A further informal connection between lattice surgery, defect surgery, and ZX-calculus was made in Ref. [22]. A non-diagrammatic method for equivalence checks for braiding patterns using Boolean expressions appeared in Ref. [36].

This paper introduces a graphical language **KNOT**, to formalize defect-based surface code computation. We build upon the informal diagrams introduced originally by Raussendorf [43] and presented in numerous forms in Refs. [21, 19, 18] to define a formal language for defect braiding. We gather the rules appearing in these works and present them in a systematic way as a category with formal equivalences, and with our assumptions clearly stated. We also prove that the usual semantics appearing in the literature are sound (functorial) for ZX-calculus. In other words, by formalizing the largely intuitive and geometric arguments used in the literature, we provide a systematic account of defect braiding in the absence of byproduct operations. Furthermore, if we consider a pair of wires as encoding a qubit, we recover alternative semantics [43] which is sound and complete for the $(0, \pi)$-fragment of ZX-calculus.

This formal language for braiding patterns will facilitate automated verification and optimisation of defect-based protocols, contributing to the simplifying analysis of topological quantum computation.

Structure of the Paper. We proceed as follows. Section 2 gives brief background to braided surface codes. Section 3 introduces the the new formal framework for representing braiding operations using **KNOT**. Section 4 establishes different mappings into the $(0, \pi)$-fragment of the ZX calculus: from **KNOT**, and from two subtheories of a 'doubled' **KNOT**, which lets us demonstrate the equivalence between these two subtheories of **KNOT** and the $(0, \pi)$-fragment of ZX calculus.

2 Surface codes and defect braiding

A surface code lives on a square lattice whose edges carry physical qubits, see Figure 3 [6, 13, 42, 20, 19]. The code space is the $+1$-eigenstate subspace of all stabiliser operators. Two families of commuting stabilisers act locally: vertex (*star*) operators $A_v = \prod_{e \in \text{star}(v)} X_e$ and plaquette operators $B_p = \prod_{e \in \partial p} Z_e$. Commutation ensures a consistent code space, with

$$[A_v, A_{v'}] = [B_p, B_{p'}] = [A_v, B_p] = 0 \text{ for all } v, v', p, p'.$$

The code space is thus the subspace of the Hilbert space $\mathscr{H}$ where all stabilisers satisfy

$$A_v|\psi\rangle = |\psi\rangle, \quad B_p|\psi\rangle = |\psi\rangle \text{ for all } v, p.$$

Error detection is given by changes of these stabiliser measurements over time.

Defects are regions of the lattice where stabilisers are deliberately not measured, creating internal boundaries. This creates spare degrees of freedom in the lattice which are used to encode logical qubits. Smooth defects (or dual defects) correspond to missing A_v stabilisers, while rough defects (or primal defects) correspond to missing B_p stabilisers. Stabiliser patterns may change from a one measurement round to the next, so defects can appear, disappear, merge, divide, or move.

Logical operators are defined as paths (connecting boundaries) or loops of Pauli operators on the lattice (see Figure 3). For smooth defects, logical operators are $X_L = \prod_{e \in P_X} X_e$, $\quad Z_L = \prod_{e \in P_Z} Z_e$, where P_X is a path of X-operators connecting two smooth defects, and P_Z is a loop of Z-operators encircling a smooth defect. For rough defects, the roles of X and Z are reversed: X_L encircles a rough defect, and Z_L connects two rough defects. Logical operators correspond to chains of errors undetectable by the code. Boundaries also can be rough(smooth), supporting undetected $Z(X)$-chains.

The **dual lattice** has a vertex for each plaquette of the primal lattice and an edge for each primal edge. Stabilisers A_v on the primal lattice thus correspond to stabilisers B_p on the dual lattice, and logical operators map between the primal and dual lattices. This duality ensures symmetry between X- and Z-type errors, reinforcing the fault tolerance of the surface code.

Visualising **braiding** of logical operators is easiest in a spacetime view (although see also [20, §V] for clear step-by-step diagrams of a logical CNOT in the 2D view). We construct a 3D lattice, $\mathscr{L}$, with physical qubits placed on both the edges and faces of the lattice. This can be viewed as a *measurement-based quantum computation (MBQC)* simulation of the 2D surface code [42, 18], with the third lattice dimension as simulated time. The 3D cluster state is initialised by preparing qubits on the edges and faces of $\mathscr{L}$ in the $|+\rangle$ state and applying controlled-X gates between face qubits and edge qubits along their boundaries. Stabilisers for this state are given by:

$$K(c_2) = X(c_2)Z(\partial c_2) \quad \text{and} \quad K(\bar{c}_2) = X(\bar{c}_2)Z(\partial \bar{c}_2),$$

where $c_2 \in C_2$ corresponds to primal faces, and $\bar{c}_2 \in \bar{C}_2$ corresponds to dual faces. Logical qubits are encoded in defects that extend through the 3D lattice, and logical operators are defined in terms of these defects. For primal defects, the logical Z_L operator is the boundary of a dual 2-chain, $Z_L = Z(\partial \bar{c}_2)$, and the logical X_L operator is defined along a primal 1-chain c_1 as $X_L = X(c_1)$.

The 3D cluster is divided into three regions. The *defect region* ($D = D_p \cup D_d$) includes both primal defects (D_p) and dual defects (D_d) and represents areas where stabilisers are deliberately removed. The *vacuum region* ($V = V_p \cup V_d$) contains the rest of the lattice where stabilisers are intact, including primal (V_p) and dual (V_d) vacuum regions. Finally, the *singular qubit region* (S) includes qubits at the interface

of defects and vacuum. A specific measurement-based computation is performed based on four cluster areas. Qubits along defect chains ($d \subset D$) are measured in the Z-basis. Qubits on defect faces whose boundaries intersect d are measured in the X-basis. Qubits in the vacuum region (V) are measured in the X-basis. Finally, qubits in the singular region (S) are measured in the $\frac{X\pm Y}{\sqrt{2}}$-basis.

To ensure compatibility between the stabiliser structure and the measurement pattern, the stabilisers must commute with the measurements:

$$[K(c_2)K(\bar{c}_2), X_a] = 0 \text{ for all } a \in V, \quad [K(c_2)K(\bar{c}_2), Z_b] = 0 \text{ for all } b \in D.$$

These conditions imply that for any primal 2-chain $c_2 \in C_2$ and dual 2-chain $\bar{c}_2 \in \bar{C}_2$:

$$\{c_2\} \subset V_p, \quad \{\partial c_2\} \subset D_p, \quad \{\bar{c}_2\} \subset V_d, \quad \{\partial \bar{c}_2\} \subset D_d.$$

Within this 3D description, a primal defect carries $Z_L = Z(\partial \bar{c}_2)$ and $X_L = X(c_1)$, where $\partial \bar{c}_2$ encircles the defect and c_1 connects time-separated cross-sections. Braiding a smooth defect (dual defect) around a rough defect (primal defect) induces transformations of the logical operators as the stabilisers of the correlation surfaces are multiplied at the point of braiding and then separated again (analogous to the merge and splitting operations of lattice surgery):

$$X_L^{(\text{smooth})} \to X_L^{(\text{smooth})} X_L^{(\text{rough})} \quad \text{and} \quad Z_L^{(\text{rough})} \to Z_L^{(\text{smooth})} Z_L^{(\text{rough})}.$$

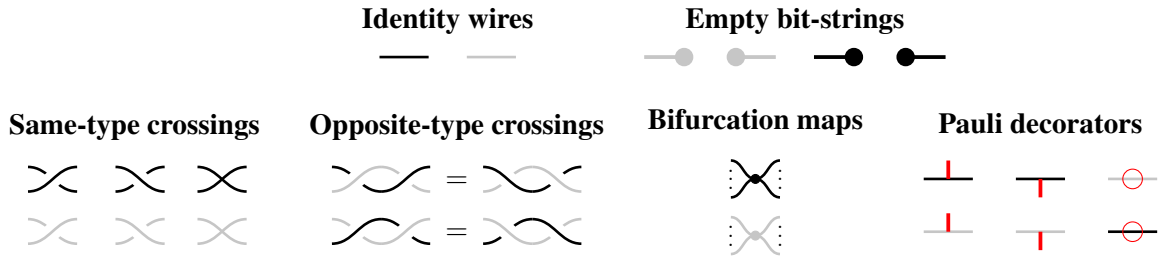
This corresponds to the application of a CNOT gate, where the smooth defect is the control qubit and the rough defect is the target qubit. Gates can be induced between logical qubits of the same type by introducing intermediate braids.[1]

Informal 'topological calculi' diagrams have been used since the beginning to illustrate the complex procedures of braiding, and to aid calculation of stabilisers [42, 43]. Later work used, again informally, computer-aided 3D diagram generation [22, 38, 40, 17]. However, reasoning about braids has remained cumbersome, and lacks complete formal tools for circuit design and compilation.

3 Braiding of defects as a graphical calculus: the category KNOT

We represent maps in **KNOT** by diagrams, composed from generators as below. These depict braiding of defects in a surface code, in the form of a 2D diagram with additional information for wires crossing over or under other wires, in the manner of knot diagrams.

Definition 3.1. The category **KNOT** (also called the *category of defects*) is a monoidal category whose objects are bit-strings, tensor product is given by the concatenation of bit-strings, morphisms are generated by the below diagrams and for domains and codomains of morphisms 0 is represented by a grey wire and 1 by a black wire.



[1] As every operation to merge, split, move, introduce, or remove defects involve changes to the stabilisers being measured, there will also be uncontrollable but *heralded* byproduct operations, of the sort familiar from MBQC [15, 46] and lattice surgery [3, 4]. Strictly speaking, these must be accounted for to describe general braiding operations as CPTP maps. However, we follow convention in acknowledging and then dropping explicit mention of them, as a first step in the development of a workflow for a more comprehensive and systematic approach to analysing defect braiding procedures.

Diagrams are read from left to right, representing a progression from the input to the output. A grey or a black dot on the right (left) end of a diagram represents an empty bit-string as a domain (codomain); similarly, cups and caps have empty bit-strings as domains or codomains. The generators satisfy the equivalences shown in Table 1.

Diagrams of **KNOT** represent the braiding of defects in a surface code, and correlation surfaces (spacetime stabilisers) supported on those defects.[2]

Correlation surfaces. In the literature, the red decorators above are often omitted as they represent operations unachievable by braiding and can be seen as undetectable errors. However, logical transformations from braiding can be described by rewriting diagrams to "move" red generators from input to output. Correlation surfaces in the surface code [42, 43] track the evolution of logical operators or errors as the code's topology changes. They geometrically encode how logical operators—initially represented by chains or loops—are deformed by boundary shifts, defect braiding, or lattice manipulations. This tracking constructs space-time stabilisers, summarizing logical transformations and showing how an initial operator, such as $X_{\text{in }1}$, evolves into $X_{\text{out }1}X_{\text{out }2}$ via a logical map M (the action of a CNOT gate).

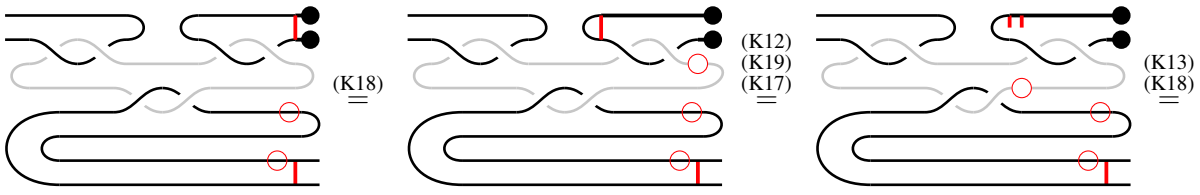
Soundness. A basic property of a rewrite system with a semantic map should be the soundness for its interpretation. The informal diagrams given in e.g. Ref. [43] can be interpreted as describing a specific family of surface code computation procedures described by a braiding pattern. The goal of such a procedure is to perform a map on the logical operators (or non-correctable errors) from the start to the end of the computation (to define a so-called space-time stabiliser). Thus, one can instead give an interpretation in the maps on the space of logical operators. This is what we also do in this paper. We are not concerned with the physical implementation of a specific braiding pattern as a set of operations on the surface code. We are instead interested in the space-time stabilisers that could be implemented by such procedures. We discuss this more in the Appendix B.

Theorem 3.1. KNOT is sound for the logical maps implemented by the braided surface code error correction procedure up to classical byproducts.

[Proof in Appendix B.]

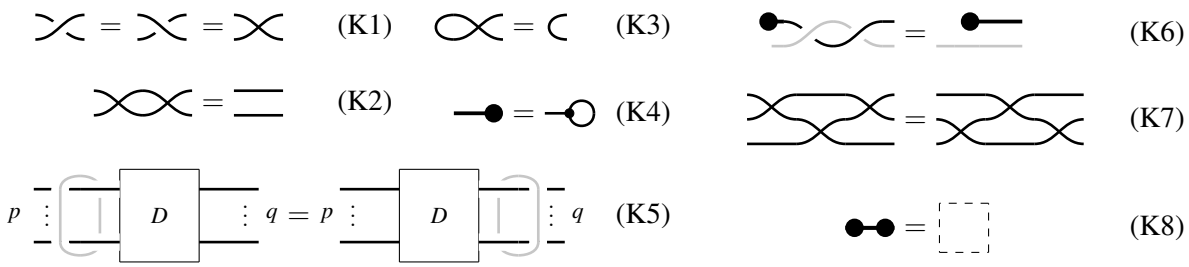
Different boundary conditions lead to different sets of equivalences for braiding. For all defects, we allow the surfaces to connect a defect to a boundary, encircle the defects, and connect two defects of the same type. A correlation surface connecting two defects of the same type, is equivalent to two independent correlation surfaces connecting the defects to a common boundary of the appropriate type; we freely use this equivalence to simplify our analysis.

Example 3.1. This example is an analogue of post-selected teleportation. In the next sections, we will explain how the generators can be associated with Hilbert spaces.

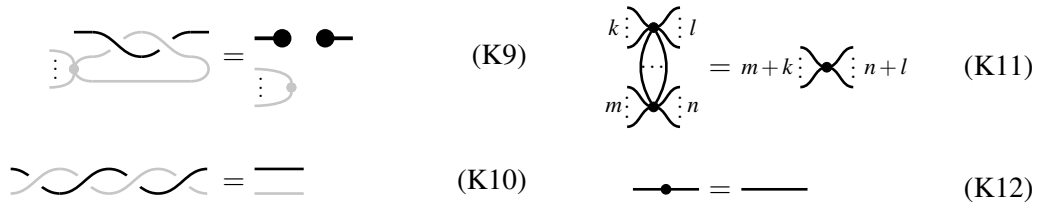


[2]The effect of those operations, supporting the equivalences of Table 1, are standard in the literature on defect braiding [43]. For the sake of brevity, we defer any account of these effects to our presentation of a map from **KNOT** into ZX-diagrams.

Planar isotopy rules:



Base rules:



Decorator rules:

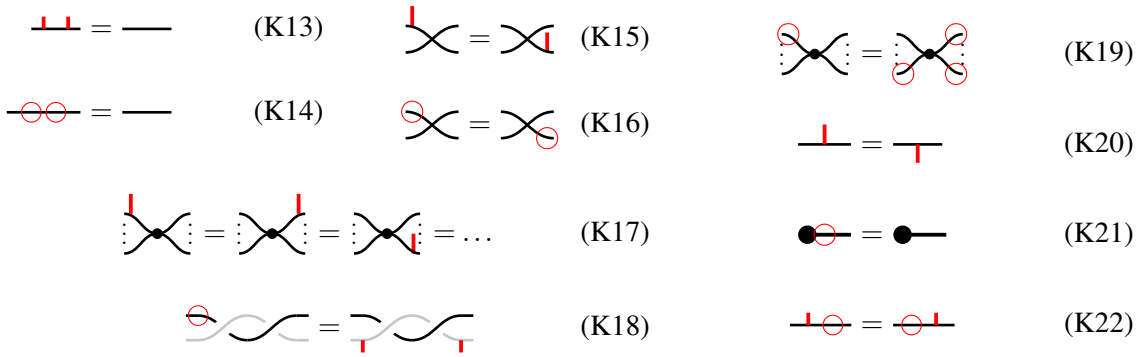
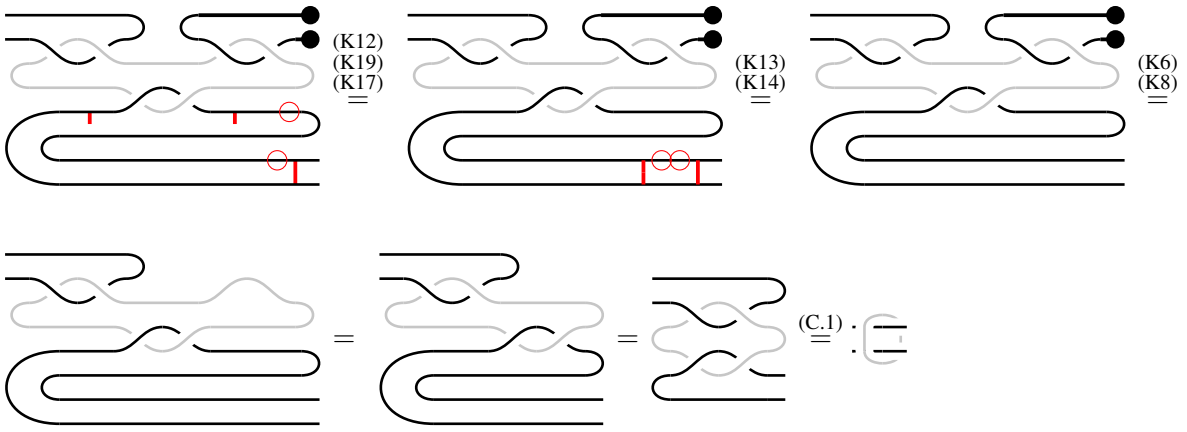


Table 1: The rules of the **KNOT** calculus. K5: D can be any diagram in **KNOT** without any red decorators and have arbitrary input and output wires of the same colour as the loops around the wires. We also require $p,q \in \{0,1,2\}$. *The planar isotopy rules* encode (some of) the rules of planar isotopy for opposite-type wires only. Same-type wires do not interact with each other. *The base rules* encode the equalities which are not covered by planar isotopy.



4 Interpreting the diagrams

In this section, we study the meaning of the **KNOT** diagrams and provide interpretations. We also show how, by applying a doubling construction we may recover the standard defect-pair encoding: in this setting, we describe two sublanguages generated by commonplace braiding techniques, and show that they are each sound and complete for the $(0, \pi)$-fragment of ZX.

4.1 A partial logical semantics of KNOT via the ZX-calculus

Thus far, we have only indicated that **KNOT** diagrams are intended to denote the logical effects of the corresponding braiding operations ignoring the byproduct operations. We have not said what these effects are, or proven the soundness of the rewrites (K1)–(K22): this sort of denotation is common in the literature, but relies on knowledge of what those effects are. One of our contributions is to describe the logical effect of braiding operations, by providing semantics for **KNOT** via ZX-diagrams. We prove the correctness of these semantics in the Appendix, but for now they serve as exposition.

We proceed by defining a map z from the generators of **KNOT** to ZX-diagrams as follows (where we group together generators which have the same semantics for brevity): we present this in Table 2.

Proposition 4.1. The map z lifts to a functor $Z : \mathbf{KNOT} \to \mathbf{ZX}$.

[Proof in Appendix A.1]

Corollary 4.1. **KNOT** is sound for the $(0, \pi)$-fragment of ZX-calculus via the functor Z.

Thus, we may interpret and analyse **KNOT** with reference to (a fragment of) the ZX-calculus, via the functor Z. Mappings similar to ours have appeared in the literature (e.g., [4, 26, 40, 24]), but we provide a formal proof that these rules are sound with respect to the maps they implement.

Remark. Similar expositions are often based on the assumption of an infinite surface code patch, which lacks boundaries (e.g., see [42, 43]). In contrast, we assume that the patch includes both boundary types at every time slice of the error correction procedure. This assumption introduces some changes to the semantics and syntax of the diagrams if we treat one defect as representing a qubit. If we treat a defect pair as representing a qubit, the usual semantics of the braided diagrams from the literature can be recovered as we will see below.

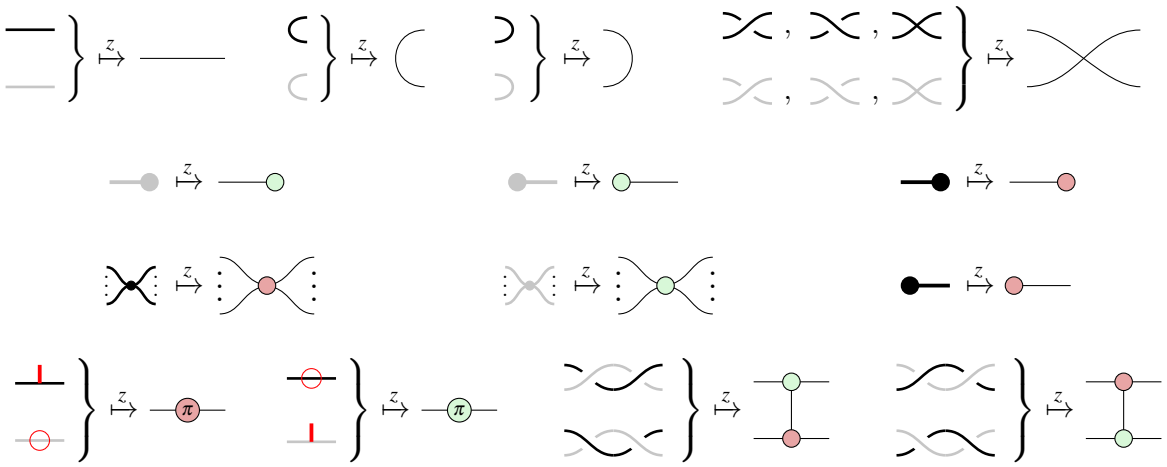
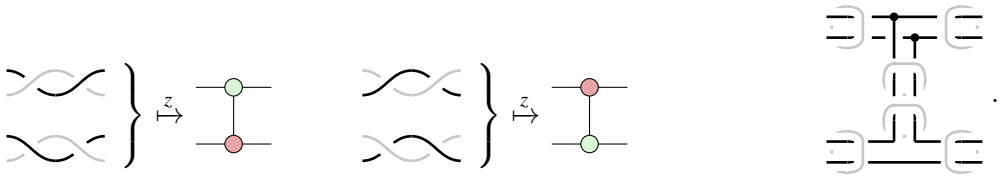


Table 2: Semantics for **KNOT** diagram generators, in terms of ZX diagrams. These semantics represent the effect of the corresponding braiding procedures (in the absence of byproduct operations), on qubits encoded in correlation surfaces. Note that, as a consequence, these maps are not necessarily norm-preserving.



(a) (b)

Figure 4: Two ways of performing a CNOT gate. (a) Using a single defect, the control is always a black wire and the target is always a gray wire. (b) A CNOT gate using a pair of defects – the gate can be performed in either direction.

4.2 Double defects

As we describe below, braiding individual defects to encode logical operations has some practical limitations. A standard remedy is to use a further 'defect pair' encoding for logical qubits [26, 42, 43, 39]. With a single wire representing a qubit, a CNOT gate can only be performed with a black 'primal' corresponding to the control qubit, and a gray 'dual' wire corresponding to the target, using the mapping z defined as in Table 2 (see Figure 4a).

A solution to this is offered in Ref. [43] (albeit motivated in a somewhat different context). This solution is to encode logical qubits in the logical operators of a pair of defects, and to realise CNOT operations in this encoding with a braiding procedure that we may denote in **KNOT** as shown in Figure 4b.

This construction enables CNOT gates to be performed in both directions, as the roles of control and target are encoded in the geometry of the procedure rather than the defect type. This technique requires a different encoding from what is described in Table 2, but which we may describe in relation to the semantics provided by the functor Z described above.

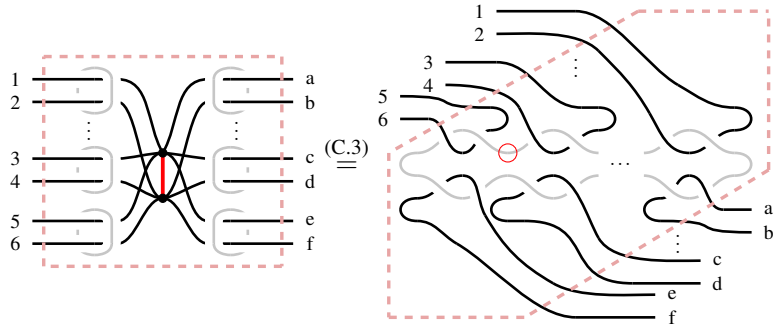
We now formalise and generalise this construction, in the setting of a finite lattice with external boundaries, by providing semantics for the doubled construction in the ZX-calculus in which pairs of

defects correspond to a single wire. The category is similar to CPM[**KNOT**] with the addition of cross-caps for arbitrary wires. In our subcategory, the black wires will be treated as 'qubit' wires, and the grey wires as mediating the interactions between the black wires. This corresponds to a convention in compiling procedures to defect braiding [36, 39, 40, 38, 37], in which 'primal' defects are the main defects and the dual defects serve only to help realise logical transformations.

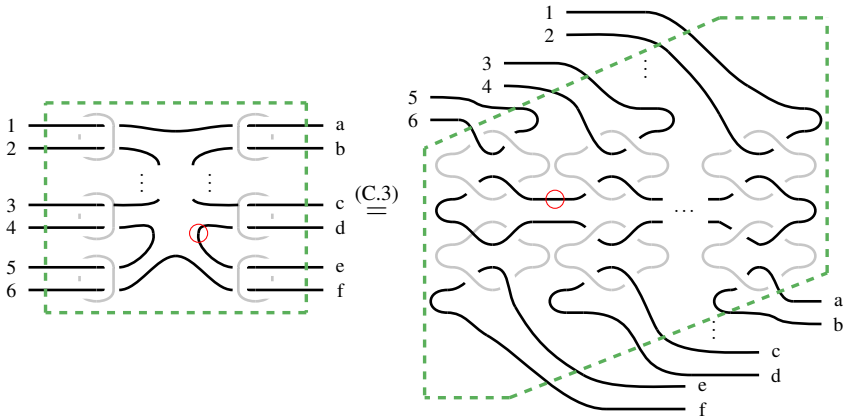
We refer to the structures of eqs. (1) and (2) on the LHS to be ribbon-like and the structures on the RHS to be tangle-like. Below, we describe two subtheories in which these different types of structures are generators, and show that each are complete for the $(0, \pi)$-fragment of ZX.

4.2.1 Tangle-like and ribbon-like structures

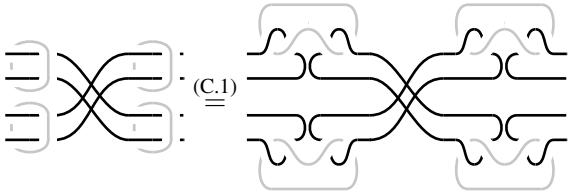
Definition 4.1. Define $\mathbf{KNOT}_{\text{doubled}}$ to be a subcategory of **KNOT** with morphisms generated by the following doubled diagrams:



(1)



(2)

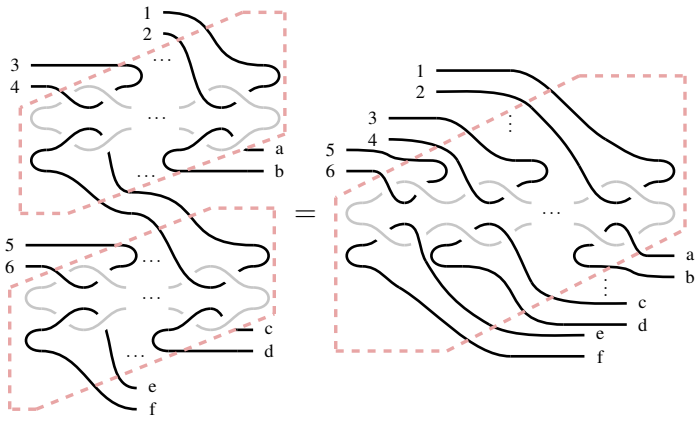


(3)

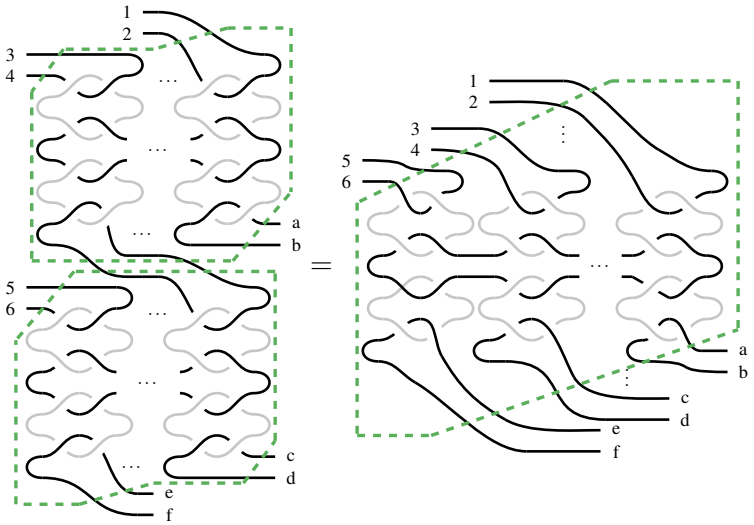
where the red decorators are optional and we also quotient by the precise placement of the red decorators on inner loops (whose exact position is negotiable through the rule (K18)). The numerical indices for the wires and the dashed lines are meant only to help the reader and are not a part of the theory. Two diagrams are considered equal if they can be transformed into one another by the rules of **KNOT** (possibly passing through immediate steps which are not in $\mathbf{KNOT}_{\text{doubled}}$).

All six of these generators are generalisations of patterns that are used to describe braiding procedures [43, 36]. However, it is possible to show (see Lemma C.3) using only rewrites of **KNOT** that the left-hand side generators can be rewritten into the right-hand side generators. By the soundness of **KNOT** for **ZX** through Z, we have that the ZX-diagrams corresponding to the diagrams on the right-hand side can be rewritten to the ZX-diagrams corresponding to the diagrams on the right-hand side. We can therefore reduce our analysis of this category to a treatment of the right-hand side generators; we do so below.

A number of interesting identities hold in $\mathbf{KNOT}_{\text{doubled}}$. For example, we have the following:



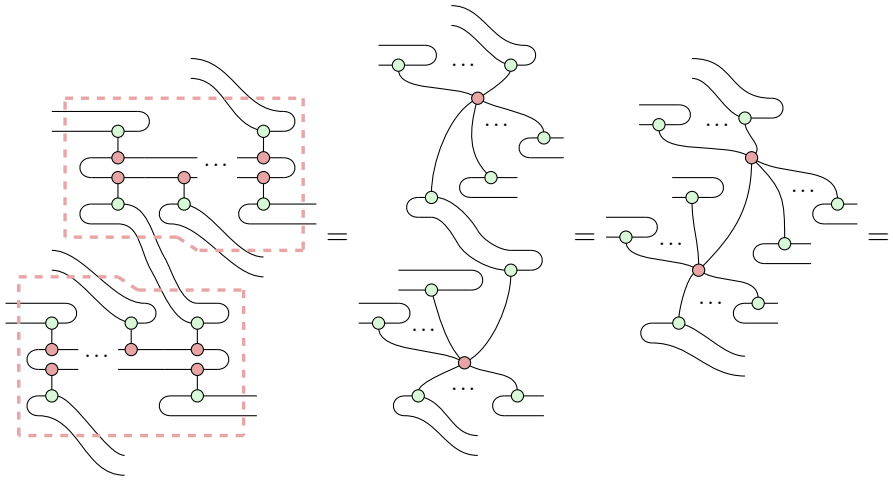
(4)



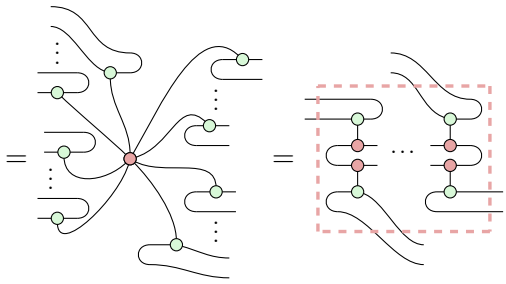
(5)

If we translate the LHS of eq. (4) via the functor Z, we obtain the following equations in the ZX-calculus.

The one below corresponds to the spider fusion law:

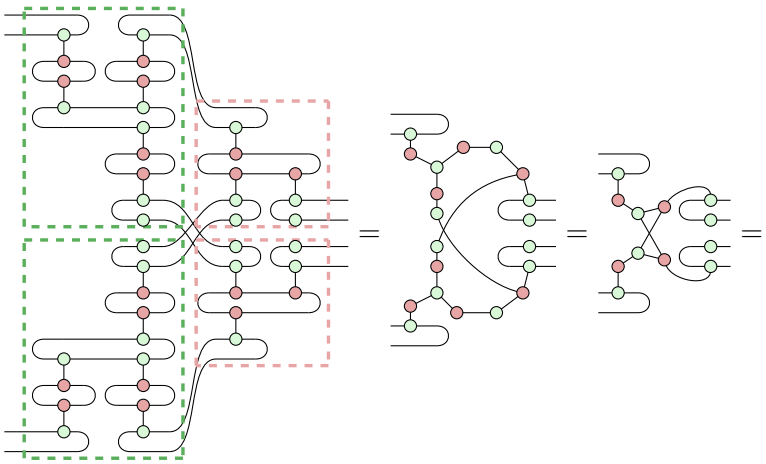


(6)

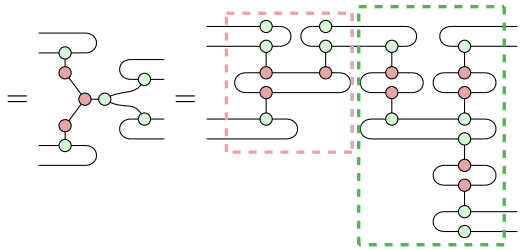


(7)

This is the equivalent of bialgebra law (for the **KNOT** diagram see eq. (34)):



(8)

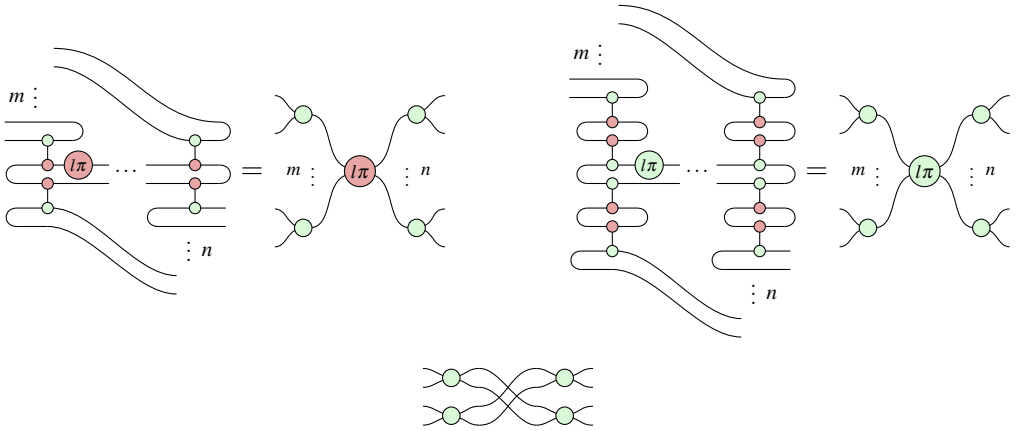


(9)

We give the full list of these identities in the Appendix (see Lemmas C.5-C.15). These lemmas might seems arbitrary but are crucial in observing the relationship between the ZX-calculus and $\mathbf{KNOT}_{\text{doubled}}$. In fact, eq. (34) is akin to the bialgebra law in the ZX-calculus, and the other equations are the spider and copy equations. The right-hand side of (28) is the green spider of ZX-calculus and the right-hand side of (29) is the red spider of ZX-calculus.

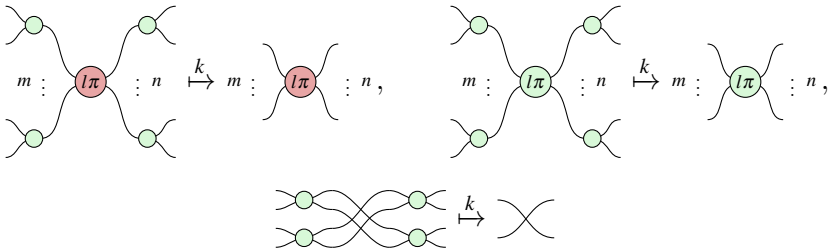
Definition 4.2. Let $\mathbf{ZX}_{\text{doubled}}$ be the category of the ZX diagrams in the image of the functor Z restricted to $\mathbf{KNOT}_{\text{doubled}}$.

Lemma 4.1. Explicitly, the morphisms of $\mathbf{ZX}_{\text{doubled}}$ are generated by:



Proof. Using the definition of the functor Z and by applying the spider fusion law of the ZX-calculus. □

Definition 4.3. Let $K : \mathbf{ZX}_{\text{doubled}} \to \mathbf{ZX}$ be the functor which acts as division by 2 on the objects and on morphisms it is defined by a map k:



The identity, cups, and caps can be obtained from a 2-legged spider with appropriate domain/codomain.

The functor K can be seen as a projection of a 2-qubit space, where the $|+\rangle$ is encoded by a Bell state, to a 1-qubit space. The "physical" maps acting on the 2-qubit space are each interpreted as maps acting on the "logical" 1-qubit space. We are simply using a 4-dimensional space to represent a qubit. This is the difference between the functor $K \circ Z$ (depicted in Table 3) and the functor Z – in effect a further level of encoding in which logical qubits are represented by (the correlation surfaces on) pairs of defects rather than single defects.

Proposition 4.2. We have the following correspondence between $\mathbf{KNOT}_{\text{doubled}}$ and $(0, \pi)$-ZX-calculus: C.5, C.6, C.7 and C.10 are the spider rules for both colours, eq. C.8 and C.9 are the copy rules for both colours, eq. C.11 is the bialgebra law, eq. C.12, C.13, C.14 and C.15 are the π-copy rules.

Proof. Convert to ZX-calculus using the functor Z and apply the spider fusion rule. □

Theorem 4.1. $\mathbf{KNOT}_{\text{doubled}}$ is sound and complete for the $(0, \pi)$-fragment of ZX-calculus.

[Proof in Appendix A.2].

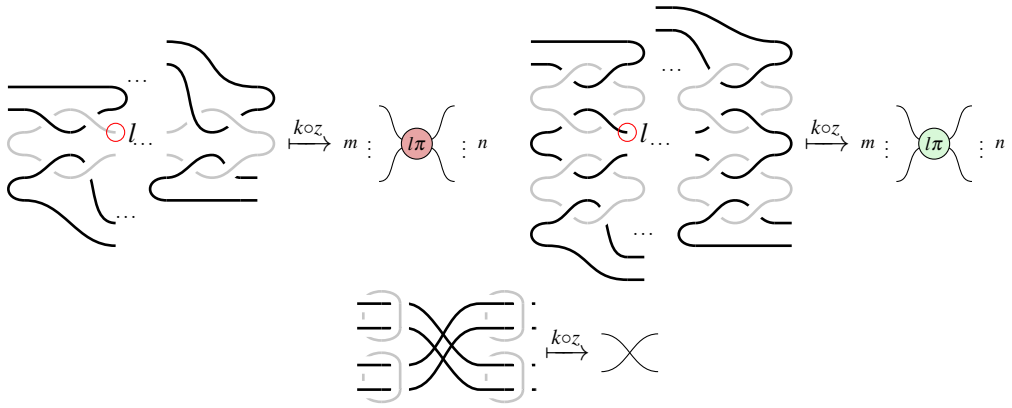


Table 3: We modify the semantics specified by the functor Z to account for the doubled structure of $\mathbf{KNOT}_{\text{doubled}}$. By binary l we done the presence or an absence of an operator. We make use of the functor K (acting on maps as k) to lift the semantics of **KNOT** to $\mathbf{KNOT}_{\text{doubled}}$.

5 Conclusions

In this work, we introduced **KNOT**, a graphical calculus for defect-based surface code computing. By leveraging this formalism, we provided a systematic framework for reasoning about defect braiding, bridging the gap between informal graphical approaches and algebraic descriptions. Furthermore, we extended this framework via a doubling construction that captures conventional encoding techniques in defect-based quantum computation, and demonstrated its equivalence to the $(0, \pi)$-fragment of ZX-calculus. We established soundness and completeness results, showing that both the tangle-like and ribbon-like sublanguages of **KNOT** faithfully represent $(0, \pi)$-ZX-diagrams. This formalisation enables automated verification and optimisation of defect braiding protocols, facilitating a deeper understanding of topological quantum computation. This also provides a formal way of understanding the semantics of braided surface codes.

We currently have partial results demonstrating how this framework can be used to reason about correlation surfaces in surface codes. In particular, we are working towards a more general diagrammatic calculus capturing the equivalences of correlation surfaces. The usual boundary conditions (an infinite lattice) yield a structure of a Hopf-Frobenius algebra where the correlation surface between a pair of defects satisfies the relations of a fragment of ZX-calculus.

Future work includes extending **KNOT** to incorporate byproduct operations and generalising its applicability beyond defect braiding with specific boundary conditions to broader classes of topological codes. The study of the connections between the homology of surface codes and different graphical calculi is also a promising direction for future research. Working towards capuring the homological equivalences for surfaces code for arbitrary topologies and boundaries using diagrammatics could help to build tools for compilation and verification for a broader class of surfaces codes. We also plan to explore the use of **KNOT** for optimising defect braiding protocols in practical quantum computing scenarios, including the development of automated tools for verifying and optimising defect-based quantum circuits.

Acknowledgements. We would like to thank Robert Booth for the feedback on an earlier draft of this paper. We would also like to thank Alexandru Paler for useful discussions on braided surface codes.

References

[1] Miriam Backens & Ali Nabi Duman (2016): *A complete graphical calculus for Spekkens' toy bit theory*. *Foundations of Physics* 46(1), pp. 70–103, doi:10.1007/s10701-015-9957-7.

[2] Miriam Backens, Simon Perdrix & Quanlong Wang (2017): *A Simplified Stabilizer ZX-calculus*. In Ross Duncan & Chris Heunen, editors: Proceedings 13th International Conference on *Quantum Physics and Logic*, Glasgow, Scotland, 6-10 June 2016, *Electronic Proceedings in Theoretical Computer Science* 236, Open Publishing Association, pp. 1–20, doi:10.4204/EPTCS.236.1.

[3] Niel de Beaudrap & Dominic Horsman (2020): *The ZX calculus is a language for surface code lattice surgery*. *Quantum* 4, p. 218, doi:10.22331/q-2020-01-09-218.

[4] Hector Bombin, Daniel Litinski, Naomi Nickerson, Fernando Pastawski & Sam Roberts (2024): *Unifying flavors of fault tolerance with the ZX calculus*. *Quantum* 8, p. 1379, doi:10.22331/q-2024-06-18-1379.

[5] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2017): *Interacting Hopf Algebras*. *Journal of Pure and Applied Algebra* 221(1), pp. 144–184, doi:10.1016/j.jpaa.2016.06.002.

[6] S. B. Bravyi & A. Yu. Kitaev (1998): *Quantum codes on a lattice with boundary*. Available at `https://arxiv.org/abs/quant-ph/9811052`.

[7] Daniel Cicala (2018): *Categorifying the ZX-calculus*. In Bob Coecke & Aleks Kissinger, editors: Proceedings 14th International Conference on *Quantum Physics and Logic*, Nijmegen, The Netherlands, 3-7 July 2017, *Electronic Proceedings in Theoretical Computer Science* 266, Open Publishing Association, pp. 294–314, doi:10.4204/EPTCS.266.19.

[8] Robin Cockett, Cole Comfort & Priyaa Srinivasan (2018): *The Category CNOT*. In: *Electronic Proceedings in Theoretical Computer Science*, 266, Open Publishing Association, pp. 258–293, doi:10.4204/eptcs.266.18.

[9] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.

[10] Cole Comfort (2019): *Circuit Relations for Real Stabilizers: Towards TOF+ H*. *arXiv preprint arXiv:1904.10614*, doi:10.48550/arXiv.1904.10614.

[11] Cole Comfort (2023): *The Algebra for Stabilizer Codes*. doi:10.48550/arXiv.2304.10584. arXiv:2304.10584.

[12] Cole Comfort & Aleks Kissinger (2022): *A Graphical Calculus for Lagrangian Relations*. In: *Quantum Physics and Logic*, *Electronic Proceedings in Theoretical Computer Science* 372, pp. 338–351, doi:10.4204/EPTCS.372.24.

[13] Eric Dennis, Alexei Kitaev, Andrew Landahl & John Preskill (2002): *Topological quantum memory*. *Journal of Mathematical Physics* 43(9), p. 4452–4505, doi:10.1063/1.1499754.

[14] Lucas Dixon & Ross Duncan (2008): *Extending Graphical Representations for Compact Closed Categories with Applications to Symbolic Quantum Computation*. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki & Freek Wiedijk, editors: *Intelligent Computer Mathematics*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 77–92, doi:10.1007/978-3-540-85110-3_8.

[15] Ross Duncan (2013): *A graphical approach to measurement-based quantum computing*. Oxford University Press.

[16] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. *Quantum* 4, p. 279, doi:10.22331/q-2020-06-04-279.

[17] Austin G Fowler & Simon J Devitt (2012): *A bridge to lower overhead quantum computation*. *arXiv preprint arXiv:1209.0510*, doi:10.48550/arXiv.1209.0510.

[18] Austin G. Fowler & Kovid Goyal (2009): *Topological cluster state quantum computing*. *Quantum Info. Comput.* 9(9), p. 721–738, doi:10.26421/QIC9.9-10-1.

[19] Austin G. Fowler, Matteo Mariantoni, John M. Martinis & Andrew N. Cleland (2012): *Surface codes: Towards practical large-scale quantum computation*. *Phys. Rev. A* 86, p. 032324, doi:10.1103/PhysRevA.86.032324.

[20] Austin G. Fowler, Ashley M. Stephens & Peter Groszkowski (2009): *High-threshold universal quantum computation on the surface code*. Physical Review A 80(5), doi:10.1103/physreva.80.052312.

[21] Keisuke Fujii (2015): *Quantum Computation with Topological Codes: from qubit to topological fault-tolerance*. 8, Springer, doi:10.1007/978-981-287-996-7_4.

[22] Craig Gidney (2020): *A Rosetta Stone for the ZX Calculus and the Surface Code*. ZX seminar. Available at `https://www.youtube.com/watch?v=1ojXEEm_JiI`.

[23] Craig Gidney & Austin G. Fowler (2019): *Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation*. Quantum 3, p. 135, doi:10.22331/q-2019-04-30-135.

[24] Michael Hanks, Marta P. Estarellas, William J. Munro & Kae Nemoto (2020): *Effective Compression of Quantum Braided Circuits Aided by ZX-Calculus*. Phys. Rev. X 10, p. 041030, doi:10.1103/PhysRevX.10.041030.

[25] Allen Hatcher (2002): *Algebraic Topology*. Cambridge University Press.

[26] Dominic Horsman (2011): *Quantum picturalism for topological cluster-state computing*. New Journal of Physics 13(9), p. 095011, doi:10.1088/1367-2630/13/9/095011.

[27] Dominic Horsman, Austin G Fowler, Simon Devitt & Rodney Van Meter (2012): *Surface code quantum computing by lattice surgery*. New Journal of Physics 14(12), p. 123011, doi:10.1088/1367-2630/14/12/123011.

[28] Ali Javadi-Abhari, Pranav Gokhale, Adam Holmes, Diana Franklin, Kenneth R. Brown, Margaret Martonosi & Frederic T. Chong (2017): *Optimized surface code communication in superconducting quantum computers*. In: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-50 '17, Association for Computing Machinery, New York, NY, USA, p. 692–705, doi:10.1145/3123939.3123949.

[29] André Joyal & Ross Street (1991): *The geometry of tensor calculus, I*. Advances in Mathematics 88(1), pp. 55–112, doi:10.1016/0001-8708(91)90003-P.

[30] Aleks Kissinger (2012): *Pictures of Processes: Automated Graph Rewriting for Monoidal Categories and Applications to Quantum Computing*. Ph.D. thesis, University of Oxford. DPhil Thesis.

[31] Aleks Kissinger (2022): *Phase-free ZX diagrams are CSS codes (...or how to graphically grok the surface code)*. doi:10.48550/arXiv.2204.14038. arXiv:2204.14038.

[32] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. In Bob Coecke & Matthew Leifer, editors: Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019, Electronic Proceedings in Theoretical Computer Science 318, Open Publishing Association, pp. 229–241, doi:10.4204/EPTCS.318.14.

[33] Aleks Kissinger & John van de Wetering (2022): *Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions*. Quantum Science and Technology 7(4), p. 044001, doi:10.1088/2058-9565/ac5d20.

[34] Aleks Kissinger & John van de Wetering (2024): *Scalable Spider Nests (...Or How to Graphically Grok Transversal Non-Clifford Gates)*. In Alejandro Díaz-Caro & Vladimir Zamdzhiev, editors: Proceedings of the 21st International Conference on Quantum Physics and Logic, Buenos Aires, Argentina, July 15-19, 2024, Electronic Proceedings in Theoretical Computer Science 406, Open Publishing Association, pp. 79–95, doi:10.4204/EPTCS.406.4.

[35] Alexei Y. Kitaev (2003): *Fault-tolerant quantum computation by anyons*. Annals of Physics 303(1), pp. 2–30, doi:10.1016/S0003-4916(02)00018-0.

[36] Alexandru Paler (2014): *Design methods for reliable quantum circuits*. Ph.D. thesis, Universität Passau.

[37] Alexandru Paler (2019): *SurfBraid: A concept tool for preparing and resource estimating quantum circuits protected by the surface code*. arXiv preprint arXiv:1902.02417, doi:10.48550/arXiv.1902.02417.

[38] Alexandru Paler & Simon J. Devitt (2018): *Specification format and a verification method of fault-tolerant quantum circuits*. Phys. Rev. A 98, p. 022302, doi:10.1103/PhysRevA.98.022302.

[39] Alexandru Paler, Simon J Devitt & Austin G Fowler (2016): *Synthesis of arbitrary quantum circuits to topological assembly*. Scientific reports 6(1), p. 30600, doi:10.1038/srep30600.

[40] Alexandru Paler, Ilia Polian, Kae Nemoto & Simon J Devitt (2017): *Fault-tolerant, high-level quantum circuits: form, compilation and description*. *Quantum Science and Technology* 2(2), p. 025003, doi:10.1088/2058-9565/aa66eb.

[41] Roger Penrose (1971): *Applications of negative dimensional tensors*. *Combinatorial mathematics and its applications* 1, pp. 221–244.

[42] Robert Raussendorf, Jim Harrington & Kovid Goyal (2006): *A fault-tolerant one-way quantum computer*. *Annals of Physics* 321(9), pp. 2242–2270, doi:10.1016/j.aop.2006.01.012.

[43] Robert Raussendorf, Jim Harrington & Kovid Goyal (2007): *Topological fault-tolerance in cluster state quantum computation*. *New Journal of Physics* 9(6), p. 199, doi:10.1088/1367-2630/9/6/199.

[44] Benjamin Rodatz, Boldizsár Poór & Aleks Kissinger (2024): *Floquetifying stabiliser codes with distance-preserving rewrites*. *arXiv preprint arXiv: 2410.17240*, doi:10.48550/arXiv.2410.17240.

[45] Peter Selinger (2007): *Dagger Compact Closed Categories and Completely Positive Maps: (Extended Abstract)*. *Electronic Notes in Theoretical Computer Science* 170, pp. 139–163, doi:10.1016/j.entcs.2006.12.018. Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005).

[46] Will Simmons (2021): *Relating Measurement Patterns to Circuits via Pauli Flow*. In Chris Heunen & Miriam Backens, editors: Proceedings 18th International Conference on *Quantum Physics and Logic*, Gdansk, Poland, and online, 7-11 June 2021, *Electronic Proceedings in Theoretical Computer Science* 343, Open Publishing Association, pp. 50–101, doi:10.4204/EPTCS.343.4.

[47] Robert W Spekkens (2007): *Evidence for the epistemic view of quantum states: A toy theory*. *Physical Review A—Atomic, Molecular, and Optical Physics* 75(3), p. 032110, doi:10.1103/PhysRevA.75.032110.

[48] Alex Townsend-Teague, Julio Carlos Magdalena de la Fuente & Markus S. Kesselring (2023): *Floquetifying the Colour Code*. In Shane Mansfield, Benoît Valiron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, QPL 2023, Paris, France, 17-21st July 2023*, *EPTCS* 384, pp. 265–303, doi:10.4204/EPTCS.384.14.

[49] Quanlong Wang (2018): *Completeness of the ZX-calculus*. Ph.D. thesis, University of Oxford.

A ZX-calculus

Graphical languages can be formalised as categories. One formulation is to treat the diagrams as morphisms and input/outputs or strings as objects (since diagrams represent processes which are modelled by morphisms) (e.g. see Ref. [8, 10, 5]). Other examples include formulating diagrams as a bicategory [7], algebraic [30, 41], topological [30, 29], and combinatoric [14] formulations. In particular, string diagrams are often represented as a **PROP** [49].

Definition A.1. A **PROP** is a strict symmetric monoidal category where every object is of the form

$$x^{\otimes n} = x \otimes x \otimes \cdots \otimes x \tag{10}$$

for a single object x and $n \geq 0$.

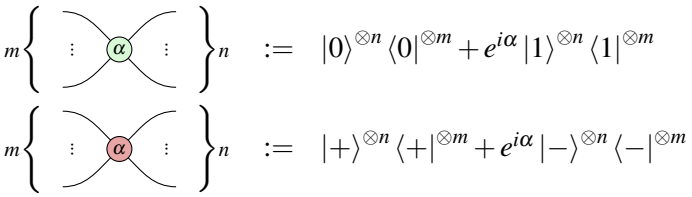
One can also consider a **PROP** as a category of natural numbers with addition as a tensor product. Since many graphical languages can be considered as a **PROP**, we use it to define maps between our languages as categories.

In considering the graphical languages, we follow the approach of [49]. For any diagrammatic system in this article, we have S the set of basic diagrams together with an empty diagram. As in this chapter we interpret the diagrams from left (inputs) to right (outputs), we can compose the basic diagrams horizontally ($\circ$) and vertically ($\otimes$). The diagrams satisfy a set of equivalence relations R. Thus, any category $\mathbf{C}$ having diagrams as morphisms based on a generator set S and satisfying the relations R is implicitly considered modulo R ($\mathbf{C}/R$) in this chapter. Any two morphisms are the same in $\mathbf{C}/R$ if their diagrammatic representations in $\mathbf{C}$ are equal according to R. This has a role in considering the mappings between different categories based on graphical languages. A mapping $F : \mathbf{A}/R_A \to \mathbf{B}/R_B$ between two categories $\mathbf{A}$ modulo relations R_A and $\mathbf{B}$ modulo relations R_B is functorial if we map the generators of $\mathbf{A}$ to $\mathbf{B}$ and if the relations R_A of $\mathbf{A}$ hold in $\mathbf{B}$ after the mapping F. This ensures that two diagrams equal in $\mathbf{A}$ (representing the same morphism in $\mathbf{A}/R_A$) are mapped to one morphism in $\mathbf{B}/R_B$ (that the functor F is well-defined).

A.1 ZX-calculus

In this work, we focus on a specific fragment of ZX-calculus where all the phases are either 0 and π. This is a subcategory of ZX-calculus similar to the calculi presented in Refs. [31]. This does not preserve some of the semantics of ZX-calculus in terms of **FHilb** (we do not have any rules involving zero scalars), but the rewrite rules are sufficient for tasks such as ZX-diagram simplification [32], circuit optimisation [16], stabiliser simulation [33]. Therefore, soundness and completeness we give is in terms of the "most common" rules of ZX-calculus.

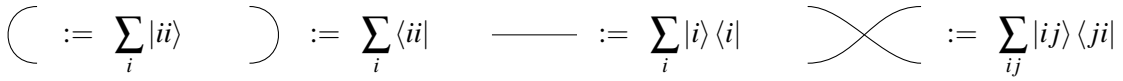
In ZX-calculus, there are two types of spiders: *Z spiders* and *X spiders*. These are defined with respect to the eigenbases of the Pauli Z and Pauli X operators, respectively.



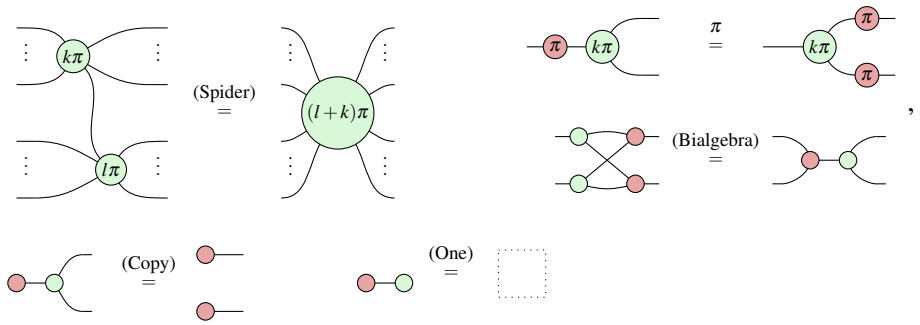
(11)

We usually omit phases if $\alpha = 0$ inside spiders as a convention.

In addition to spiders, we allow identity wires, swaps, cups, and caps in ZX diagrams, which are defined as follows:



Finally, we have the following rewrite rules for the $(0,\pi)$-fragment of ZX-calculus modulo scalars [31]:

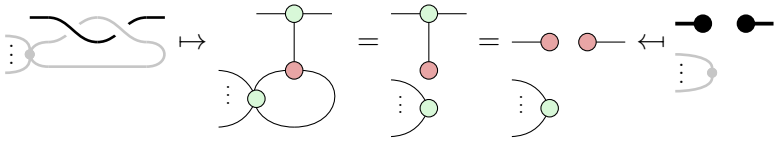


where l,k are in $\mathbb{F}_2$.

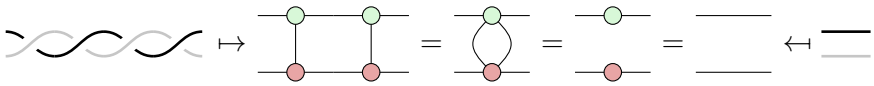
Note. The rules given above are non-standard: we ignore scalars completely and only allow the phases to be either 0 or π (which introduces the X, Z commutation rule). Strictly speaking, this category is actually the subcategory of the Spekkens' toy theory (**Spek**$_2$) [47] where the phase group is $\mathbb{Z}_2$ (if we only allow the spiders to be decorated with 11 as in the convention from [1] or alternatively with π,π following [9, §11.2.2]) – we will describe this relationship more in section B. Interestingly, even though this is a subcategory of **Spek**$_2$, because we do not have the problematic $\pm\frac{\pi}{2}$ (01, 10 from Ref. [1]) phases, this is also a subcategory of **Stab**$_2$ – we can define a functor from this category into **Stab**$_2$.

Proof of Proposition 4.1. We need to show that the rules of **KNOT** hold in ZX-calculus via the interpretation map z. We only explicitly prove the rules which require multiple steps. All other rules of **KNOT** can be proven to hold in ZX-calculus with the application of a single ZX-calculus rule.

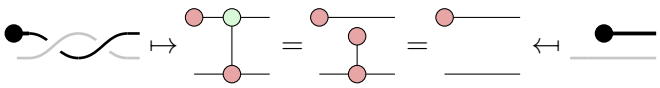
- Rule K9:



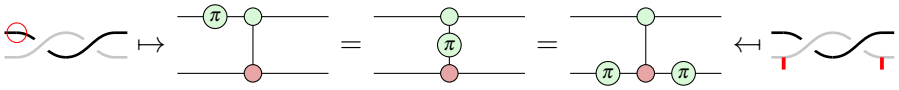
- Rule K10:



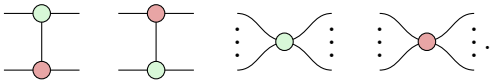
- Rule K6:



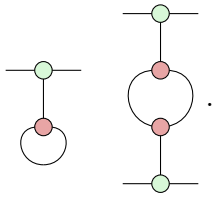
- Rule K18:



- Rule K5: the case for $p = 0$ and $q = 0$ is trivial. If we translate the subdiagram D into ZX-calculus, the resulting diagram will be constructed from the following generators:

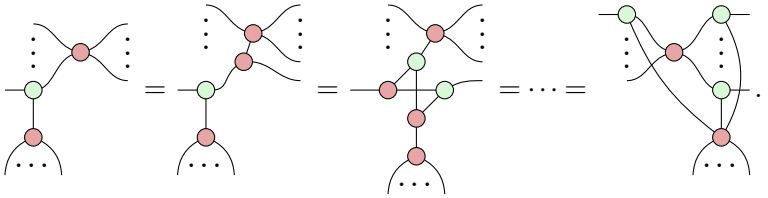


The loop translated in ZX-calculus will have the following two forms depending on p and q:



(*)

All  and  will be positioned such that the loops from (*) will commute through them (CNOTs are always positioned such that we can only get green spider on a wire corresponding to a black defect and red spider on a wire corresponding to a grey defect). For the spiders we show the following:



On the left hand side of the red spider, we might end up with green spiders (if the number of input wires is not one). We can however always remove the connections, since the wire might have a red spider connected to it (which would kill the connection) or (in **KNOT**) the equivalent of the loop might be connected to the wire. In this case applying the equation above would kill the connection by the Hopf rule. In the end, we are able to commute the equivalent of the loop through the diagram $z(D)$.

□

A.2 Soundness and completeness of $\mathbf{KNOT}_{\mathbf{doubled}}$ for the $(0, \pi)$-fragment of ZX-calculus

Lemma A.1. The diagrams of the category $\mathbf{KNOT}_{\text{doubled}}$ have a normal form.

Proof. To prove this we will use the availability of a normal form for $(0, \pi)$-fragment of ZX-calculus (modulo scalars).

First, for any given $\mathbf{KNOT}_{\text{doubled}}$ diagram, let's take a ribbon-like structure (LHS of eq. (1) and eq. (2)) and rewrite it to the corresponding tangle-like structure (RHS) for which we provided explicit definitions of $\mathbf{ZX}_{\text{doubled}}$ (Definition 4.1) and of the functor K (Definition 4.3).

Notice that the functor K is full and faithful. If we restrict the functor Z to $\mathbf{KNOT}_{\text{doubled}}$, the restricted functor $\widetilde{Z}$ is also full and faithful. This makes the composite $K \circ \widetilde{Z}$ a full and faithful functor from $\mathbf{KNOT}_{\text{doubled}}$ to $\mathbf{ZX}$.

(0, π)-fragment of ZX-calculus (modulo scalars) has a unique normal form. Let us consider two $\mathbf{KNOT}_{\text{doubled}}$ diagrams d_1 and d_2. If $K \circ \widetilde{Z}(d_1)$ and $K \circ \widetilde{Z}(d_2)$ reduce to the same normal form, then the d_1 and d_2 reduce to a corresponding $\mathbf{KNOT}_{\text{doubled}}$ diagram d_r (also by Proposition 4.2 $\mathbf{KNOT}_{\text{doubled}}$ is sound for (0, π)-fragment of ZX-calculus). We always stay in the image of the functor $K \circ \widetilde{Z}$.

In the other direction, if d_1 and d_2 are equal in $\mathbf{KNOT}_{\text{doubled}}$, then they are equal in $\mathbf{ZX}_{\text{doubled}}$ ($Z(d_1) = Z(d_2)$) using the soundness result of Proposition 4.1. This means that we also have $K \circ Z(d_1) = K \circ Z(d_2)$, which means that the ZX diagrams must have the same normal form, which by the fullness and faithfulness of the functor $K \circ \widetilde{Z}$ has a corresponding $\mathbf{KNOT}_{\text{doubled}}$ diagram d_r. □

Another way of seeing how this holds is by observing that all the rules of the (0, π)-fragment of ZX-calculus (modulo scalars) have an equivalent rule in $\mathbf{KNOT}_{\text{doubled}}$ by Proposition 4.2 ($\mathbf{KNOT}_{\text{doubled}}$ is sound for the (0, π)-fragment of ZX-calculus (modulo scalars)). (0, π)-fragment of ZX-calculus (modulo scalars) has a unique normal form which is obtained by using a rewrite system composed of a spider, bialgebra, and Hopf rule (which is provable from other rules). We can therefore use the same rewrite system in $\mathbf{KNOT}_{\text{doubled}}$ to obtain a $\mathbf{KNOT}_{\text{doubled}}$ diagram. At each stage of the rewrite procedure, the **KNOT** diagrams will also be $\mathbf{KNOT}_{\text{doubled}}$ diagrams. The resulting diagram will be a normal form for $\mathbf{KNOT}_{\text{doubled}}$ diagrams only with respect to the rules from Proposition 4.2. As all the rules in $\mathbf{KNOT}_{\text{doubled}}$ come from the rules of **KNOT**, we need to show that the normal form is a normal form with respect to the remaining rules of **KNOT**.

Suppose that we have two $\mathbf{KNOT}_{\text{doubled}}$ diagrams whose above normal forms are different (by the rules of Proposition 4.2). If we map these two normal forms into $\mathbf{ZX}_{\text{doubled}}$ via the functor Z, the resulting two diagrams must be different, but from the definition of the functor Z, the set of all the rules of **KNOT** is a subset of the set of all the equivalent rules of $\mathbf{ZX}_{\text{doubled}}$. Therefore, the two diagrams must also be different in $\mathbf{KNOT}_{\text{doubled}}$ according to the rules of **KNOT**.

Lemma A.2. All identities holding in $\mathbf{KNOT}_{\text{doubled}}$ via the rules of **KNOT** are derivable from the relations in Proposition 4.2.

Proof. This is a corollary of Lemma A.1. If two $\mathbf{KNOT}_{\text{doubled}}$ diagrams are equal according to the rules **KNOT**, they have the same unique normal form which is obtained using the rules from Proposition 4.2. □

Theorem A.1. $\mathbf{KNOT}_{\text{doubled}}$ via the interpretation $K \circ Z$ is sound and complete for the (0, π)-fragment of ZX-calculus.

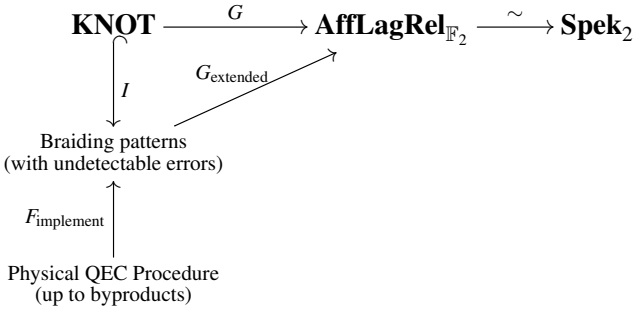
Proof. Follows from the discussion above. Proposition 4.2 shows that the rules of the (0, π)-fragment of ZX-calculus hold in $\mathbf{KNOT}_{\text{doubled}}$ and the Lemma A.2 shows that they are the only rules. □

B Soundness of KNOT for the evolution of logical operators

B.1 Overview

The maps implemented by surface code braiding procedures are defined by their space-time stabilisers. These stabilisers are defined by how the logical (undetectable) errors evolve with the computation. The logical operators in the surface code are described by first homology groups of the underlying topology of a surface code lattice. To deduce a map implemented by a procedure we must thus have a way of capturing how the first homology groups evolve alongside the computation. Our assumptions about the

lattice enable doing this with (affine) Lagrangian relations: a logical map implemented by a braiding procedure is fully characterised (modulo probabilistic byproducts) by an (affine) Lagrangian relation. The soundness property shown below is thus phrased in terms of the evolution of logical operators performed by *some* physical surface code procedures implementing a particular braiding pattern. The following "cartoon" commutative diagram summarises this reasoning:



Here, **Spek**$_2$ is the category of Spekkens' toy bit theory (for alternative presentations, see [1] and [9, §11.2.2]). We give the definition of the functor G in section B.4. We do not claim that **KNOT** captures all possible braiding patterns and equivalences between them, and I represents a functor into a bigger category of all such braiding patterns, containing **KNOT** is a subcategory. Affine Lagrangian relations over $\mathbb{F}_2$ are exactly the circuits belonging to the Spekkens' toy bit theory [1, 47].

As we will see, the image of the functor G does not have problematic phases causing a mismatch between **Spek**$_2$ and **Stab**$_2$ which is a subcategory of **Mat**$(\mathbb{C})$ generated by the qubit Clifford group, $\langle 0|$, and $|0\rangle$, modulo invertible scalars (for a graphical presentation, see [2]). We will therefore treat the image of the functor G as corresponding to morphisms of **Stab**$_2$.

B.2 Background preliminaries

B.2.1 Homological algebra

We first introduce some background from homological algebra. In particular, the tool of choice is the chain complex.

Definition B.1. A chain complex $C_\bullet$ in the category **C** is: a collection of objects $\{C_n\}_{n\in\mathbb{Z}}$, and of morphisms $\ldots \xrightarrow{\partial_3} C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} C_{-1} \xrightarrow{\partial_{-1}} \ldots$ $(\partial_n : C_n \to C_{n-1})$, such that $\partial_n \circ \partial_{n+1} = 0$ for all $n \in \mathbb{Z}$.

Definition B.2. Given a chain complex $C_\bullet$, let $Z_n(C_\bullet) = \ker(\partial_{\text{n}-1})$ and $B_n(C_\bullet) = \text{im}(\partial_n)$ which are called respectively n-cycles and n-boundaries. A quotient $H_n(C_\bullet) = Z_n(C_\bullet)/B_n(C_\bullet)$ is the nth homology group of $C_\bullet$.

A **homology group** provides an algebraic way to capture the structure of a chain complex by distinguishing between cycles and boundaries within it. For a given chain complex $C_\bullet$, the **nth homology group** $H_n(C_\bullet)$ is defined as the quotient group:

$$H_n(C_\bullet) = Z_n(C_\bullet)/B_n(C_\bullet),$$

where $Z_n(C_\bullet) = \ker(\partial_{n-1})$ denotes the **n-cycles**, which are elements in C_n that map to zero in C_{n-1} under the boundary map ∂_{n-1}, $B_n(C_\bullet) = \text{im}(\partial_n)$ denotes the **n-boundaries**, which are elements in C_n that arise as boundaries from C_{n+1} under the boundary map ∂_n.

Thus, the homology group $H_n(C_\bullet)$ identifies elements of $Z_n(C_\bullet)$ (cycles) that are not boundaries, capturing information about "holes" or "voids" in the structure represented by the chain complex.

Definition B.3. A chain subcomplex $D_\bullet$ of a chain complex $C_\bullet$ is a chain complex with the collection of objects D_n being the subobjects of C_n such that the following resultant square commutes:

$$\begin{array}{ccccccccc} \cdots & \longrightarrow & D_{n+1} & \xrightarrow{\partial_n^{D_\bullet}} & D_n & \xrightarrow{\partial_{n-1}^{D_\bullet}} & D_{n-1} & \longrightarrow & \cdots, \\ & & \downarrow{\scriptstyle f_{n+1}} & & \downarrow{\scriptstyle f_n} & & \downarrow{\scriptstyle f_{n-1}} & & \\ \cdots & \longrightarrow & C_{n+1} & \xrightarrow{\partial_n^{C_\bullet}} & C_n & \xrightarrow{\partial_{n-1}^{C_\bullet}} & C_{n-1} & \longrightarrow & \cdots \end{array}$$

The subcomplex inclusion map induced by the monomorphisms is called the chain map in the category of chain complexes. We call it the chain inclusion map.

Definition B.4. Let $C_\bullet$ be a chain complex and $D_\bullet$ its chain subcomplex. The quotient $C_\bullet/D_\bullet$ (C_n/D_n for all n) is the chain complex of $D_\bullet$-relative chains. The chain homology of the quotient is the $D_\bullet$-relative homology of $C_\bullet$: $H_n(C_\bullet, D_\bullet) := H_n(C_\bullet/D_\bullet)$.

In relative homology, we study cycles in a chain complex $C_\bullet$ that are allowed to "end" in a subcomplex $D_\bullet$. The **relative homology group** $H_n(C_\bullet, D_\bullet)$ captures information about cycles in $C_\bullet$ that are not boundaries within $D_\bullet$, thus describing features of $C_\bullet$ relative to $D_\bullet$.

Lemma B.1. Let $H_n(C_\bullet)$ be the nth homology group of a chain complex $C_\bullet$ over the field $\mathbb{F}_2$. Then $H_n(C_\bullet)$ is isomorphic to the vector space $\mathbb{F}_2^{\text{rank}(H_n(C_\bullet))}$, where $\text{rank}(H_n(C_\bullet))$ denotes the rank of the homology group, or the number of independent cycles in $H_n(C_\bullet)$.

Similarly, for a pair $(C_\bullet, D_\bullet)$ where $D_\bullet$ is a subcomplex of $C_\bullet$, the nth relative homology group $H_n(C_\bullet, D_\bullet)$ is isomorphic to $\mathbb{F}_2^{\text{rank}(H_n(C_\bullet, D_\bullet))}$.

Definition B.5. A cochain complex $C^\bullet$ in the category $\mathbf{C}$ is: a collection of objects $\{C^n\}_{n\in\mathbb{Z}}$, and of morphisms $\cdots \xrightarrow{d^{n-2}} C^{n-1} \xrightarrow{d^{n-1}} C^n \xrightarrow{d^n} C^{n+1} \xrightarrow{d^{n+1}} \cdots$, such that $d^{n+1} \circ d^n = 0$ for all $n \in \mathbb{Z}$.

Definition B.6. Given a cochain complex $C^\bullet$, let $Z^n(C^\bullet) = \ker(d^n)$ and $B^n(C^\bullet) = \text{im}(d^{n-1})$, which are called respectively n-cocycles and n-coboundaries. A quotient $H^n(C^\bullet) = Z^n(C^\bullet)/B^n(C^\bullet)$ is the nth cohomology group of $C^\bullet$.

A cohomology group provides an algebraic way to capture the structure of a cochain complex by distinguishing between cocycles and coboundaries within it. For a given cochain complex $C^\bullet$, the nth cohomology group $H^n(C^\bullet)$ is defined as the quotient group:

$$H^n(C^\bullet) = Z^n(C^\bullet)/B^n(C^\bullet),$$

where $Z^n(C^\bullet) = \ker(d^n)$ denotes the n-cocycles, which are elements in C^n that map to zero in C^{n+1} under the coboundary map d^n, and $B^n(C^\bullet) = \text{im}(d^{n-1})$ denotes the n-coboundaries, which are elements in C^n that arise as coboundaries from C^{n-1} under the coboundary map d^{n-1}.

B.2.2 Affine Lagrangian relations

The contents of this section are based on [12].

Definition B.7. The †-compact closed **PROP** of linear relations over $\mathbb{F}_2$, $\mathbf{LinRel}_{\mathbb{F}_2}$ is a category of natural numbers and linear relations (linear subspaces of $\mathbb{F}_2^n \oplus \mathbb{F}_2^m$ for natural numbers n, m) with relational composition, a direct sum of linear subspaces as the tensor product, and the relational converse as the dagger.

Definition B.8. Let $\mathbf{AffRel}_{\mathbb{F}_2}$ denote **PROP** whose morphisms $n \to m$ are the (possibly empty) affine subspaces of ${\mathbb{F}_2}^n \oplus {\mathbb{F}_2}^m$, with composition given by relational composition and tensor product given by the direct sum.

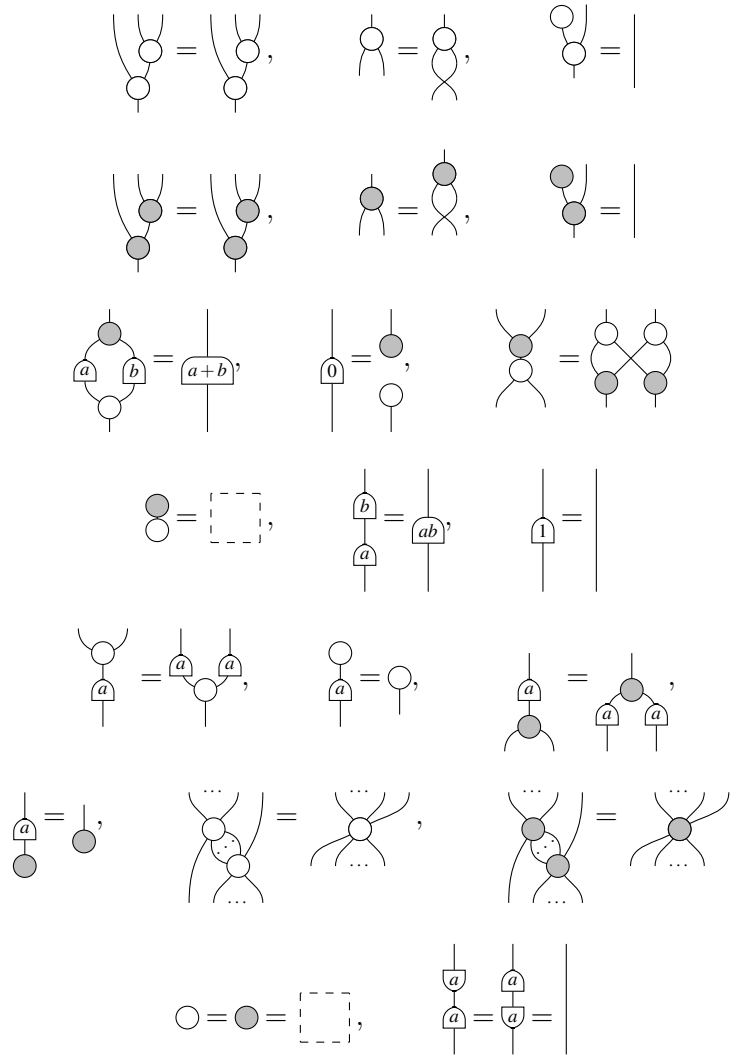
Definition B.9. Given a field $\mathbb{F}_2$, the **PROP** of **Lagrangian relations**, $\mathbf{LagRel}_{\mathbb{F}_2}$, has morphisms ${\mathbb{F}_2}^{2n} \to {\mathbb{F}_2}^{2m}$ as Lagrangian subspaces of the symplectic vector space ${\mathbb{F}_2}^{n+m} \oplus {\mathbb{F}_2}^{n+m}$ with symplectic form:

$$\omega := \begin{bmatrix} 0_n & I_n \\ I_n & 0_n \end{bmatrix}. \tag{12}$$

Lagrangian subspaces W are the subspaces, for which we have $W^\omega := \{v \in V : \forall w \in W,\, \omega(v,w) = 0\} = W$. Composition is given by relational composition and the tensor product is given by the direct sum.

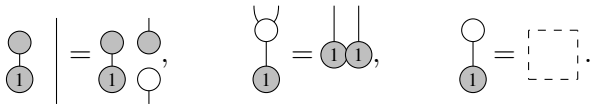
Definition B.10. Let $\mathbf{AffLagRel}_{\mathbb{F}_2}$ denote the **PROP** whose morphisms are the affine Lagrangian subspaces of ${\mathbb{F}_2}^{2n} \oplus {\mathbb{F}_2}^{2m}$ of some symplectic vector space ${\mathbb{F}_2}^{n+m} \oplus {\mathbb{F}_2}^{n+m}$.

Lemma B.2. Given a field $\mathbb{F}_2$, the **PROP** of matrices over $\mathbb{F}_2$ under the direct sum, $\mathbf{LinRel}_{\mathbb{F}_2}$, is presented by the following generators and equations, for $a, b \in \mathbb{F}_2$:

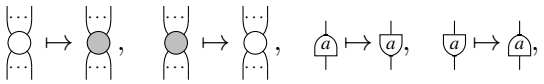


Definition B.11. The **PROP** of affine relations over $\mathbb{F}_2$, $\mathbf{AffRel}_{\mathbb{F}_2}$, is constructed in the same way as $\mathbf{LinRel}_{\mathbb{F}_2}$, except a map $n \to m$ is instead a (possibly empty) affine subspace of ${\mathbb{F}_2}^n \oplus {\mathbb{F}_2}^m$.

Lemma B.3. $\mathbf{AffRel}_{\mathbb{F}_2}$ is generated by adding an extra generator to $\mathbf{LinRel}_{\mathbb{F}_2}$ modulo the following equations:



Lemma B.4. The functor $(\_)^\perp : \mathbf{AffRel}_{\mathbb{F}_2} \to \mathbf{AffRel}_{\mathbb{F}_2}$ defined by:



we also have the orthogonal complement:

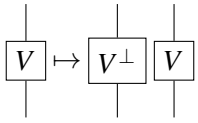
$$V \mapsto V^\perp := \{v \in V : \forall w \in V, \langle v, w \rangle = 0\}.$$

Affine relations are represented as affine subspaces that map between vector spaces, with objects given by natural numbers (indicating vector space dimensions) and morphisms as affine subspaces of ${\mathbb{F}_2}^n \oplus {\mathbb{F}_2}^m$. These subspaces define possible pairings between vectors in the input and output spaces, allowing for transformations with both linear and constant shift components.

In the category $\mathbf{AffRel}_{\mathbb{F}_2}$, composition is relational: given two affine relations, one from an n-dimensional space to an m-dimensional space and another from m to an ℓ-dimensional space, their composition is the set of pairs in ${\mathbb{F}_2}^n \oplus {\mathbb{F}_2}^\ell$ that can be linked by an intermediate vector in ${\mathbb{F}_2}^m$.

For example, a relation from ${\mathbb{F}_2}^2$ to $\mathbb{F}_2$ might be represented by the affine subspace $\{(x,y,z) \in {\mathbb{F}_2}^2 \oplus \mathbb{F}_2 : z = x + y + 1\}$, where the output is shifted by 1. Another relation from ${\mathbb{F}_2}^3$ to ${\mathbb{F}_2}^2$ could be given by $\{(x,y,z,w,v) \in {\mathbb{F}_2}^3 \oplus {\mathbb{F}_2}^2 : w = x + y, v = y + z\}$, with outputs as specific linear combinations of inputs.

Lemma B.5. There is a faithful, strong symmetric monoidal functor $L : \mathbf{LinRel}_{\mathbb{F}_2} \to \mathbf{LagRel}_{\mathbb{F}_2}$ given by the following action on the generators of $\mathbf{LinRel}_{\mathbb{F}_2}$ (i.e. doubling, and then changing the colours of one of the copies):



Lemma B.6. The forgetful functor $E : \mathbf{LagRel}_{\mathbb{F}_2} \to \mathbf{LinRel}_{\mathbb{F}_2}$ is faithful, strong symmetric monoidal.

Definition B.12. Let $\mathbf{AffLagRel}_{\mathbb{F}_2}$ denote the monoidal category whose objects are symplectic vector spaces with morphisms are generated by the image of $\mathbf{LagRel}_{\mathbb{F}_2} \xrightarrow{E} \mathbf{LinRel}_{\mathbb{F}_2} \to \mathbf{AffRel}_{\mathbb{F}_2}$ as well as all affine shifts and where the tensor product is the direct sum.

B.3 Homology within surface codes and Lagrangian relations

As discussed in the introduction, braiding defects (or, more generally, performing code deformations) in a surface code induces logical maps on the code space. From the standpoint of error correction, understanding these maps amounts to tracking how the logical operators are transformed under these braiding procedures. We assume the topology of the surface code to be such that we can define logical operators which come in two main flavors:

- Loops encircling one or more defects/boundaries, and
- Strings connecting pairs of defects or boundaries.

Other operators can exist (and most topologies of a surface code patch we will produce other operators), but we only focus on the two types in this work. From a homological perspective, the logical operators are the first homology (or cohomology for the dual lattice) groups on the underlying lattice seen as a chain complex [42]. Tracking the evolution of these groups is akin to specifying the Pauli webs or the spacetime stabilisers of maps. The operators and their action (and the first homology groups) at the start of the computation are equivalent to the operators at the end of the computation. From homological algebra, we know that:

$$H^1\big(C_\bullet;\mathbb{F}_2\big) \quad \text{and} \quad H_2\big(C^\bullet;\mathbb{F}_2\big) \tag{13}$$

are vector spaces over $\mathbb{F}_2$ where each basis element corresponds to a logical operator. This means that as we evolve the defects, we define a linear relation between the two $\mathbb{F}_2$-vector spaces of logical operators. For surface codes, this space is actually a direct product of vector spaces for primal and dual logical operators. In homological algebra, this becomes a direct product of spaces corresponding to the first homology and the second cohomology groups.

Assume there is an isomorphism between the first homology group and the second cohomology group by Lefschetz duality, that is, that our chain complex comes from an appropriate three-dimensional compact manifold[3]:

$$H^1(C^\bullet,\partial C^\bullet;\mathbb{F}_2) \cong H_2(C_\bullet;\mathbb{F}_2) \tag{14}$$

The intersection number between the operators supported on a defect is well-defined, because our assumptions guarantee a primal and dual operator supported on the same defect,

$$H^1(C^\bullet;\mathbb{F}_2) \times H_2(C_\bullet;\mathbb{F}_2) \to \mathbb{F}_2, \tag{15}$$

which evaluates to 1 if the two operators are supported on the same defect and 0 otherwise. This also captures the commutation relations, since X and Z operators anticommute if when supported on the same defect.

We treat our (co-)homology groups as vector spaces over $\mathbb{F}_2$. Denote the ambient vector space by

$$V = H_1 \oplus H_1^*. \tag{16}$$

Define the pairing as

$$\langle\cdot,\cdot\rangle\colon H \times H^* \to \mathbb{F}_2,$$

where H is the vector space of logical operators and H^* is the space of dual operators, and $\langle x,\beta\rangle$ is defined for vectors $x \in H$ and $\beta \in H^*$. In our case, it is given by $x^T\beta$. In other words, we compare the parities of dual and primal operators. This captures the commutation relations of operators defined on a surface code. This allows us to define a symplectic form on V by:

$$\omega_1\big((x,\alpha),(y,\beta)\big) = \langle x,\beta\rangle + \langle y,\alpha\rangle, \tag{17}$$

which is zero if the parities across two operators $(x,\alpha),(y,\beta)$ for dual and primal and primal components have the same parity.

[3]For more information on the Lefschetz and Poincaré duality see Section 3.3 of Ref. [25].

Let us now consider the situation where we attempt to map between operators at the beginning of some surface code procedure to the operators at the end. This is a linear relation between the spaces of logical operators on both "time-slices". For the second (final) time-slice, define:

$$V_2 = H_2 \oplus H_2^*. \tag{18}$$

The same construction (via Lefschetz duality and the Kronecker pairing) gives a symplectic form on V_2:

$$\omega_2\big((x,\alpha),(y,\beta)\big) = \langle x,\beta\rangle + \langle y,\alpha\rangle. \tag{19}$$

A linear relation (a linear subspace) $g\colon V_1 \to V_2$ can be defined component-wise:

$$g = g_p \oplus g_d : H_1 \oplus H_1^* \to H_2 \oplus H_2^*, \tag{20}$$

where

$$g_p\colon H_1 \to H_1, \quad g_d\colon H_1^* \to H_2^*.$$

Let $\Gamma(g)$ be the graph of the relation g, which is compatible with the duality pairings; in other words, it is equipped with a symplectic form

$$\Omega = \omega_1 \oplus (-\omega_2), \tag{21}$$

and we assume

$$\omega_1(v_1,w_1) \;=\; \omega_2(v_2,w_2) \quad \text{whenever} \quad (v_1,v_2),(w_1,w_2) \in \Gamma(g). \tag{22}$$

We want to find maximal such graphs (each logical operator on the initial state should be mapped to another logical operator on the final state if it possible). These assumptions and definitions can be interpreted within the context of logical operators in surface codes as enforcing of preservation of commutation relation as the computation progresses. Logical degrees of freedom are defined by the commutation relations and we do not want to alter the structure of the code, possibly merging distinct logical states or splitting one logical state into two. This would no longer describe "the same logical qubits," so one would not be performing a valid operation on the original code.

Consider $V_1 \oplus \overline{V_2}$, where $\overline{V_2}$ denotes V_2 equipped with the symplectic form $-\omega_1$. Since Lefschetz duality and the Kronecker pairing endow both V_1 and V_2 with symplectic forms (namely, ω_1 and ω_2, respectively), we have

$$\Omega\big((v_1,v_2),(w_1,w_2)\big) = \omega_1(v_1,w_1) - \omega_2\big(v_2,w_2\big), \quad \text{for} \quad (v_1,v_2),(w_1,w_2) \in \Gamma(g). \tag{23}$$

Thus, for any two elements (v_1,v_2) and (w_1,w_2) in $\Gamma(g)$ we obtain:

$$\Omega\big((v_1,v_2),(w_1,w_2)\big) = \omega_1(v_1,w_1) - \omega_1\big(v_1,w_1\big) = 0. \tag{24}$$

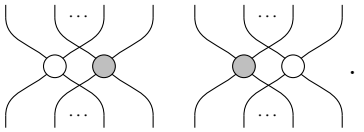
This shows that $\Gamma(g)$ is isotropic in $V_1 \oplus \overline{V_2}$.

To prove that $\Gamma(g)$ is coisotropic, we recall the assumption that we want to find maximal graphs $\Gamma(g)$. This requires the graph to be maximal isotropic. This means that the graph is coisotropic. Thus, the graph of the relation g is a Lagrangian relation.

Theorem B.1. The evolution of logical operators in a surface code between two points of the computation is a Lagrangian relation.

Under our assumptions, braiding defects realise specific affine Lagrangian relations on the surface–code homology. Each defect supports two independent classes: a loop that encircles the defect and a string that reaches a boundary. For a primal defect the loop class is dual (an X-type operator) and the string class is primal (a Z-type operator); for a dual defect the roles are reversed.

When only one species of defects is present, the elementary topological move is a junction that splits one defect into several or merges several into one. A junction copies the loop class—after the move every participating defect inherits the same encircling loop—and it identifies the string classes: strings from different participating defects to the boundary are pairwise homologous and therefore represent a single equivalence class rather than independent operators. Because the two species act on opposite homology types, a primal-type junction copies dual classes while identifying primal classes, whereas a dual-type junction copies primal classes while identifying dual ones. Neglecting processes in which the two species meet, these operations give two distinct affine Lagrangian maps on the surface–code homology, one for each species of defect:



Of course, we cannot compose sequentially two defects of different species and therefore, a map from above corresponding to one type of defect cannot be composed with a map corresponding to the other type of defect (tensor product is allowed though).

We also specifically designate the following (standard [42, 43]) interpretations:

The resulting set of Lagrangian relations is not a subcategory of $\mathbf{LagRel}_{\mathbb{F}_2}$, since it is not closed under sequential composition. Sequentially, we only allow spiders of the same type to be composed.

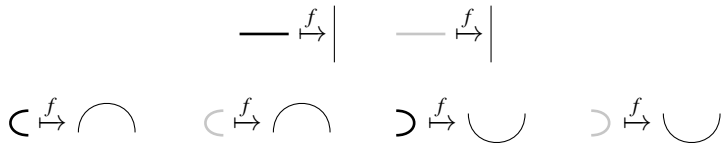
We now have a language to describe how the logical operators evolve in our setting as the computation progresses. This can be understood as an analysis of a hypothetical situation describing what would happen if these operators were applied. If we actually were to apply one of these operators, it would negate or cancel the hypothetical operator considered in the previous sentence. This is what the semantics of the red decorators are. This is actually what the affine Lagrangian relations describe in $\mathbb{F}_2$.

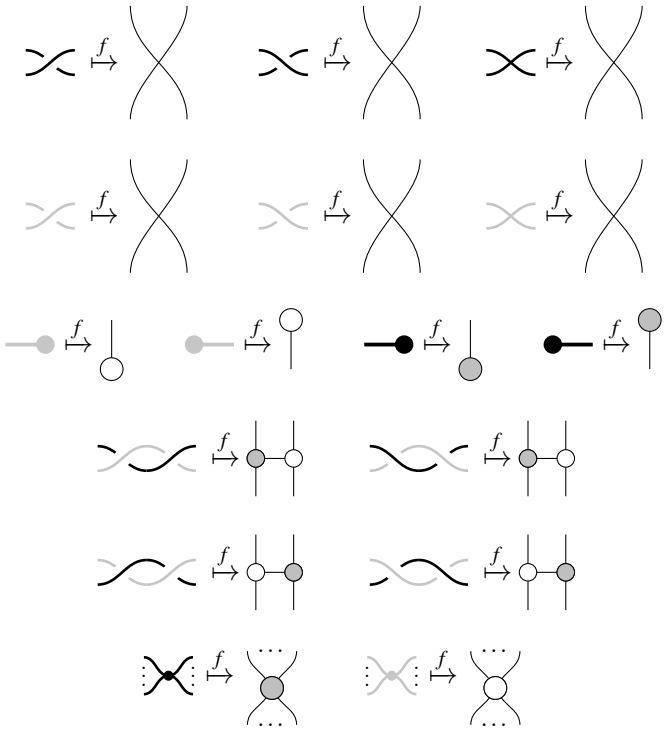
Corollary B.1. The evolution of logical operators in a surface code between two points of the computation together with the logical (non-detectable) errors is an affine Lagrangian relation.

B.4 Soundness of KNOT for Affine Lagrangian relations

Let $\widehat{\mathbf{KNOT}}$ be the category generated by the generators of **KNOT** without the red circles and bars.

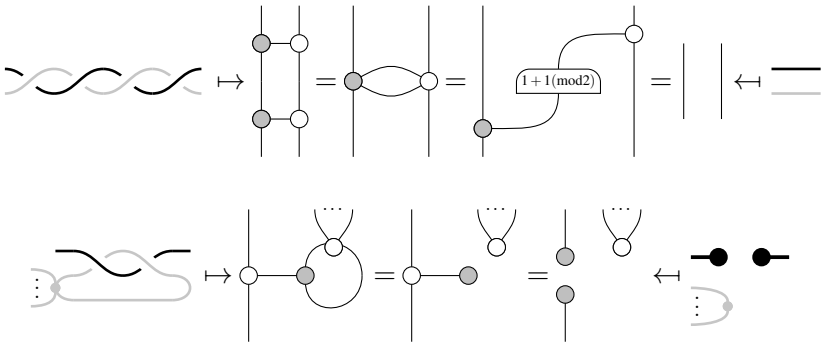
Definition B.13. Define a function f that maps the generators of $\widehat{\mathbf{KNOT}}$ to ZX-diagrams as follows:





Lemma B.7. The map f lifts to a functor $F\colon \widehat{\textbf{KNOT}} \to \textbf{AffRel}_{\mathbb{F}_2}$. It is defined on objects as the length of the bit string and on morphisms as f.

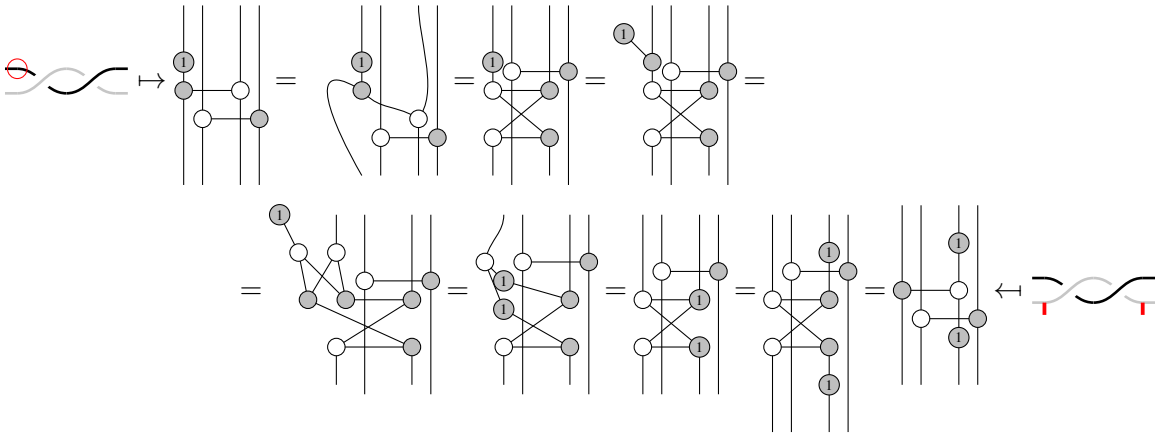
Proof. The axioms of **KNOT** hold in $\textbf{AffRel}_{\mathbb{F}_2}$, as they follow from the equivalent rules in ZX-calculus (see A.1). Let us show two examples.



Equation (K5) can be proven in the same way as the translation into ZX (see A.1). All other rules can be proven with an application of a single rule. □

Composing gives a functor $\widehat{G} = L \circ F : \widehat{\textbf{KNOT}} \to \textbf{LagRel}_{\mathbb{F}_2}$. Finally, we extend $\widehat{G}$ to a functor $G\colon \textbf{KNOT} \to \textbf{AffLagRel}_{\mathbb{F}_2}$ by mapping the red circles and bars as follows:

This is indeed a well-defined functor, as



and the other decorator rules hold trivially.

Corollary B.2. **KNOT** is sound for $\mathbf{AffLagRel}_{\mathbb{F}_2}$.

The objects and diagrams in the image of G do not constitute a category: the morphisms in the image are not closed under composition. This lemma, however, allows us to think of the **KNOT** diagrams as pairs of linear relations over $\mathbb{F}_2$.

Corollary B.3. **KNOT** is sound for the transformations between the first homology groups of the input and output slices of the surface code computation.

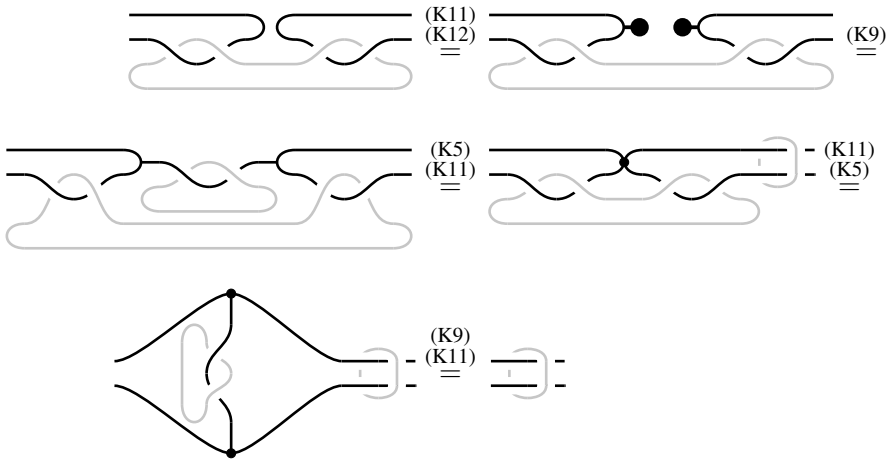
Corollary B.4. **KNOT** is sound for the logical maps implemented by the braided surface code error correction procedure up to classical byproducts.

C Lemmas

Lemma C.1.

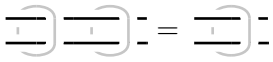
(25)

Proof.



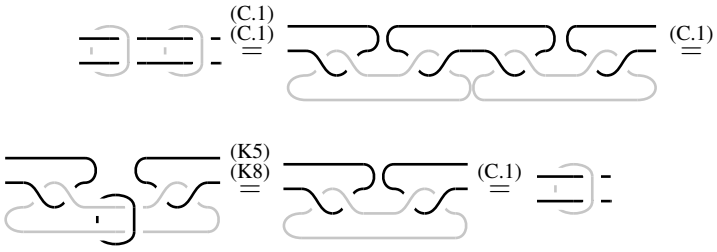
□

Lemma C.2.



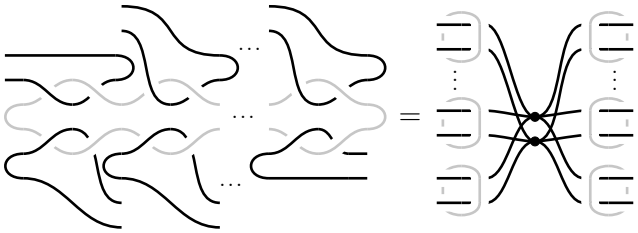
(26)

Proof.



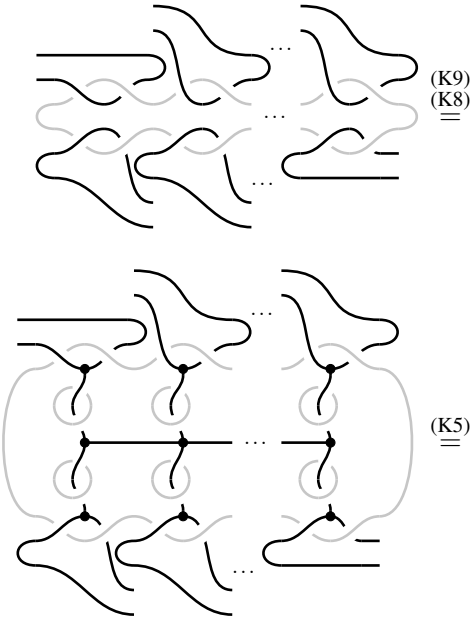
□

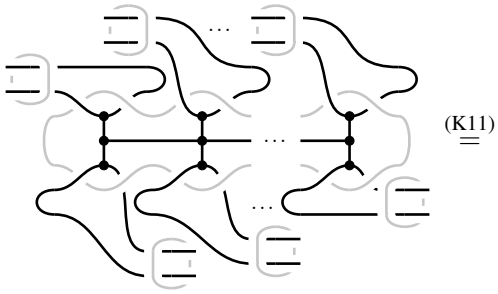
Lemma C.3.

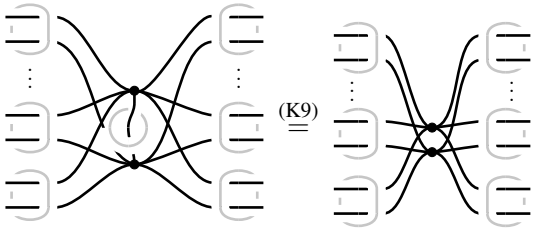


(27)

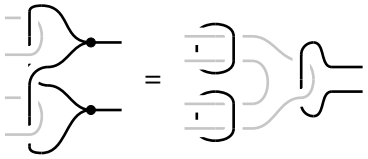
Proof.



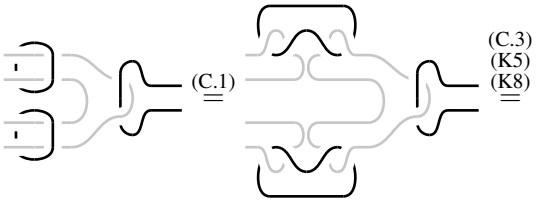


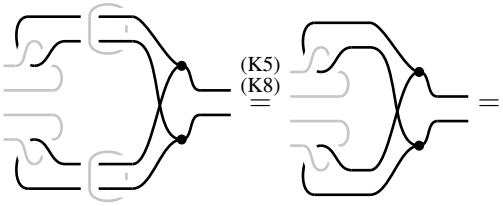


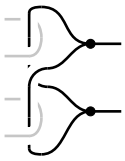
□

Lemma C.4. 

Proof.

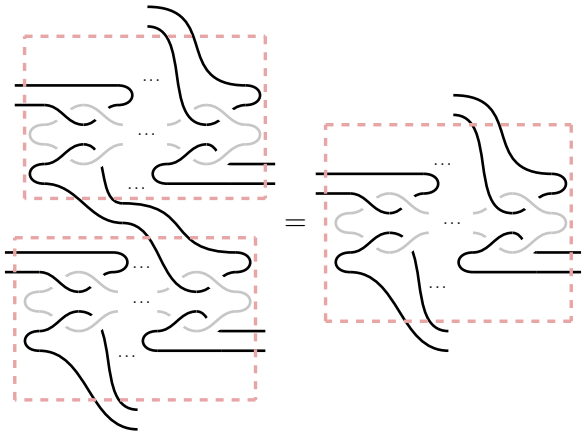






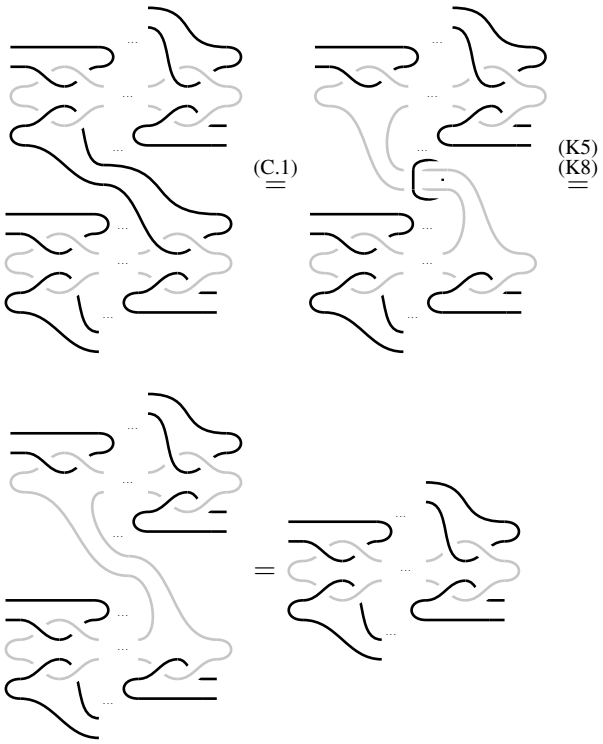
□

Lemma C.5.



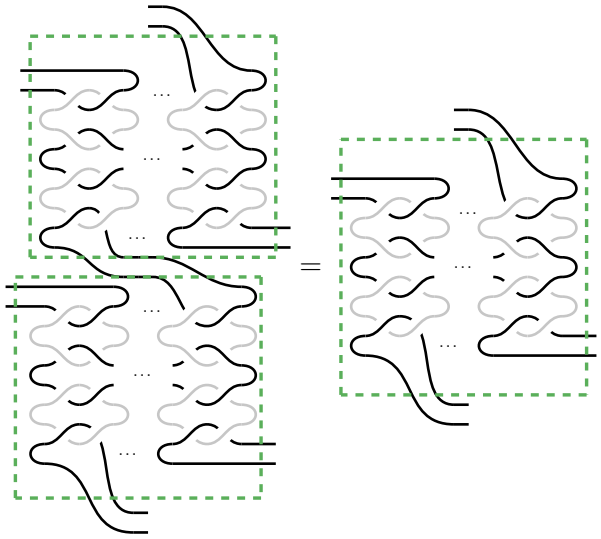
(28)

Proof.



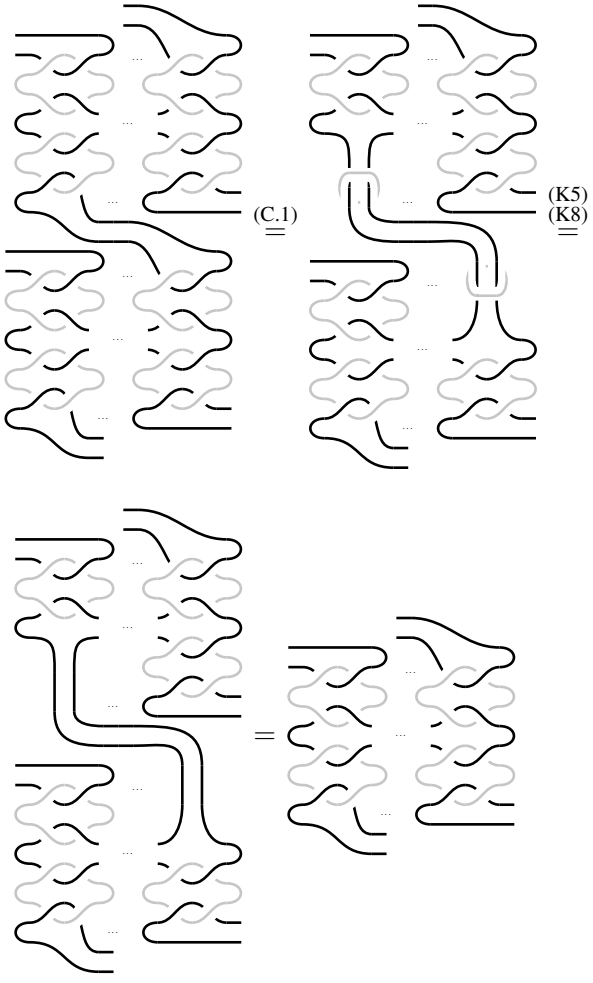
□

Lemma C.6.



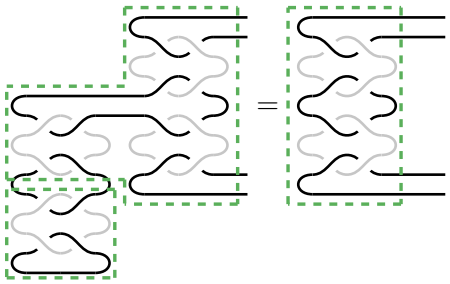
(29)

Proof.



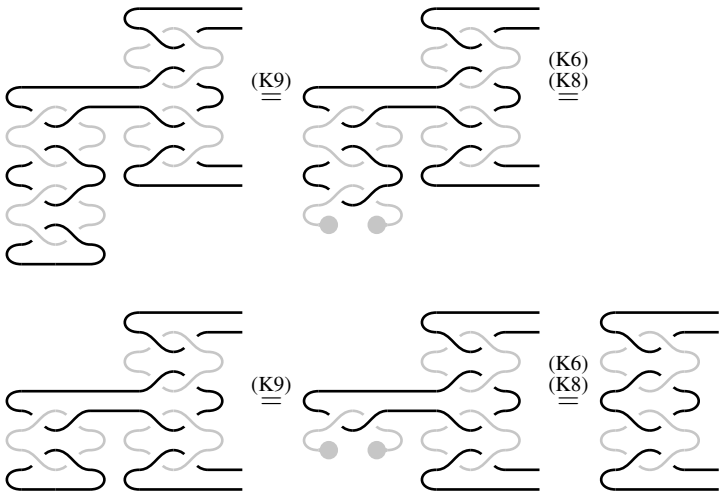
□

Lemma C.7.



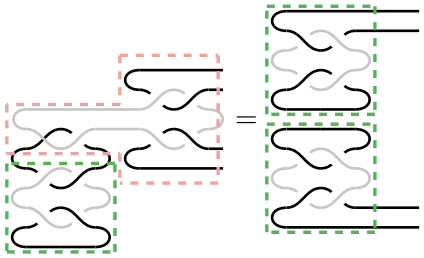
(30)

Proof.



□

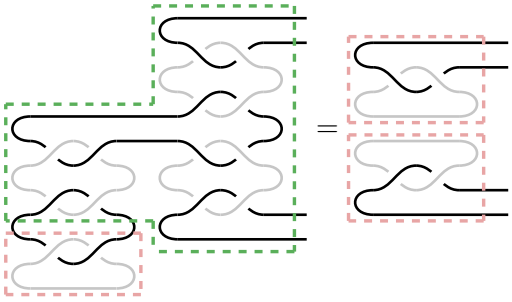
Lemma C.8.



(31)

Proof. Follows from eq. (K9), eq. (K6), and eq. (K8) similarly to C.7. □

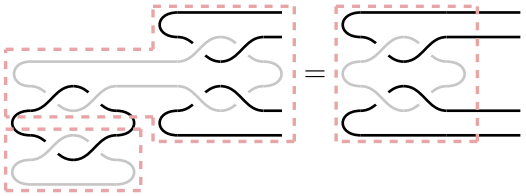
Lemma C.9.



(32)

Proof. Follows from eq. (K9), eq. (K6), and eq. (K8) similarly to C.7. □

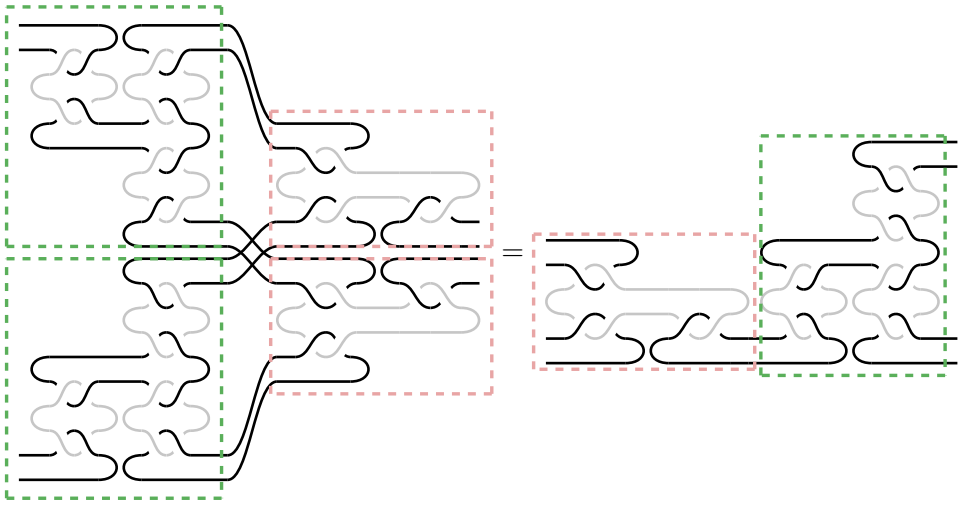
Lemma C.10.



(33)

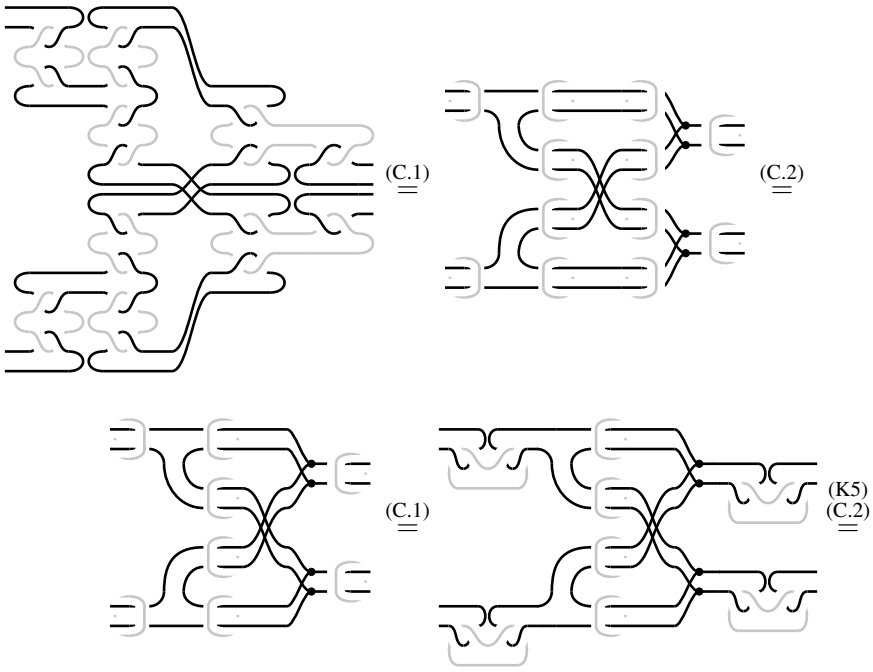
Proof. Follows from eq. (K9), eq. (K6), and eq. (K8) similarly to C.7. □

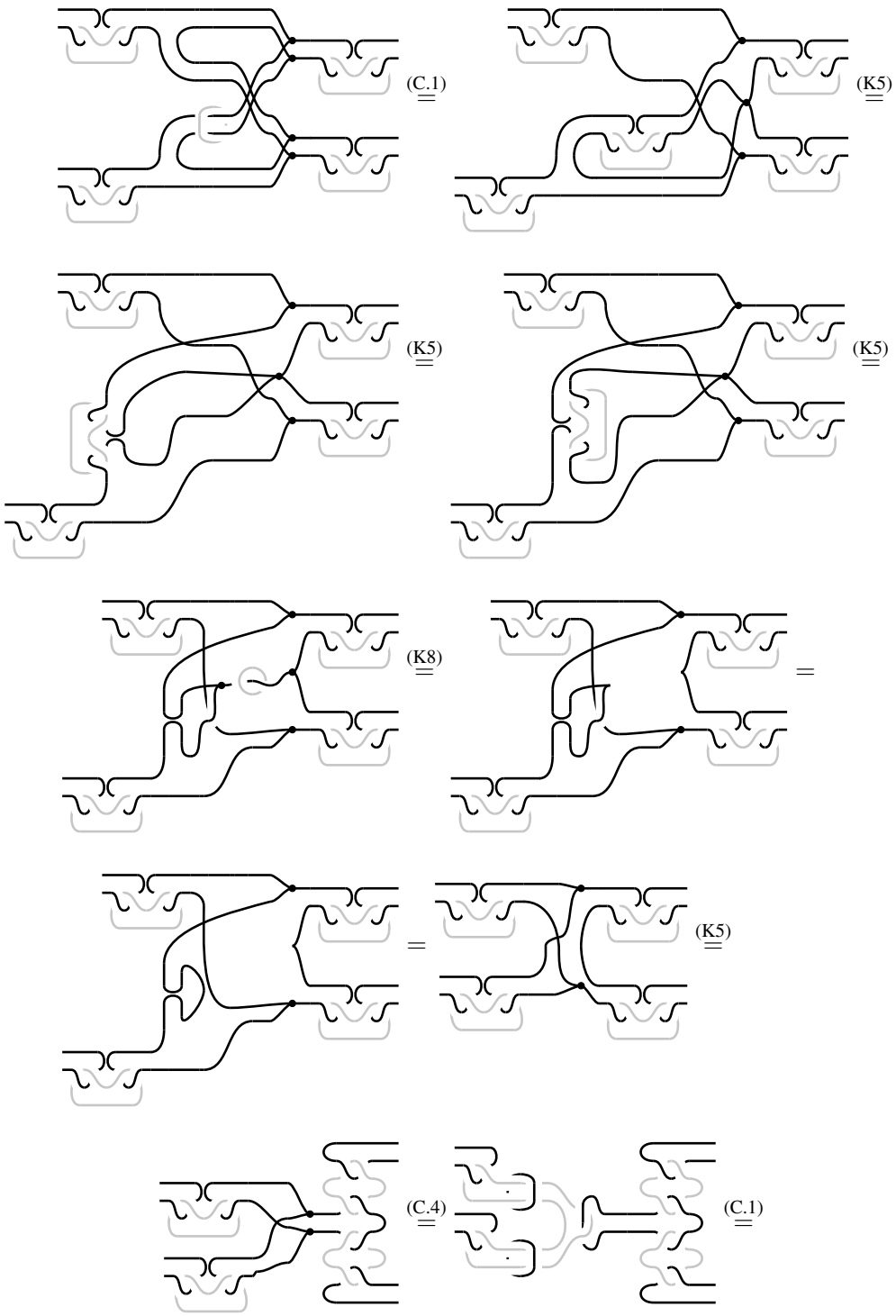
Lemma C.11.

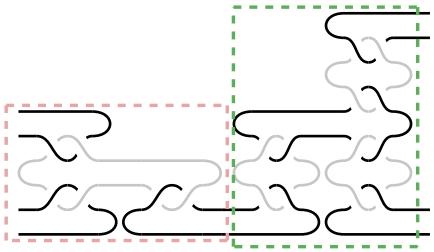


(34)

Proof.

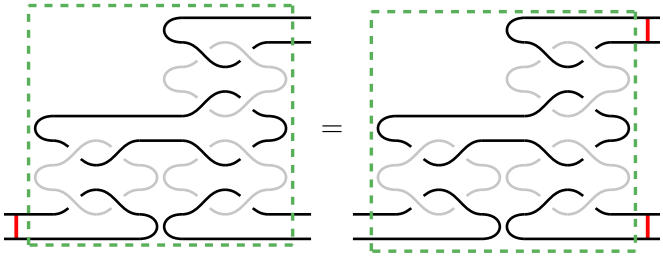



(C.1)
(K5)
(K5)
(K5)
(K8)
=
=
(K5)
(C.4)
(C.1)



□

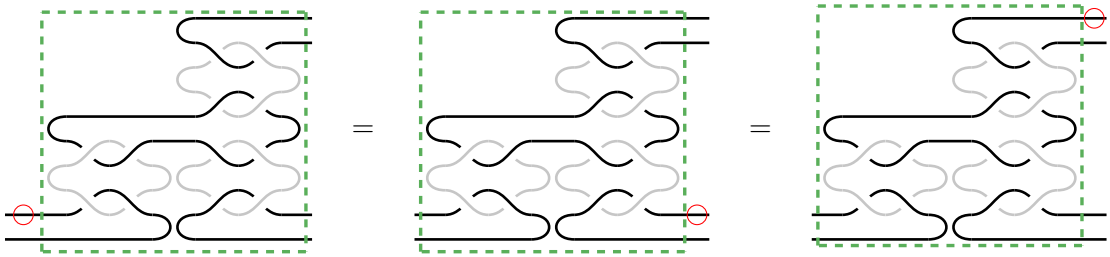
Lemma C.12.



(35)

Proof. Holds by applying rules (K13), (K14), (K15), (K16), (K18) □

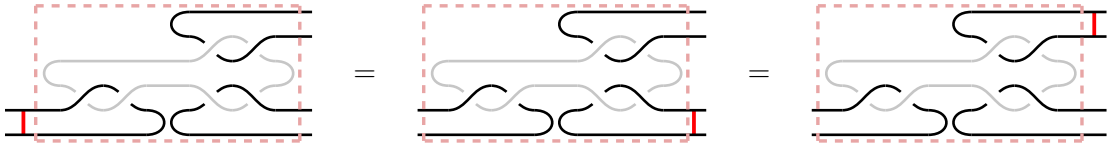
Lemma C.13.



(36)

Proof. Holds by applying rules (K13), (K14), (K15), (K16), (K18) □

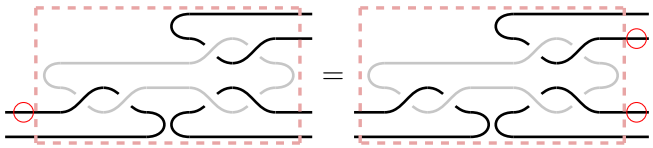
Lemma C.14.



(37)

Proof. Holds by applying rules (K13), (K14), (K15), (K16), (K18) □

Lemma C.15.



(38)

Proof. Holds by applying rules (K13), (K14), (K15), (K16), (K18) □

A Classification Program for Nonlocality Paradoxes of Three Qubits

Nadish de Silva Santanil Jana Ming Yin
Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

Nonlocality is a quintessential signature of nonclassical behaviour and a resource for quantum advantages in communication and computation. The paradoxical correlations witnessed by strong nonlocality undergird the standard probabilistic form of nonlocality and provide optimal advantages in numerous informational tasks.

Three-qubit systems are the simplest ones that admit strong nonlocality. Abramsky et al. (*TQC*, 2017) established the existence of an infinite family of three-qubit paradoxes, beyond the well-known GHZ paradox, which exhibited a novel conditional structure.

In this work, we introduce several new infinite families of three-qubit paradoxes and articulate a detailed roadmap towards the complete classification of all three-qubit nonlocality paradoxes. In particular, we prove that our paradoxes exhaust all those satisfying reasonable regularity conditions. We give an example of a highly exotic paradox and place constraints on the search for new exotic paradoxes. We conjecture that all paradoxes must involve states from a one-parameter family and provide significant evidence in support of this conjecture.

1 Introduction

Bell nonlocality [6, 7] describes how certain quantum correlations cannot be reproduced by a local hidden variable model. Beyond its original foundational significance, it has more recently become the subject of intense research interest as a resource for realising advantages in quantum information-theoretic tasks.

The original arguments due to Bell and Clauser-Horne-Shimony-Holt [10] derive probabilistic inequalities satisfied by classical correlations, yet violated by certain quantum measurements on an entangled state. Greenberger et al. [15, 16] gave a stronger argument for nonlocality, involving the GHZ state, that does not rely on inequalities; rather it involves a logical argument that relies only on the *possibilistic* data concerning outcomes of joint measurements. A set of constraints is given that is satisfied by every empirically observable joint outcome, yet when taken together, form a paradox. That is, no classical local hidden variable is consistent with the empirical observations of certain quantum systems.

Modern structural frameworks unify and generalise these concepts beyond the few previously known examples. Logical paradoxes were shown by Abramsky-Brandenburger [2] in their sheaf-theoretic framework to unify diverse examples of *strong nonlocality*. Paradoxes undergird probabilistic nonlocality in that all nonlocal correlations are dilutions of paradoxes by classical randomness. The nonlocal fraction [5, 13] measures nonlocality by quantifying the degree to which correlations are paradoxical.

Bell nonlocality now plays a pivotal role as a resource in quantum information, underpinning advantages in, e.g. nonlocal games [11], device-independent quantum cryptography [12, 14], and randomness generation [3]. Nonlocality paradoxes also serve as resources in quantum computation. For example, Anders-Browne [4] showed how access to measurements on GHZ states can promote a linear computer to classical universality; Raussendorf [17] extended this to general measurement-based computation. GHZ-type paradoxes also played a critical role in the result of Bravyi-Gosset-König on unconditional

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 197–214, doi:10.4204/EPTCS.426.7

insimulability by shallow circuits [9]. Many results on nonlocality and quantum advantage show that advantages scale with the nonlocal fraction with paradoxes yielding deterministic advantages.

Motivated by the importance of nonlocality paradoxes in the study of quantum foundations and quantum advantage, we develop a program of exhaustively classifying all nonlocality paradoxes realisable by systems of three qubits. This is the simplest system, in the sense of having minimal dimension, admitting strong nonlocality. The first step in this direction was taken by Abramsky et al. [1], who showed that three-qubit nonlocality paradoxes must necessarily involve states LU-equivalent to ones in the SLOCC class of GHZ and local measurements in the equatorial XY-plane. They further exhibited a family of such three-qubit paradoxes, indexed by even positive integers. These paradoxes revealed a striking conditional structure not previously described. Raussendorf [18] gave a topological interpretation of them.

A complete classification of three-qubit nonlocality paradoxes is a difficult mathematical question; we articulate a detailed roadmap towards its resolution and prove many key steps. Further study of the structure of quantum paradoxes has potential applications across quantum information and computing.

1.1 Summary of main results

- We study paradoxes up to a notion of equivalence based on local unitary equivalence and permutations of qubits. We restrict ourselves to paradoxes that are minimal in the sense of not involving any extraneous measurements. In Section 2, we define the family of *interpolant states* to be those that interpolate between $|\mathrm{GHZ}\rangle$ and $|\mathrm{Bell}\rangle \otimes |+\rangle$ via a parameter $\lambda \in \left[0, \frac{\pi}{2}\right)$. All previously known paradoxes and the ones we demonstrate below involve these states. We provide strong theoretical evidence in Section 4.2 for our Conjecture 4.15 that all three-qubit nonlocality paradoxes are equivalent to one using an interpolant state.
- We establish some constraints on the structure of impossible events for a fixed context that apply in any three-qubit model (see Lemmas 3.2 and 3.3). Any two impossible events must differ in at least two outcomes; thus, every context can have at most four impossible events.
- Any paradox using an interpolant state exhibits a conditional $\mathbb{Z}_2$-linear structure (Lemma 3.4). *Maximal rank* paradoxes are those witnessed by summing *all* linear equations to find $0 = 1$.
- We limit how many impossible events any choice of measurements on an interpolant state can yield overall and define a notion of *maximally impossible* quantum scenarios of an interpolant state and measurement sets. We focus on these quantum scenarios, following the intuition that paradoxes require many impossible events, and show how their measurement sets are constrained (Lemmas 3.6 and 3.9).
- Theorem 4.6 introduces several new infinite families of three-qubit nonlocality paradoxes. One family is an extension of [1, Theorem 8], which considered $\lambda_N = \frac{\pi}{2} - \frac{\pi}{N}$ indexed by even $N \in \mathbb{N}$. Our construction, indexed by a tuple (N,t), corresponds to a subset of the rationals and includes infinitely more paradoxes. We give two more infinite families with significantly differing structure. Theorem 4.7 shows that all our paradoxes are in distinct equivalence classes.
- We give a partial classification of three-qubit nonlocality paradoxes up to equivalence in Theorem 4.13. We define *N-regular* quantum scenarios of a state and measurement sets that are maximally impossible, maximal rank, and use two measurements on the third qubit. Our families of paradoxes exhaust all *N-regular* paradoxes up to equivalence.
- We give an example of a paradox that is not N-regular and thus exhibits a very exotic structure. We also study the consequences of relaxing the assumptions of regularity and indicate the most

promising directions for completing the classification of quantum tripartite nonlocality paradoxes.

Outline. Section 2 summarises background material on nonlocality, and three-qubit paradoxes. Section 3 establishes our technical foundations by articulating the logical structure of reasonably regular paradoxes. In Section 4, we present our main results, including the presentation of new families of paradoxes in Subsection 4.1, a partial classification of three-qubit paradoxes in Subsection 4.2, and a proposed roadmap towards a complete classification in Subsection 4.3. The appendices contain some detailed proofs and figures illustrating exotic paradoxes.

2 Background

2.1 Strong nonlocality

We recall the basic definitions of the Abramsky-Brandenburger [2] framework for nonlocality and contextuality (adapted to the setting of n-qubit nonlocality), and the notation of Abramsky et al. [1].

Measurement scenarios provide an abstract framework for describing experimental setups. Typically, a measurement scenario $(\mathscr{X},\mathscr{O},\mathscr{C})$ is defined by a set of measurement labels $\mathscr{X}$, a set of possible outcomes $\mathscr{O}$, and a cover $\mathscr{C}$ of $\mathscr{X}$, consisting of measurement contexts, which are maximal sets of measurements that can be jointly performed. Throughout this paper, we will set $\mathscr{O}$ to be $\mathbb{Z}_2 = \{0,1\}$ since our focus will be on quantum realisable paradoxes and projective measurements.

A **Bell measurement scenario** $\mathscr{M}$ on n qubits is an n-tuple of finite sets $(M_1,\ldots,M_n)$ with each M_i a set of measurements on the i-th qubit. The **contexts** of $\mathscr{M}$ are given by $\mathscr{C} = \prod_{i=1}^{n} M_i$, i.e. choices of exactly one measurement per qubit. The **empirical model** $\varepsilon(|\psi\rangle,\mathscr{M})$ yielded by a **quantum scenario** $(|\psi\rangle,\mathscr{M})$ is a set of probability distributions $P_C : \mathscr{O}^n \to [0,1]$, indexed by contexts $C \in \mathscr{C}$, given by the Born rule.

A local measurement can be written in the form $E_{\theta,\varphi} := \sin(\theta)[\cos(\varphi)X + \sin(\varphi)Y] + \cos(\theta)Z$, where $\theta \in \left[0,\frac{\pi}{2}\right)$ and $\varphi \in [0,2\pi)$. This is a measurement with $+1$ eigenstate $|\theta,\varphi\rangle := \frac{1}{\sqrt{2}}(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle)$ and -1 eigenstate $|\pi-\theta,\varphi+\pi\rangle$. Thus we can label a context by a tuple $(\vec{\theta},\vec{\varphi}) := ((\theta_1,\varphi_1),\ldots,(\theta_n,\varphi_n))$. We can then denote an **event** by $(\vec{\theta},\vec{\varphi}) \to \vec{o}$, where $\vec{o} \in \mathscr{O}^n$ labels measurement outcomes (relabelling $+1,-1$ to $0,1$ respectively).

An empirical model $\varepsilon(|\psi\rangle,\mathscr{M})$ is **strongly nonlocal** if, given any **global assignment** $g : \bigsqcup_{i=1}^{n} M_i \to \mathscr{O}$ of outcomes to the measurements of $\mathscr{M}$, there exists a context $(\vec{\theta},\vec{\varphi})$ such that the event $(\vec{\theta},\vec{\varphi}) \to (g(\theta_1,\varphi_1),\ldots,g(\theta_n,\varphi_n))$ is impossible. In quantum terms, if we denote the resulting eigenstate due to the event $(\vec{\theta},\vec{\varphi}) \to \vec{o}$ by $|(\vec{\theta},\vec{\varphi}) \to \vec{o}\rangle$, then the event is **impossible** if and only if $\langle(\vec{\theta},\vec{\varphi}) \to \vec{o}\,|\,\psi\rangle = 0$. We refer to a quantum scenario $(|\psi\rangle,\mathscr{M})$ such that $\varepsilon(|\psi\rangle,\mathscr{M})$ is strongly nonlocal as a **(quantum nonlocality) paradox**.

2.2 Three-qubit nonlocality paradoxes

Brassard et al. [8] proved that two-qubit quantum states do not exhibit strongly nonlocal behaviour. Thus, at least three qubits are required for strong nonlocality. Abramsky et al. [1] showed the following:

Theorem 2.1 ([1, Theorem 6])**.** *A tripartite quantum state admitting strong nonlocality must be in the SLOCC class of the GHZ state and, in particular, must be* balanced*. Moreover, any such strongly nonlocal behaviour can be witnessed using only* equatorial measurements*.*

A **balanced** state has the form $|\mathrm{B}(\vec{\lambda},\Phi)\rangle = \frac{1}{\sqrt{2}}(|v_{\vec{\lambda}}\rangle + e^{i\Phi}|w_{\vec{\lambda}}\rangle)$, where $\vec{\lambda} = (\lambda_1,\lambda_2,\lambda_3)$ with $\lambda_j \in \left[0,\frac{\pi}{2}\right)$, $\Phi \in [0,2\pi)$, and $|v_{\vec{\lambda}}\rangle = \bigotimes_{i=1}^3 |v_{\lambda_i}\rangle$, $|w_{\vec{\lambda}}\rangle = \bigotimes_{i=1}^3 |w_{\lambda_i}\rangle$ with

$$|v_{\lambda_j}\rangle := \cos\frac{\lambda_j}{2}|0\rangle + \sin\frac{\lambda_j}{2}|1\rangle, \quad |w_{\lambda_j}\rangle := \sin\frac{\lambda_j}{2}|0\rangle + \cos\frac{\lambda_j}{2}|1\rangle. \tag{1}$$

Equatorial measurements are of the form $E_\varphi := \cos(\varphi)X + \sin(\varphi)Y$ for $\varphi \in [0,2\pi)$. Since the only measurements that contribute to strong nonlocality are equatorial ones, we only need to consider measurement scenarios $\mathscr{M} = (M_1,M_2,M_3)$ where each set M_i consists of measurement angles $\varphi \in [0,2\pi)$. The following remark yields further simplifications.

Remark 2.2. As discussed in Abramsky et al. [1, Section 2.3], it suffices to only consider global assignments $g : \bigsqcup_{i=1}^n M_i \to \mathscr{O}$ with the property that, for any local equatorial measurement E_φ on any qubit,

$$g(\varphi) = g(\varphi+\pi) \oplus 1. \tag{2}$$

That is, to show $\varepsilon(|\psi\rangle,\mathscr{M})$ is strongly nonlocal, it suffices to demonstrate inconsistency of only those global assignments satisfying Eq. 2. Conversely, if $\varepsilon(|\psi\rangle,\mathscr{M})$ is not strongly nonlocal then there is a consistent global assignment that also satisfies Eq. 2. Thus, when constructing measurement scenarios, we can consider each set M_i as only containing angles in the interval $[0,\pi)$.

Letting $\vec{\varphi} = (\varphi_1,\varphi_2,\varphi_3)$, define $|\vec{\varphi}\rangle = \bigotimes_{i=1}^3 |\varphi_i\rangle$. We want to understand when the amplitude $\langle\vec{\varphi}|\mathrm{B}(\vec{\lambda},\Phi)\rangle$ is 0, since this corresponds to the event $\vec{\varphi} \to (0,\ldots,0)$ being impossible.

Lemma 2.3 ([1])**.** *Let $\beta : \left[0,\frac{\pi}{2}\right) \times [0,2\pi) \to \mathbb{R}$ be defined as follows:*

$$\beta(\lambda,\varphi) = \varphi - 2\arctan\left(\frac{\cos\frac{\lambda}{2}\sin\varphi}{\sin\frac{\lambda}{2}+\cos\frac{\lambda}{2}\cos\varphi}\right). \tag{3}$$

Then

$$\langle\vec{\varphi}|\mathrm{B}(\vec{\lambda},\Phi)\rangle = 0 \iff \sum_{i=1}^3 \beta(\lambda_i,\varphi_i) \equiv \pi - \Phi \mod 2\pi. \tag{4}$$

For the rest of the paper, the equivalence symbol $\equiv$ is used to denote when two quantities are equal modulo 2π, unless otherwise specified. We note that the formula for the function β differs slightly from that of [1, Section 5.3]. The proof is straightforward and is therefore omitted.

Lemma 2.4. *The following properties of the function β can be easily verified.*

1. *Modulo 2π, for all fixed $\lambda \in \left[0,\frac{\pi}{2}\right)$, $\beta(\lambda,\varphi)$ is strictly decreasing as a function of φ on $(0,2\pi)$ and is thus bijective on $[0,2\pi)$.*
2. *$\beta(0,\varphi) \equiv -\varphi$.*
3. *For all $\lambda \in \left[0,\frac{\pi}{2}\right)$, $\beta(\lambda,\varphi) \equiv 0$ if and only if $\varphi \equiv 0$, and $\beta(\lambda,\varphi) \equiv \pi$ if and only if $\varphi \equiv \pi$.*
4. *$\beta(\lambda,\frac{\pi}{2}) \equiv \lambda - \frac{\pi}{2}$.*

To avoid redundancy, we consider quantum scenarios $(|\psi\rangle,\mathscr{M})$, $(|\psi\rangle',\mathscr{M}')$ to be **equivalent** if $U|\psi\rangle = |\psi'\rangle$ and $M \in M_i$ if and only if $UMU^\dagger \in M'_{P(i)}$; here P is some permutation of qubit labels and we have used shorthand notation to denote the rotation of an equatorial measurement, mod π, by some local unitary U. Note that if $\vec{\lambda}$ has at least one zero entry, without loss of generality $\lambda_1 = 0$, then $(|\mathrm{B}(\vec{\lambda},\Phi)\rangle,\mathscr{M})$ is equivalent to $(|\mathrm{B}(\vec{\lambda},0)\rangle,\mathscr{M}')$ via the local unitary $\mathrm{diag}(1,e^{-i\Phi})\otimes I\otimes I$.

For the rest of the paper, we mostly focus on a specific subclass of balanced states, defined as follows:

Definition 2.5. An **interpolant state** $|\mathrm{B}(\lambda)\rangle$ is a balanced state of the form $|\mathrm{B}((0,0,\lambda),0)\rangle$.

All examples of paradoxes found in [1] and in Section 4.1 of this paper involve interpolant states. These states interpolate between $|\mathrm{B}(0)\rangle = |\mathrm{GHZ}\rangle$ and $\lim_{\lambda\to\frac{\pi}{2}}|\mathrm{B}(\lambda)\rangle = |\mathrm{Bell}\rangle\otimes|+\rangle$.

3 Technical preliminaries

In this section, we collect novel theoretical developments required for our main results.

Throughout this paper, we will assume that all paradoxes fulfil the following definition.

Definition 3.1. A paradox $(|\psi\rangle, \mathscr{M})$ is **minimal** if $(|\psi\rangle, \mathscr{M}')$ is not a paradox for any $\mathscr{M}'$ containing strictly fewer measurements than $\mathscr{M}$, i.e. $M_i' \subseteq M_i$ for all i and at least one containment is strict.

3.1 Impossible events in quantum tripartite models

We define an additional function,

$$\delta(\lambda, \varphi) := \beta(\lambda, \varphi + \pi) - \beta(\lambda, \varphi), \tag{5}$$

which tells us how β changes when we flip a measurement outcome on one qubit. Note that

$$\delta(\lambda, \varphi) \equiv \pi - 2\arctan(\sin\varphi \tan\lambda), \tag{6}$$

for which we use the fact that $\arctan u + \arctan v = \arctan\left(\frac{u+v}{1-uv}\right) + k\pi$ for some $k \in \{-1, 0, 1\}$.

Lemma 3.2. *Let $\lambda \in \left[0, \frac{\pi}{2}\right)$ and $\varphi \in [0, \pi)$. Then one can easily check the following properties of δ.*

1. $\delta(\lambda, \varphi) \in (0, \pi]$.
2. $\delta(\lambda, \varphi) \equiv \pi$ *if and only if* $\sin\varphi \tan\lambda = 0 \iff \lambda = 0$ *or* $\varphi = 0$.
3. $\delta(\lambda_1, \varphi_1) \pm \delta(\lambda_2, \varphi_2) \equiv 0 \iff \sin\varphi_1 \tan\lambda_1 = \mp \sin\varphi_2 \tan\lambda_2$. *Further, by Lemma 3.2.1,*

$$\delta(\lambda_1, \varphi_1) + \delta(\lambda_2, \varphi_2) \equiv 0 \iff \delta(\lambda_1, \varphi_1) = \delta(\lambda_2, \varphi_2) \equiv \pi. \tag{7}$$

Suppose that, considering measurement of the context (A, B, C) on a state, the event $ABC \to abc$ is impossible. By Lemma 3.2.1, $\delta(\lambda, \varphi) \not\equiv 0 \ \forall \lambda, \varphi$, so any other impossible event of the same context must have a tuple of outcomes whose Hamming distance from abc is at least 2. Therefore, the number of impossible events associated with any context is at most 4.

The following lemma is a consequence of Lemmas 3.2.2 and 3.2.3.

Lemma 3.3. *Fix a context (A, B, C).*

1. *For a given $c \in \mathbb{Z}_2$, if one of the events $ABC \to 00c$, $ABC \to 11c$ is impossible, then the other event is impossible if and only if $\delta(\lambda_1, A) \equiv \delta(\lambda_2, B) \equiv \pi$.*
2. *For a given $c \in \mathbb{Z}_2$, if one of the events $ABC \to 01c$, $ABC \to 10c$ is impossible, then the other event is impossible if and only $\delta(\lambda_1, A) \equiv \delta(\lambda_2, B)$.*

Similar results hold by permuting the qubits.

3.2 Logical structure of three-qubit nonlocality paradoxes using interpolant states

The family of three-qubit paradoxes introduced in [1] differed from that of the GHZ state in that they involved not a single unsatisfiable system of linear equations over $\mathbb{Z}_2$ but two such systems. These two systems were conditioned on the outcome of the third-qubit measurement and found to be inconsistent in both cases. Here, we show that this conditional structure is a feature of every paradox involving an interpolant state $|\mathrm{B}(\lambda)\rangle$. We express this structure in a convenient form that we use in the remainder of the paper.

First we note that, by Lemma 2.4.2, Eq. 4 has a much simpler form for interpolant states. Given a context (A_j, B_k, C_l), the outcome (a_j, b_k, c_l) is *impossible* if and only if

$$A_j + B_k \equiv \beta(\lambda, C_l) + c_l \delta(\lambda, C_l) + (1 \oplus a_j \oplus b_k)\pi, \tag{8}$$

where $\oplus$ denotes addition modulo 2. Note that $\beta(\lambda, C_l) + c_l \delta(\lambda, C_l) = \beta(\lambda, C_l + c_l \pi)$. By rearranging Eq. 8, we see that a necessary condition for an event $A_j B_k C_l \to a_j b_k c_l$ to be impossible is that $A_j + B_k - \beta(\lambda, C_l) - c_l \delta(\lambda, C_l)$ must be a multiple of π.

Define $r_{jkl}(z) := \pi^{-1}[(A_j + B_k) - \beta(\lambda, C_l + z\pi)] \mod 2 \in [0, 2)$. By Lemma 2.3 and Lemma 2.4.2, the outcome (a_j, b_k, c_l) is *possible* if and only if

$$r_{jkl}(c_l) \notin \mathbb{Z}_2 \quad \text{or} \quad (a_j \oplus b_k) = r_{jkl}(c_l). \tag{9}$$

For $z \in \mathbb{Z}_2$, define the set of $\mathbb{Z}_2$-linear equations:

$$\Psi_l(z) := \{a_j \oplus b_k = r_{jkl}(z) \mid \forall j,k \text{ such that } r_{jkl}(z) \in \mathbb{Z}_2\}. \tag{10}$$

A global assignment $g : \bigsqcup M_i \to \mathscr{O}$ is consistent with an empirical model of $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ if and only if, for every l, $\Psi_l(z)$ is satisfied by $a_j = g(A_j)$, $b_k = g(B_k)$, $c_l = g(C_l)$. We have thus shown the following.

Lemma 3.4. *The quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$, with $M_3 = \{C_0, \ldots, C_{n-1}\}$, is a paradox if and only if for every $\vec{z} \in \mathbb{Z}_2^n$, the system of $\mathbb{Z}_2$-linear equations $\Psi(\vec{z}) := \cup_l \Psi_l(z_l)$ is inconsistent.*

3.3 Maximally impossible quantum scenarios

Intuitively, an empirical model with more impossible events has a higher likelihood of witnessing a paradox compared to one with fewer impossible events. This motivates us to focus on empirical models in which the number of impossible events is maximised. We formalise this idea in the following definition.

Definition 3.5. Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be a quantum scenario involving an interpolant state. Let $C \in M_3$, $z \in \mathbb{Z}_2$ and define:

$$\mathbb{A}_z(C) := \{A \in M_1 \mid \exists B \in M_2,\ a, b \in \mathbb{Z}_2 : ABC \to abz \text{ is impossible}\} \tag{11}$$

$$\mathbb{B}_z(C) := \{B \in M_2 \mid \exists A \in M_1,\ a, b \in \mathbb{Z}_2 : ABC \to abz \text{ is impossible}\}. \tag{12}$$

Also define $\mathbb{A}(C) := \mathbb{A}_0(C) \cap \mathbb{A}_1(C)$ and $\mathbb{B}(C) := \mathbb{B}_0(C) \cap \mathbb{B}_1(C)$. Then $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is **maximally impossible** if for all $C \in M_3$, we have $\mathbb{A}(C) = M_1$ and $\mathbb{B}(C) = M_2$.

The definition of $\mathbb{A}(C)$ can be reformulated as follows: $A \in \mathbb{A}(C)$ if and only if $\exists B_1, B_2 \in M_2$ such that $[A + B_1 \equiv \beta(\lambda, C)$ or $\beta(\lambda, C) + \pi]$ and $[A + B_2 \equiv \beta(\lambda, C + \pi)$ or $\beta(\lambda, C + \pi) + \pi]$. The definition of $\mathbb{B}(C)$ can be reformulated in a similar way.

Now we prove some properties of maximally impossible quantum scenarios. First, observe that, by Eq. 4, if $A \in \mathbb{A}_z(C)$, then there is exactly one measurement B that satisfies the required property in Eq. 11, and likewise for measurements in $\mathbb{B}_z(C)$. This justifies our nomenclature.

Lemma 3.6. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be maximally impossible. Then $|M_1| = |M_2|$.*

Proof. Let $C \in M_3$ and fix an outcome $z \in \mathbb{Z}_2$. We define the function $f_1 : M_1 \to M_2$ as follows: $f_1(A) = B$, where B is such that $ABC \to abz$ is an impossible event for some choice of $a, b \in \mathbb{Z}_2$. Since $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is maximally impossible, f_1 is well-defined and injective. By symmetry, we can define a function $f_2 : M_2 \to M_1$ such that f_2 is injective. This implies $|M_1| = |M_2|$. □

Remark 3.7. Thus, given a maximally impossible quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ with $|M_1| = |M_2| = N$ and $|M_3| = n$, there exists a (unique) function $K : \mathbb{Z}_N \times \mathbb{Z}_n \times \mathbb{Z}_2 \to \mathbb{Z}_N$ such that for all $(j,l,z) \in \mathbb{Z}_N \times \mathbb{Z}_n \times \mathbb{Z}_2$ there exists an impossible event $A_j B_{K(j,l,z)} C_l \to abz$ for some $a,b \in \mathbb{Z}_2$. Furthermore, $K(-,l,z)$ gives a bijection $\mathbb{Z}_N \leftrightarrow \mathbb{Z}_N$ for all $(l,z) \in \mathbb{Z}_n \times \mathbb{Z}_2$. We can then write the $\mathbb{Z}_2$-linear equations of Eq. 10 using the following notation, which emphasises the dependencies among the indices:

$$\Psi_l(z) = \{a_j \oplus b_{K(j,l,z)} = r(j,l,z) \mid j \in \mathbb{Z}_N\}. \tag{13}$$

Remark 3.8. Suppose in particular that $M_3 = \{C_0, C_1\}$ (i.e. $n = 2$). Let $(z_0, z_1) \in \mathbb{Z}_2^2$ and consider the linear system $\Psi(z_0,z_1) = \Psi_0(z_0) \cup \Psi_1(z_1)$. By the discussion above, each of the variables a_j, b_k for $j,k \in \mathbb{Z}_N$ appears in exactly two of the $2N$ equations in $\Psi(z_0,z_1)$, so the sum of the LHS of the system is 0. Write the system in matrix form, $\Gamma\vec{a} = \vec{r}$, where $\vec{a} = (a_0, \ldots, a_{N-1}, b_0, \ldots, b_{N-1})^T$ and $\vec{r} = \big(r(0,0,z_0),\ \ldots,\ r(N-1,0,z_0),\ r(0,1,z_1),\ \ldots,\ r(N-1,1,z_1)\big)^T$. Then the kernel of the coefficient matrix Γ is nontrivial and hence the rank of Γ is at most $2N-1$. The system is inconsistent if and only if the augmented matrix $(\Gamma \mid \vec{r})$ has a higher rank than Γ.

Lemma 3.9. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be a maximally impossible quantum scenario. Let $|M_1| = |M_2| = N$ and $M_3 = \{C_0, \ldots, C_{n-1}\}$. Then $\delta(\lambda, C_l)$ and $\beta(\lambda, C_{l_1}) - \beta(\lambda, C_{l_2})$ are integer multiples of $\frac{\pi}{N}$ for all $l, l_1, l_2 \in \mathbb{Z}_n$.*

Proof. Let $C_l \in M_3$ and K be the function as defined in Remark 3.7. For every $A_j \in M_1$ and outcome $c_l = z \in \mathbb{Z}_2$, Eq. 8 is satisfied by $k = K(j,l,z)$. Therefore

$$A_j + B_{K(j,l,0)} \equiv \beta(\lambda, C_l) + (1 \oplus a_j \oplus b_{K(j,l,0)})\pi \qquad \text{for } j \in \mathbb{Z}_N \tag{14}$$

$$\text{and} \quad A_j + B_{K(j,l,1)} \equiv \beta(\lambda, C_l) + \delta(\lambda, C_l) + (1 \oplus a_j \oplus b_{K(j,l,1)})\pi \quad \text{for } j \in \mathbb{Z}_N. \tag{15}$$

If we sum the N equations from Eq. 14 and separately sum the N equations from Eq. 15, then subtract the former from the latter, we get $N\delta(\lambda, C_l) \equiv 0$ or π. Note that the sum of the $B_{K(j,l,0)}$'s and the sum of the $B_{K(j,l,1)}$'s cancel out. So, $\delta(\lambda, C_l) \equiv s\frac{\pi}{N}$ for some $s \in \{1, \ldots, N\}$.

Let $C_{l_1}, C_{l_2} \in M_3$. When $c_{l_1} = c_{l_2} = 0$, the maximally impossible property implies the following:

$$A_j + B_{K(j,l_1,0)} \equiv \beta(\lambda, C_{l_1}) + (1 \oplus a_j \oplus b_{K(j,l_1,0)})\pi \quad \text{for } j \in \mathbb{Z}_N \tag{16}$$

$$\text{and} \quad A_j + B_{K(j,l_2,0)} \equiv \beta(\lambda, C_{l_2}) + (1 \oplus a_j \oplus b_{K(j,l_2,0)})\pi \quad \text{for } j \in \mathbb{Z}_N. \tag{17}$$

Similarly to above, if we add all the equations on the first line and subtract the equations on the second line, we get $N(\beta(\lambda, C_{l_1}) - \beta(\lambda, C_{l_2})) \equiv 0$ or π. Therefore, $\beta(\lambda, C_{l_1}) - \beta(\lambda, C_{l_2}) \equiv t\frac{\pi}{N}$ for some $t \in \{0, \ldots, N-1\}$. □

4 Main results

In this section, we introduce several families of paradoxes, including a natural extension of [1, Theorem 8]. We provide graphical visualisations of these paradoxes and establish that they are pairwise distinct up to equivalence. Next, we classify all N-regular (see Definition 4.1) paradoxes by showing that the previously constructed paradoxes exhaust all such cases. Additionally, we present a paradox that is not N-regular, suggesting the existence of exotic paradoxes with novel structures. Finally, we propose a conjecture that paradoxes must involve interpolant states, and provide supporting evidence.

4.1 New families of paradoxes

We first focus on maximally impossible quantum scenarios $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ such that there are two measurements on the third qubit. Referring to Remark 3.8, we introduce an additional assumption that, for all $z_0, z_1 \in \mathbb{Z}_2$, the coefficient matrix Γ of the corresponding linear system $\Psi(z_0, z_1)$ achieves its maximum possible rank of $2N-1$. We shall use the term **maximal rank** to describe the quantum scenarios satisfying this assumption.

Definition 4.1. The quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is ***N*-regular** if it is maximally impossible and maximal rank, with $|M_1| = |M_2| = N$ and $M_3 = \{C_0, C_1\}$.

Lemma 4.2. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be N-regular. For any $z_0, z_1 \in \mathbb{Z}_2$, the linear system $\Psi(z_0, z_1)$ is inconsistent if and only if*

$$\bigoplus_{j=0}^{N-1} r(j, 0, z_0) \neq \bigoplus_{j=0}^{N-1} r(j, 1, z_1). \tag{18}$$

Proof. Since the quantum scenario is maximal rank, $\Psi(z_0, z_1)$ is inconsistent if and only if the rank of the corresponding augmented matrix is $2N$, which holds if and only if the sum of all the entries of $\vec{r}$ (the RHS of the matrix equation as defined in Remark 3.8) is 1. □

Remark 4.3. The 'if' direction of Lemma 4.2 holds even if we remove the property of maximal rank.

Corollary 4.4. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be an N-regular paradox.*

1. *For any $z_0, z_1 \in \mathbb{Z}_2$, we have $\bigoplus_j r(j, 0, z_0) \neq \bigoplus_j r(j, 1, z_1)$.*
2. *For each $l \in \mathbb{Z}_2$, $\bigoplus_j r(j, l, 0) = \bigoplus_j r(j, l, 1)$.*

The following lemma establishes necessary and sufficient conditions on the third-qubit measurements for an N-regular quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ to be a paradox.

Lemma 4.5. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be an N-regular quantum scenario. It is a paradox if and only if $\delta(\lambda, C_0)$ and $\delta(\lambda, C_1)$ are* even *multiples of $\frac{\pi}{N}$ and $\beta(\lambda, C_0) - \beta(\lambda, C_1)$ is an* odd *multiple of $\frac{\pi}{N}$.*

Proof. Suppose that $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox. We follow the proof of Lemma 3.9, but in this case we can use Corollary 4.4 to deduce further restrictions. Let $l \in \mathbb{Z}_2$. After summing the N equations of Eq. 14 and then subtracting the N equations of Eq. 15, the coefficient of π on the RHS is

$$\bigoplus_{j=0}^{N-1} \left(a_j \oplus b_{K(j,l,0)}\right) \oplus \bigoplus_{j=0}^{N-1} \left(a_j \oplus b_{K(j,l,1)}\right) = \bigoplus_{j=0}^{N-1} (1 \oplus r(j, l, 0)) \oplus \bigoplus_{j=0}^{N-1} (1 \oplus r(j, l, 1)) = 0, \tag{19}$$

where the last equality is by Corollary 4.4.2. Thus, we in fact have $N\delta(\lambda, C_l) \equiv 0$ for each $l \in \mathbb{Z}_2$. Similarly, we can use Corollary 4.4.1 to deduce that $N(\beta(\lambda, C_0) - \beta(\lambda, C_1)) \equiv \pi$. Dividing each of these equations by N gives us the desired result.

Conversely, we can reverse the argument above to deduce that, for any $z_0, z_1 \in \mathbb{Z}_2$, the linear system $\Psi(z_0, z_1)$ satisfies $\bigoplus_j r(j, 0, z_0) \oplus \bigoplus_j r(j, 1, z_1) = 1$. By Lemma 4.2, $\Psi(z_0, z_1)$ is inconsistent, and Lemma 3.4 then implies that $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox. □

Therefore, an N-regular quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox if and only if the third-qubit measurements satisfy certain conditions. Below we list multiple families of quantum scenarios that satisfy these conditions and in the next subsection, we prove that these exhaust all N-regular paradoxes.

Theorem 4.6. *Let $N \in \mathbb{N}$, $N \geq 2$, and let $M_3 = \{C_0, C_1\}$. Then, $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox whenever the parameters $\lambda, C_0, C_1, M_1, M_2$ satisfy one of the following conditions:*

(a) $N = 2$: The parameters describe the standard GHZ paradox.

(b) $N > 2$, N is even: There exist $s, t \in \mathbb{Z}$, with s even and t odd, satisfying $1 \leq s < N$ and $1 \leq t \leq s-1$, such that

$$C_0 = 0, \quad \delta(\lambda, C_1) \equiv \frac{\pi}{N} s, \quad \beta(\lambda, C_1) \equiv -\frac{\pi}{N} t. \tag{20}$$

Additionally, $M_1 = M_2 = \frac{\pi}{N}\mathbb{Z}_N$. In particular, if $t = s/2$, then $\lambda = \frac{\pi}{2} - \frac{\pi}{N} t$ and $C_1 = \frac{\pi}{2}$.

(c) There exist $s, t' \in \mathbb{Z}$, with s even, satisfying $1 \leq s < N$ and $0 \leq t' \leq s/2 - 1$, such that

$$\begin{gathered} C_1 = \pi - C_0, \quad \delta(\lambda, C_0) \equiv \delta(\lambda, C_1) \equiv \frac{\pi}{N} s, \\ \beta(\lambda, C_0) \equiv -\frac{\pi}{N}(t' + \tfrac{1}{2}), \quad \beta(\lambda, C_1) \equiv -\frac{\pi}{N}(s - t' - \tfrac{1}{2}). \end{gathered} \tag{21}$$

Additionally, $M_1 = \frac{\pi}{N}\mathbb{Z}_N$, and $M_2 = \frac{\pi}{N}(\mathbb{Z}_N + \frac{1}{2})$.

(d) There exist $s, s', t' \in \mathbb{Z}$, with s, s' even and t' odd, satisfying $1 \leq s < s' < N$ and $1 \leq |t'| \leq N - 1$, such that

$$\delta(\lambda, C_0) \equiv \frac{\pi}{N} s, \quad \delta(\lambda, C_1) \equiv \frac{\pi}{N} s', \quad \beta(\lambda, C_1) - \beta(\lambda, C_0) \equiv \frac{\pi}{N} t'. \tag{22}$$

Additionally, let $\beta(\lambda, C_0) \equiv -\frac{\pi}{N} t$ $(0 < t < s)$ and $\nu := \lceil t \rceil - t$. Then $M_1 = \frac{\pi}{N}\mathbb{Z}_N$, and $M_2 = \frac{\pi}{N}(\mathbb{Z}_N + \nu)$.

Proof. In each case, the quantum scenario $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is maximally impossible and $|M_1| = |M_2| = N$. Moreover, the conditions on δ and β from Lemma 4.5 are satisfied, so for any $z_0, z_1 \in \mathbb{Z}_2$, the linear system $\Psi(z_0, z_1)$ satisfies $\bigoplus_j r(j, 0, z_0) \oplus \bigoplus_j r(j, 1, z_1) = 1$. Therefore, by Remark 4.3, $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox. □

Except for case (a) and the specific instance of case (b) where $t = s/2$, the existence of parameters λ, C_0, C_1 satisfying the conditions (b)–(d) of the above theorem is nontrivial. The existence of such parameters is established in Theorem 4.13 in the following subsection.

These paradoxes can be visualised by picturing the measurement angles of M_1, M_2 and the values of $\beta(\lambda, C_l)$ for $l \in \mathbb{Z}_2$ as points distributed around a circle. For example, Figure 1 visualises two different paradoxes satisfying condition (b) with $N = 8$, $C_1 = \frac{\pi}{2}$ and $t = s/2 = 1$, 3. The measurements in M_1 and M_2 are evenly spread modulo π, with each connected by a dotted line to its diametric opposite, corresponding to flipping the outcome. In the third circle, each pair of values $\beta, \beta + \delta$, which are joined by dotted line segments of the same colour, is symmetric under reflection about the x-axis. For different s and t, the spacing between these pairs and their β values varies but remains an integer multiple of $\frac{\pi}{N}$. See Figure 3 for illustrations of the paradoxes in the other cases.

In case (b), when $t = s/2$, the paradoxes take the form $(|\mathrm{B}(\lambda_{N,t})\rangle, \mathscr{M})$, where $\lambda_{N,t} = \frac{\pi}{2} - \frac{\pi}{N} t$ for all even N and odd t satisfying $1 \leq t < N/2$. The third measurement set is $\{0, \frac{\pi}{2}\}$. This family is an extension of [1, Theorem 8], which considered $\lambda_N = \frac{\pi}{2} - \frac{\pi}{N}$ indexed by even $N \in \mathbb{N}$. Our construction, indexed by a tuple (N, t), corresponds to a subset of the rationals and includes infinitely more paradoxes.

Remarkably, the other cases in Theorem 4.6 exhibit paradoxes from stranger empirical models. In each of these cases, although the resulting values of the functions δ and β have relatively simple properties, the value of λ and the measurement angles on the third qubit are complicated and appear to be

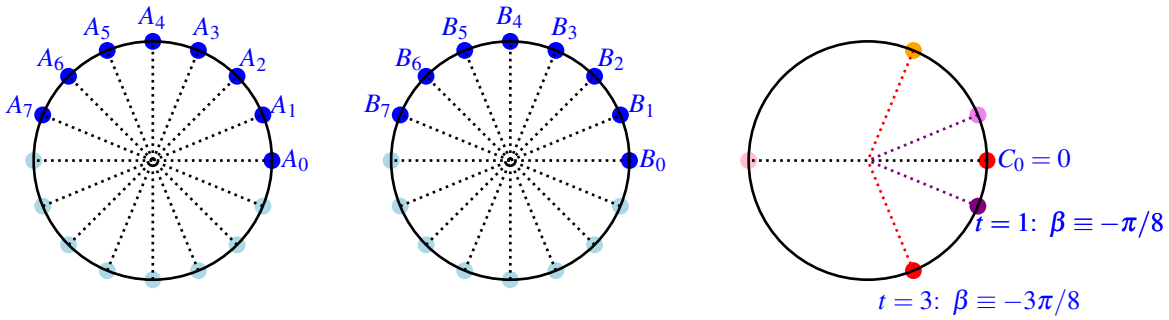


Figure 1: Two paradoxes satisfying condition (b), where $N = 8$, $C_1 = \frac{\pi}{2}$, and $t = s/2$, for the cases $t = 1, 3$.

very difficult to find analytically. It is as yet not even clear in these cases whether $\lambda \in \mathbb{Q}\pi$. Notably, the paradoxes in cases (c) and (d) do not involve the measurement angle 0, i.e. the Pauli X, which is a distinguishing feature absent in all previously known examples.

Theorem 4.7. *Every paradox described in Theorem 4.6 is distinct up to equivalence.*

The proof can be found in Appendix B.1.

4.2 A partial classification of three-qubit nonlocality paradoxes

We now derive a complete classification for all N-regular paradoxes. If a paradox does not come from a maximal rank quantum scenario, then the associated linear systems may have a more complex structure that warrants further investigation. See Subsection 4.3 for further discussion.

Before presenting the main theorem, we first establish a few key lemmas. First, Lemma 4.8 is about the measurements on the first two qubits in any N-regular paradox.

Lemma 4.8. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be an N-regular quantum scenario. Then the N measurement angles in both M_1 and M_2 are evenly spaced modulo π, with adjacent angles differing by $\frac{\pi}{N}$. Mathematically, we have (up to equivalence) $M_1 = \frac{\pi}{N}\mathbb{Z}_N$ and $M_2 = \frac{\pi}{N}\mathbb{Z}_N + \mu$ for some $\mu \in [0, \pi)$.*

Proof. For each $j \in \mathbb{Z}_N$, consider the two equations

$$(A_j + B_{K(j,l,0)}) \equiv \beta(\lambda, C_l) + (1 \oplus a_j \oplus b_{K(j,l,0)})\pi \quad \text{for } l \in \mathbb{Z}_2, \tag{23}$$

which determine the impossible events on the contexts $(A_j, B_{K(j,l,0)}, C_l)$. Subtracting the $l = 1$ equation from the $l = 0$ equation gives us $B_{K(j,0,0)} - B_{K(j,1,0)} \equiv \beta(\lambda, C_0) - \beta(\lambda, C_1) \mod \pi$, i.e. the measurements $B_{K(j,0,0)}$ and $B_{K(j,1,0)}$ differ by a multiple of $\frac{\pi}{N}$. For each j, Eq. 23 gives us a pair of measurements $\{B_{K(j,0,0)}, B_{K(j,1,0)}\}$. Since $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is maximally impossible, each measurement appears in exactly two such pairs. By the maximal rank property, it is possible to order these pairs to form a chain $\{B_{k_0}, B_{k_1}\}$, $\{B_{k_1}, B_{k_2}\}$, …, $\{B_{k_{N-2}}, B_{k_{N-1}}\}$, $\{B_{k_{N-1}}, B_{k_0}\}$ consisting of all N measurements on the second qubit. By taking linear combinations of the difference equations associated with these pairs, we deduce that all second-qubit measurement angles differ by multiples of $\frac{\pi}{N}$. Since there are N measurements, the result for M_2 follows. The same result holds for M_1 by symmetry. □

The next two lemmas are used to show that the four cases (a)–(d) in Theorem 4.6 exhaust all N-regular paradoxes. Note that, in cases (a) and (b), one of the third-qubit measurements is X ($C_0 = 0$); then a necessary and sufficient condition for a paradox is that λ and the second measurement C_1 be such that $\delta(\lambda, C_1)$ is an even multiple of $\frac{\pi}{N}$ and $\beta(\lambda, C_1)$ *itself* is an odd multiple of $\frac{\pi}{N}$ (since $\beta(\lambda, C_0) = 0$). Next,

assuming that neither third-qubit measurement is X, in case (c) we have that $\delta(\lambda, C_l)$ is the same for each $l \in \mathbb{Z}_2$. To find all paradoxes with this property, we show in Lemma 4.10 that, given $\frac{\pi}{N}s$, with s even, and $\lambda \in \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$, there are at most two different measurement angles C_0, C_1 that satisfy $\delta(\lambda, C_l) \equiv \frac{\pi}{N}s$, and furthermore $C_1 = \pi - C_0$. Then, by imposing the requirement that $\beta(\lambda, C_0) - \beta(\lambda, C_1)$ be an odd multiple of $\frac{\pi}{N}$, we can completely classify all the N-regular paradoxes with equal values of $\delta(\lambda, C_l)$.

Lemma 4.9. *For $\lambda \in \left[0, \frac{\pi}{2}\right)$ and $C \in [0, \pi)$, the equations $\delta(\lambda, C) \equiv \pi$ and $\beta(\lambda, C) \equiv 0$ are satisfied simultaneously if and only if either $\lambda = 0$ or $C = 0$.*

Proof. Follows from the properties of the functions β and δ. □

Lemma 4.10. *Given $N, s \in \mathbb{N}$, $N > 2$, $1 \leq s < N$, and $t \in \mathbb{R}$, $0 < t < s$, there exist unique $\lambda = \lambda(s,t) \in \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$ and $C = C(s,t) \in (0, \pi)$ such that*

$$\delta(\lambda, C) \equiv \frac{\pi}{N}s \quad \text{and} \quad \beta(\lambda, C) \equiv -\frac{\pi}{N}t. \tag{24}$$

Furthermore,

(i) $\lambda(s, \frac{s}{2}) = \frac{\pi}{2} - \frac{\pi}{2N}s$ *and* $C(s, \frac{s}{2}) = \frac{\pi}{2}$.

(ii) For a fixed s, $\lambda(s, -)$ is two-to-one on $(0, s)$, with $\lambda(s, s-t) = \lambda(s,t)$.

(iii) For a fixed s, $C(s, -)$ is bijective on $(0, s)$, with $C(s, s-t) = \pi - C(s,t)$.

Proof. For $s < N$, we note that for a fixed λ, $\delta(\lambda, -)$ has a minimum (modulo 2π) of $\pi - 2\lambda$ at $C = \frac{\pi}{2}$. Thus Eq. 24 is satisfied only if $\lambda \geq \frac{\pi}{2} - \frac{\pi}{2N}s$.

Let $\lambda := \lambda(s,t)$ and $C := C(s,t)$. Then,

$$\delta(\lambda, C) \equiv \frac{\pi}{N}s \quad \Longleftrightarrow \quad \pi - 2\arctan(\sin C \tan\lambda) \equiv \frac{\pi}{N}s \quad \Longleftrightarrow \quad \sin C = \frac{\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)}{\tan\lambda}. \tag{25}$$

Let us set $C^0(s,\lambda) := \arcsin\left(\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)/\tan\lambda\right)$ and $C^1 := \pi - C^0$, noting that C^0 and C^1 both satisfy Eq. 25 for C. For a fixed s, we have that $C^0(s, -)$ and $C^1(s, -)$ are functions of λ that give us the at most two different angles C such that $\delta(\lambda, C) \equiv \frac{\pi}{N}s$. Substituting C^0 into the expression for β (Eq. 3), we get

$$\beta\left(\lambda, C^0(s,\lambda)\right) = \arcsin\left(\cot\left(\frac{\pi}{2N}s\right)\cot\lambda\right) - 2\arctan\left(\frac{\cos\frac{\lambda}{2}\cot\left(\frac{\pi}{2N}s\right)\cot\lambda}{\sqrt{1 - \cot^2\left(\frac{\pi}{2N}s\right)\cot^2\lambda}\cos\frac{\lambda}{2} + \sin\frac{\lambda}{2}}\right). \tag{26}$$

Let $t^0(s,\lambda) := -\frac{N}{\pi}\beta(\lambda, C^0)$. By Lemma 4.11 below, $t^0(s, -)$ has a well-defined inverse $\lambda^0(s, -) : (0, \frac{s}{2}] \to \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$. Furthermore, from the definition of β, we have $\beta(\lambda, \pi + C^0) \equiv -\beta(\lambda, \pi - C^0)$. Thus

$$\beta(\lambda, C^0) + \beta(\lambda, C^1) \equiv -\delta(\lambda, C^0) \equiv -\frac{\pi}{N}s, \tag{27}$$

so $\beta(\lambda, C^1) \equiv -\frac{\pi}{N}(s-t)$ if and only if $\beta(\lambda, C^0) \equiv -\frac{\pi}{N}t$.

Putting all this together, we can construct λ and C as functions of s and t as follows:

$$\lambda(s,t) = \begin{cases} \lambda^0(s,t) & \text{if } 0 < t \leq s/2, \\ \lambda^0(s, s-t) & \text{if } s/2 < t < s; \end{cases} \tag{28}$$

$$C(s,t) = \begin{cases} \arcsin\left(\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)/\tan\left(\lambda^0(s,t)\right)\right) & \text{if } 0 < t \leq s/2, \\ \pi - \arcsin\left(\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)/\tan\left(\lambda^0(s,s-t)\right)\right) & \text{if } s/2 < t < s. \end{cases} \tag{29}$$

The above reasoning shows that these are well-defined. Hence we have proved the statement of the lemma. Statement (i) follows by noting that $\delta(\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}) \equiv \frac{\pi}{N}s$ and $\beta(\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}) \equiv -\frac{\pi}{2N}s$. Statements (ii) and (iii) immediately follow from the constructions of $\lambda(s,t)$ and $C(s,t)$. □

Lemma 4.11. *Let $t^0(s,\lambda) := -\frac{N}{\pi}\beta(\lambda, C^0)$. The function $t^0(s,-)$ is strictly decreasing, and hence bijective, on $\left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$. Furthermore, $t^0\left(s, \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)\right) = (0, \frac{s}{2}]$.*

Proof. We have $\beta(\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}) = -\frac{\pi}{2N}s$, so $t^0(s, \frac{\pi}{2} - \frac{\pi}{2N}s) = \frac{s}{2}$. Next, the expression for $\beta(\lambda, C^0)$ in Eq. 26 shows that $t^0(s,-)$ has no discontinuities for $\lambda \in \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$, and that $t^0(s,\lambda) \to 0$ as $\lambda \to \frac{\pi}{2}^-$. To show that $t^0(s,-)$ is strictly decreasing on the given interval, one can find the partial derivative of Eq. 26 with respect to λ and derive that

$$\frac{\partial\beta(\lambda,C^0)}{\partial\lambda} = 0 \quad \Longleftrightarrow \quad \sqrt{\frac{\cos^2\left(\frac{\pi}{2N}s\right) - \sin^2\lambda}{\left(\cos^2\left(\frac{\pi}{2N}s\right) - 1\right)\sin^2\lambda}} = -\frac{1}{\sin\lambda}, \tag{30}$$

which is never the case since $-1/\sin\lambda < 0$ for $\lambda \in \left[\frac{\pi}{2} - \frac{\pi}{2N}s, \frac{\pi}{2}\right)$. Thus, $t^0(s,-)$ is strictly monotone on the given interval. □

Remark 4.12. By the analysis in the two proofs above of the image of t^0 and the relationship between $\beta(\lambda, C^0)$ and $\beta(\lambda, C^1)$, we deduce that Eq. 24 is not satisfied if $s \leq t < N$.

Figure 2 provides a visualisation of these technical lemmas, showing how the value of t relative to s affects the values of λ and C that satisfy Eq. 24.

We are now ready to prove the main theorem classifying all N-regular paradoxes.

Theorem 4.13. *Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ be an N-regular paradox. Then it is equivalent to one of the paradoxes described in Theorem 4.6. Furthermore:*

(i) Condition (b) is satisfied by a unique N-regular paradox for each $(s,t) \in \mathbb{Z}^2$, with s even and t odd, satisfying $1 \leq s < N$ and $1 \leq t \leq s-1$. Condition (c) is satisfied by a unique N-regular paradox for each $(s,t') \in \mathbb{Z}^2$, with s even, satisfying $1 \leq s < N$ and $0 \leq t' \leq s/2 - 1$.

(ii) Condition (d) is satisfied by at least one N-regular paradox for some N and $s, s', t' \in \mathbb{Z}$, with s, s' even and t' odd, satisfying $1 \leq s < s' < N$ and $1 \leq |t'| \leq N-1$.

Proof. By Lemma 4.5, $\delta(\lambda, C_l)$ is an even multiple of $\frac{\pi}{N}$ and $\beta(\lambda, C_0) - \beta(\lambda, C_1)$ is an odd multiple of $\frac{\pi}{N}$. We analyse four different cases that exhaust all possibilities for N, C_0, C_1, and $\delta(\lambda, C_l)$. In each case, the sets M_1, M_2 are determined by Lemma 4.8 and the requirement to satisfy the model's impossible event equations.

Suppose that $N = 2$. Then $\delta(\lambda, C_l) \equiv \pi$ for both $l = 0, 1$, which is the case if and only if $\lambda = 0$. Then, since $\beta(0, C_0) - \beta(0, C_1) \equiv \frac{\pi}{2}$, we have $C_1 - C_0 = \frac{\pi}{2}$. Hence the paradox satisfies condition (a) in Theorem 4.6.

Now suppose that $N > 2$. First, we check the case where $C_0 = 0$. Note that $\delta(\lambda, 0) \equiv \pi$, which is an even multiple of $\frac{\pi}{N}$ if and only if N is even. Furthermore, $\beta(\lambda, 0) = 0$ for all λ. Consequently, any C_1 that satisfies Eq. 20 will ensure that the difference $\beta(\lambda, C_0) - \beta(\lambda, C_1)$ is an odd multiple of $\frac{\pi}{N}$. By Lemma 4.10, there exist unique satisfying λ and C_1 for each $(s,t) \in \mathbb{Z}^2$, with s even and t odd, satisfying

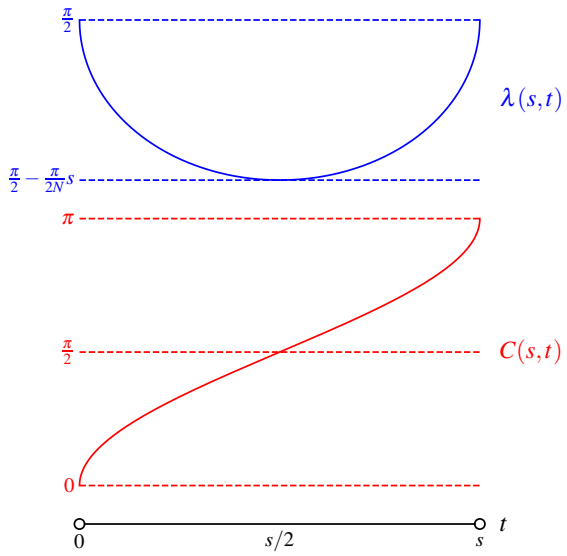


Figure 2: A visualisation of Lemma 4.10. For a fixed integer s with $1 \leq s < N$, t can take any real value greater than 0 and less than s. The two graphs above the interval line for t show the unique values of λ and C satisfying Eq. 24 for each t. Note the even symmetry of $\lambda(s,t)$ and the odd symmetry of $C(s,t)$, both about $t = s/2$.

$1 \leq s < N$ and $1 \leq t \leq s-1$. This paradox satisfies condition (b). By Remark 4.12, we cannot have a paradox if $s < t < N$.

Next, consider when both C_0 and C_1 are nonzero and λ, C_0 and C_1 are such that $\delta(\lambda,C_0) \equiv \delta(\lambda,C_1) \equiv \frac{\pi}{N}s$ for $1 \leq s < N$ and s even. We additionally need the $\beta(\lambda,C_l)$'s to differ by an odd multiple of $\frac{\pi}{N}$. Let t be such that $\beta(\lambda,C_0) \equiv -\frac{\pi}{N}t$. Lemma 4.10, in particular statements (ii) and (iii), and the paragraph of the proof containing Eq. 27, tell us that $C_1 = \pi - C_0$ and $\beta(\lambda,C_1) \equiv -\frac{\pi}{N}(s-t)$. Hence $\beta(\lambda,C_0) - \beta(\lambda,C_1) \equiv \frac{\pi}{N}(s-2t)$. Since $s-2t$ is odd if and only if $2t$ is odd, t must be a half-integer. Thus this paradox satisfies condition (c). The bounds on t' follow from Remark 4.12.

Finally, consider when both C_0 and C_1 are nonzero and $\delta(\lambda,C_0) \not\equiv \delta(\lambda,C_1)$. Referring to Eq. 25, a triple (λ,C_0,C_1) satisfying condition (d) exists only if $C_0 = \arcsin\left(\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)/\tan\lambda\right)$ or $\pi - \arcsin\left(\tan(\frac{\pi}{2} - \frac{\pi}{2N}s)/\tan\lambda\right)$, and similarly for C_1 (replacing s with s'). Substituting these into $\beta(\lambda,C_0) - \beta(\lambda,C_1)$, we obtain a very complicated function that is difficult to analyse. However, solutions to Eq. 22 with these substitutions do exist for some values of N, s, s' and t'[1]. □

4.3 A path to a complete classification of three-qubit nonlocality paradoxes

In the previous subsections, we detailed a comprehensive investigation of paradoxes that are N-regular, i.e. maximally impossible, maximal rank paradoxes with two third-qubit measurements. The next step for the classification program is to drop some or all of these assumptions.

We observe that there exist maximally impossible paradoxes with $|M_3| = 2$ that are not maximal rank. For example, let $N = 4$. There exist λ,C_0,C_1 such that $\delta(\lambda,C_l) \equiv \frac{3\pi}{4}$ for $l \in \mathbb{Z}_2$, $\beta(\lambda,C_0) \equiv -\frac{\pi}{4}$,

[1] E.g. if $N = 8$, $s = 4$, $s' = 6$ and $t' = 1$, one can numerically find the solution $\lambda \approx 0.8129$, $C_0 \approx 1.2418$, $C_1 \approx 0.4028$.

and $\beta(\lambda, C_1) \equiv -\frac{\pi}{2}$[2]. Note that here $\delta(\lambda, C_l)$ is an odd multiple of $\frac{\pi}{N}$. As usual, let $M_1 = M_2 = \frac{\pi}{4}\mathbb{Z}_4$. See Figure 3 for an illustration. In this case, the four $\mathbb{Z}_2$-linear systems $\Psi(z_0, z_1)$ are all inconsistent, so that $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ is a paradox. However, the parities $\bigoplus_j r(j,l,0)$ and $\bigoplus_j r(j,l,1)$ are not equal for either $l \in \mathbb{Z}_2$. Thus, a parity argument akin to the proof of Lemma 4.2, which considers all equations, is not applicable to two of the systems, which necessitate a parity argument using a proper subset of the equations. Further study is needed to determine the exact conditions for maximally impossible quantum scenarios to be paradoxes.

We have shown that there exist paradoxes $(|\mathrm{B}(\lambda_{N,t})\rangle, \mathscr{M})$ where $\lambda = \frac{\pi}{2} - \frac{\pi}{N}t$, $1 \leq t < N/2$, for all N even and t odd. However, there do not exist paradoxes $(|\mathrm{B}(\lambda_{N,t})\rangle, \mathscr{M})$ such that $|M_3| = 2$ and $\lambda = \frac{\pi}{2} - \frac{\pi}{N}t$ for *N odd*. It remains to be determined whether a paradox, with such a λ or otherwise, is possible if we expand the measurement set M_3 to have more than two measurements.

Let us now examine general balanced states $|\mathrm{B}(\vec{\lambda}, \Phi)\rangle$ and seek necessary conditions on the λ_i's for $(|\mathrm{B}(\vec{\lambda}, \Phi)\rangle, \mathscr{M})$ to be a paradox. A key observation from our analysis of interpolant states is that the number of impossible events is maximised in each paradox that we have found. If an empirical model does not have many impossible events across its contexts, the likelihood of a consistent global assignment intuitively seems higher. So, we expect that for a paradox to be witnessed, the empirical model must contain a sufficiently large number of impossible events.

The following lemma characterises empirical models where there are at least three contexts each containing four impossible events.

Lemma 4.14. *Let $(|\mathrm{B}(\vec{\lambda}, \Phi)\rangle, \mathscr{M})$ be a quantum scenario such that at least three contexts contain four impossible events each. Then $|\mathrm{B}(\vec{\lambda}, \Phi)\rangle$ must be an interpolant state.*

The proof can be found in Appendix B.2.

Along with Lemma 4.14, we observe other facts that also greatly restrict $\vec{\lambda}$ as long as we have sufficiently many impossible events across different contexts. For example, suppose $(|\mathrm{B}(\vec{\lambda}, \Phi)\rangle, \mathscr{M})$ is a quantum scenario in which a measurement A is in two contexts that both contain at least two impossible events, related by flipping the outcome of A and another measurement on the same qubit, say i. Then $\lambda_i = 0$.

This evidence strongly suggests that two of the λ_i's must be zero for $(|\mathrm{B}(\vec{\lambda}, \Phi)\rangle, \mathscr{M})$ to be a paradox. This leads us to propose the following conjecture.

Conjecture 4.15. *For the quantum scenario $(|\mathrm{B}(\vec{\lambda}, \Phi)\rangle, \mathscr{M})$ to be a paradox, $|\mathrm{B}(\vec{\lambda}, \Phi)\rangle$ must be an interpolant state.*

Acknowledgments

ND acknowledges support from the Canada Research Chair program, NSERC Discovery Grant RGPIN-2022-03103, the NSERC-European Commission project FoQaCiA, and the Faculty of Science of Simon Fraser University. SJ acknowledges support from the NSERC-European Commission project FoQaCiA.

References

[1] Samson Abramsky, Rui Soares Barbosa, Giovanni Carù, Nadish de Silva, Kohei Kishida & Shane Mansfield (2018): *Minimum Quantum Resources for Strong Non-Locality*. In Mark M. Wilde, editor: *12th Conference*

[2] Numerically, $\lambda \approx 0.4271$, $C_0 = \pi - C_1 \approx 1.1437$.

on the Theory of Quantum Computation, Communication and Cryptography (TQC 2017), *Leibniz International Proceedings in Informatics (LIPIcs)* 73, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 9:1–9:20, doi:10.4230/LIPIcs.TQC.2017.9.

[2] Samson Abramsky & Adam Brandenburger (2011): *The sheaf-theoretic structure of non-locality and contextuality*. *New Journal of Physics* 13(11), p. 113036, doi:10.1088/1367-2630/13/11/113036.

[3] Antonio Acín & Lluis Masanes (2016): *Certified randomness in quantum physics*. *Nature* 540(7632), pp. 213–219, doi:10.1038/nature20119.

[4] Janet Anders & Dan E. Browne (2009): *Computational Power of Correlations*. *Physical Review Letters* 102(5), doi:10.1103/physrevlett.102.050502.

[5] Jonathan Barrett, Adrian Kent & Stefano Pironio (2006): *Maximally Nonlocal and Monogamous Quantum Correlations*. *Physical Review Letters* 97(17), doi:10.1103/physrevlett.97.170409.

[6] John S. Bell (1964): *On the Einstein Podolsky Rosen paradox*. *Physics Physique Fizika* 1(3), pp. 195–200, doi:10.1103/PhysicsPhysiqueFizika.1.195.

[7] John S. Bell (1966): *On the Problem of Hidden Variables in Quantum Mechanics*. *Reviews of Modern Physics* 38(3), pp. 447–452, doi:10.1103/RevModPhys.38.447.

[8] G. Brassard, A.A. Methot & A. Tapp (2005): *Minimum entangled state dimension required for pseudo-telepathy*. *Quantum Information and Computation* 5(4 & 5), pp. 275–284, doi:10.26421/qic5.45-2.

[9] Sergey Bravyi, David Gosset & Robert König (2018): *Quantum advantage with shallow circuits*. *Science* 362(6412), pp. 308–311, doi:10.1126/science.aar3106.

[10] John F. Clauser, Michael A. Horne, Abner Shimony & Richard A. Holt (1969): *Proposed Experiment to Test Local Hidden-Variable Theories*. *Phys. Rev. Lett.* 23, pp. 880–884, doi:10.1103/PhysRevLett.23.880.

[11] R. Cleve, P. Hoyer, B. Toner & J. Watrous (2004): *Consequences and limits of nonlocal strategies*. In: *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pp. 236–249, doi:10.1109/CCC.2004.1313847.

[12] Roger Colbeck (2009): *Quantum and relativistic protocols for secure multi-party computation*. *arXiv preprint arXiv:0911.3814*.

[13] Avshalom C. Elitzur, Sandu Popescu & Daniel Rohrlich (1992): *Quantum nonlocality for each pair in an ensemble*. *Physics Letters A* 162(1), pp. 25–28, doi:10.1016/0375-9601(92)90952-i.

[14] Federico Grasselli, Gláucia Murta, Hermann Kampermann & Dagmar Bruß (2023): *Boosting device-independent cryptography with tripartite nonlocality*. *Quantum* 7, p. 980, doi:10.22331/q-2023-04-13-980.

[15] Daniel M. Greenberger, Michael A. Horne, Abner Shimony & Anton Zeilinger (1990): *Bell's theorem without inequalities*. *American Journal of Physics* 58(12), pp. 1131–1143, doi:10.1119/1.16243.

[16] Daniel M. Greenberger, Michael A. Horne & Anton Zeilinger (1989): *Going Beyond Bell's Theorem*, pp. 69–72. Springer Netherlands, doi:10.1007/978-94-017-0849-4_10.

[17] Robert Raussendorf (2013): *Contextuality in measurement-based quantum computation*. *Physical Review A* 88(2), doi:10.1103/physreva.88.022322.

[18] Robert Raussendorf (2023): *Putting Paradoxes to Work: Contextuality in Measurement-Based Quantum Computation*, pp. 595–622. Springer International Publishing, doi:10.1007/978-3-031-24117-8_16.

A Illustrations of the paradoxes

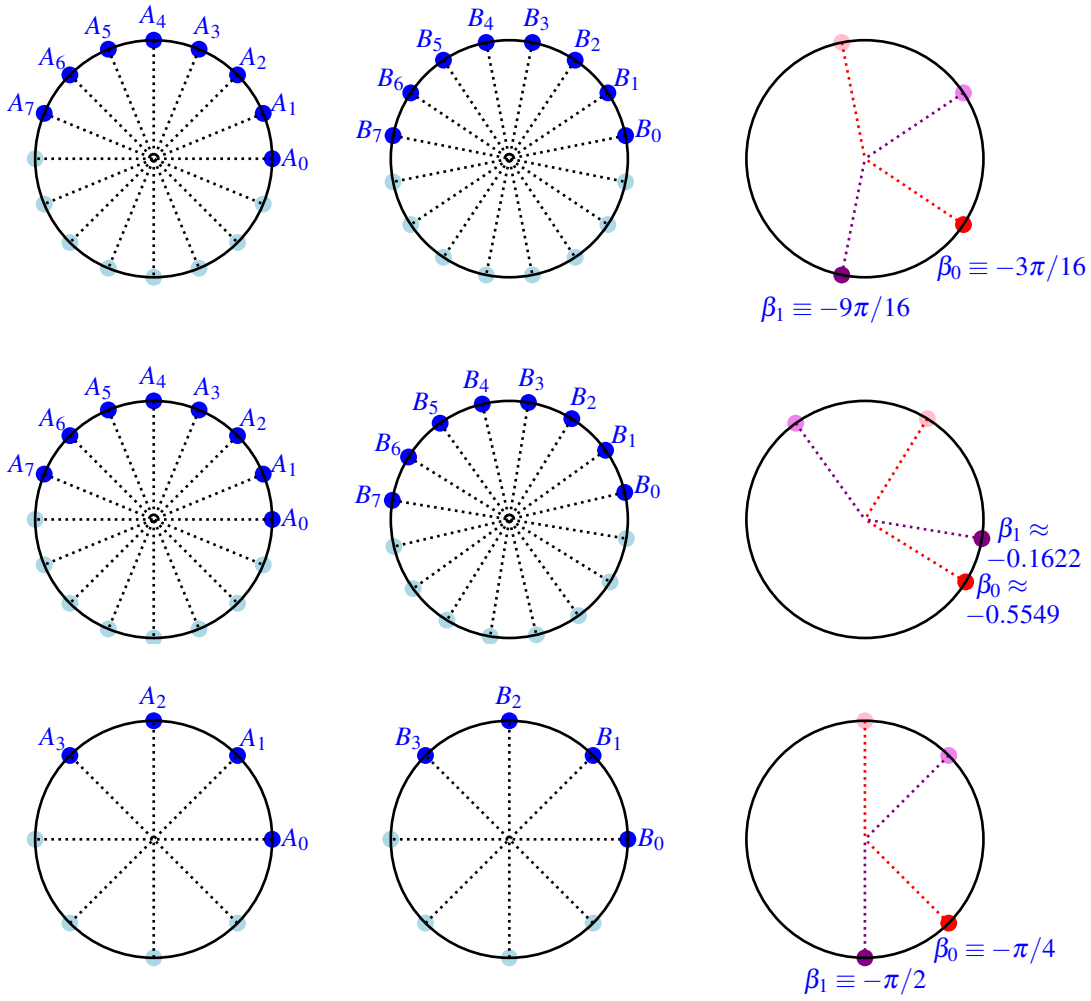


Figure 3: From top to bottom: a paradox satisfying condition (c) of Theorem 4.6, where $N = 8$, $s = 6$, and $t' = 1$; a paradox satisfying condition (d), where $N = 8$, $s = 4$, $s' = 6$, and $t' = 1$; a non-N-regular paradox satisfying $N = 4$, $\delta(\lambda, C_l) \equiv \frac{3\pi}{4}$ for $l \in \mathbb{Z}_2$, $\beta(\lambda, C_0) \equiv -\frac{\pi}{4}$, and $\beta(\lambda, C_1) \equiv -\frac{\pi}{2}$.

We present some more graphical illustrations of the paradoxes. In all our examples $|M_1| = |M_2| = N$, and $M_3 = \{C_0, C_1\}$. The existence of impossible events within a given context (A, B, C) depends on whether one of the following conditions holds:

$$A + B \equiv \beta(\lambda, C) \quad \text{or} \quad \beta(\lambda, C) + \pi \tag{31}$$

$$A + B \equiv \beta(\lambda, C + \pi) \quad \text{or} \quad \beta(\lambda, C + \pi) + \pi. \tag{32}$$

Intuitively, given a $C_l \in M_3$, many of these equations must hold for various choices of (A, B) in order to witness a paradox. By identifying $\frac{\pi}{N}\mathbb{Z}_N$ with $\mathbb{Z}_N$, these β values can be interpreted as 'ticks' of a clock,

which are achievable by $A+B$ for many different pairs (A,B). In [1, Theorem 8], all the paradoxes had ticks of ± 1 for $C_1 = \frac{\pi}{2}$. For the case $t = s/2$ in the family (b) of Theorem 4.6, the ticks are $\pm t$. Notably, in all these examples, the ticks remain symmetric about the x-axis (see Figure 1). Furthermore, for $C_0 = 0$, the corresponding ticks are precisely 0 and π. When $t \neq s/2$ in the family (b) in Theorem 4.6, the ticks for C_1 become asymmetric, specifically $-t$ and $s-t$, although we still observe a sort of symmetry about the x-axis if we view the ticks as indistinguishable. The other families in Theorem 4.6 exhibit even more intricate structures, as their tick values are nonzero for both measurements and may not even be rational multiples of π (see Figure 3). However, we note that the *differences* between tick values (i.e. the angles between ticks) are still appropriate multiples of $\frac{\pi}{N}$.

B Proofs of select results

B.1 Proof of Theorem 4.7

Since equivalence is defined up to local unitaries and qubit permutations, and any qubit permutation preserves equivalence, we restrict our analysis to local unitaries.

Lemma B.1. *Let M and M' be sets of equatorial measurements with $|M|, |M'| \geq 2$. If $UMU^\dagger = M'$ mod π, then U must be of the form $X^a P_\theta$ (up to a global phase) for some $\theta \in [0, 2\pi)$ and $a \in \mathbb{Z}_2$, where P_θ is the phase gate.*

Proof. The map $Ad : SU(2) \to SO(3)$, given by $Ad(g)(V) = gVg^{-1}$ is a 2-fold covering of $SU(2) \to SO(3)$ with $\ker(Ad) = \{I, -I\}$. Therefore, Ad induces an isomorphism $PU(2) \cong SO(3)$, which establishes that single-qubit unitaries, up to a global phase, correspond to rotations of the Bloch sphere. The rotation $R_{\vec{n}}(\theta)$ of the Bloch sphere about an arbitrary axis $\vec{n} = (n_x, n_y, n_z)$ is given by

$$R_{\vec{n}}(\theta) = \exp\left(-i\frac{\theta}{2}(n_x X + n_y Y + n_z Z)\right). \tag{33}$$

An arbitrary Bloch sphere rotation can be expressed as $U = R_z(\theta_1)R_x(\gamma)R_z(\theta_2)$, where $\theta_1, \theta_2 \in [0, 2\pi)$ and $\gamma \in [0, \pi]$. Importantly, U maps any great circle on the Bloch sphere to another great circle, as it represents a rotation of the sphere. Let us denote the equator of the Bloch sphere by S. Note that $R_z(\theta)(S) = S$ for all θ and $R_x(\gamma)(S) = S$ only if $\gamma = 0$ or π. So, $U(S) \neq S$ when $\gamma \neq 0$ or π. Since S and $U(S)$ are great circles, they intersect precisely at two points separated by π. This shows that for U to map at least two equatorial measurements that differ by less than π to equatorial measurements, γ must be 0 or π. Therefore U must be of the form $R_z(\theta)$ or $R_z(\theta_1)R_x(\pi)R_z(\theta_2)$, which is equivalent (up to a global phase) to $X^a P_\theta$ for some $\theta \in [0, 2\pi)$ and $a \in \mathbb{Z}_2$. □

Lemma B.2. *Let $\lambda, \lambda' \neq 0$. If $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ and $(|\mathrm{B}(\lambda')\rangle, \mathscr{M}')$ are equivalent, then $\lambda = \lambda'$. Moreover, the local unitary U establishing the equivalence must be of the form*

$$U = P_{\theta_1} \otimes P_{\theta_2} \otimes I \quad \text{or} \quad XP_{\theta_1} \otimes XP_{\theta_2} \otimes X \tag{34}$$

such that $\theta_1 + \theta_2 \equiv 0$.

Proof. Let $(|\mathrm{B}(\lambda)\rangle, \mathscr{M})$ and $(|\mathrm{B}(\lambda')\rangle, \mathscr{M}')$ be equivalent via the local unitary U. By Lemma B.1, U must be of the form $e^{i\phi} \bigotimes_{j=1}^{3} X^{a_j} P_{\theta_j}$ for some $\phi, \theta_j \in [0, 2\pi)$, $a_j \in \mathbb{Z}_2$ for $j = 1, 2, 3$. Also, we have that

$$|\mathrm{B}(\lambda')\rangle = e^{i\phi} \bigotimes_{j=1}^{3} X^{a_j} P_{\theta_j} |\mathrm{B}(\lambda)\rangle. \tag{35}$$

Expanding Eq. 35, we have the following:

$$\begin{aligned}&\cos\frac{\lambda'}{2}|000\rangle+\sin\frac{\lambda'}{2}|001\rangle+\sin\frac{\lambda'}{2}|110\rangle+\cos\frac{\lambda'}{2}|111\rangle\\&=\cos\frac{\lambda}{2}e^{i\phi}|a_1a_2a_3\rangle+\sin\frac{\lambda}{2}e^{i(\theta_3+\phi)}|a_1a_2\overline{a}_3\rangle+\sin\frac{\lambda}{2}e^{i(\theta_1+\theta_2+\phi)}|\overline{a}_1\overline{a}_2a_3\rangle+\cos\frac{\lambda}{2}e^{i(\theta+\phi)}|\overline{a}_1\overline{a}_2\overline{a}_3\rangle,\end{aligned}\tag{36}$$

where $\overline{a}_j=a_j\oplus 1$ and $\theta=\theta_1+\theta_2+\theta_3$. It is easy to see that for Eq. 36 to hold, the only possible choices for (a_1,a_2,a_3) are $\{(0,0,0),\ (0,0,1),\ (1,1,0),\ (1,1,1)\}$. Since all coefficients are real on the LHS, that also must be the case on the RHS. So, $\phi=0$ or π. All the coefficients on the RHS are either $\pm\cos\frac{\lambda}{2}$ or $\pm\sin\frac{\lambda}{2}$. Comparing the amplitudes of the basis vector $|000\rangle$ on both sides, we have

$$\cos\frac{\lambda'}{2}=\pm\cos\frac{\lambda}{2}\quad\text{or}\quad\pm\sin\frac{\lambda}{2}.\tag{37}$$

In both cases of Eq. 37, the only solutions are $\lambda'=\lambda$ or $\lambda'=\pi-\lambda$. However, the second case is not feasible since $\lambda,\lambda'<\frac{\pi}{2}$. This further restricts the choice for (a_1,a_2,a_3) to either $(0,0,0)$ or $(1,1,1)$. For both choices, Eq. 36 is satisfied only if the following conditions hold:

$$\phi\equiv 0,\quad \theta_3+\phi\equiv 0,\quad \theta_1+\theta_2+\phi\equiv 0,\quad\text{and}\quad \theta_1+\theta_2+\theta_3+\phi\equiv 0.\tag{38}$$

This reduces U to the desired form. □

Proof of Theorem 4.7. The triples (λ,C_0,C_1) corresponding to the paradoxes in Theorem 4.6 are distinct for different paradoxes. By Lemma B.2, equivalent paradoxes must have the same λ value and third-qubit measurements can only be reflected about the X-axis. So, the only possible scenario where $(|\mathrm{B}(\lambda)\rangle,\mathscr{M})$ and $(|\mathrm{B}(\lambda')\rangle,\mathscr{M}')$ are equivalent is if $\lambda=\lambda'$ and C_l' is a reflection of C_l about the X-axis for both $l=0,1$. This is not possible since all measurements are in $[0,\pi)$. Hence, distinct triples (λ,C_0,C_1) correspond to inequivalent paradoxes. □

B.2 Proof of Lemma 4.14

Proof. By Lemma 3.3.1, a context (A,B,C) has 4 impossible events if and only if $\delta(\lambda_1,A)\equiv\delta(\lambda_2,B)\equiv\delta(\lambda_3,C)\equiv\pi$. Moreover, $\delta(\lambda,\varphi)\equiv\pi$ if and only if $\lambda=0$ or $\varphi=0$.

If one of the three contexts is such that all measurement angles are nonzero, i.e. $A,B,C\neq 0$, then it follows that $\lambda_i=0$ for all $i=1,2,3$, implying that $|\mathrm{B}(\vec{\lambda},\Phi)\rangle$ is the GHZ state.

Alternatively, if a context has two nonzero measurements, e.g. $(A,B,0)$ with $A,B\neq 0$, then we have $\lambda_1=\lambda_2=0$. When one of the contexts is $(0,0,0)$, it is impossible to derive any conclusions regarding the λ_i's. However, if the remaining two contexts each contain exactly one nonzero measurement, then the nonzero measurements must be on different qubits, such as $(A,0,0)$ and $(0,B,0)$. In this case, we also have $\lambda_1=\lambda_2=0$. Therefore $|\mathrm{B}(\vec{\lambda},\Phi)\rangle$ is an interpolant state. □

High-Precision Multi-Qubit Clifford+T Synthesis by Unitary Diagonalization

Mathias Weiden Justin Kalloor John Kubiatowicz
University of California, Berkeley
{mtweiden, jkalloor3, kubitron}@berkeley.edu

Ed Younis Costin Iancu
Lawrence Berkeley National Laboratory
{edyounis, cciancu}@lbl.gov

Resource-efficient and high-precision approximate synthesis of quantum circuits expressed in the Clifford+T gate set is vital for Fault-Tolerant quantum computing. Efficient optimal methods are known for single-qubit R_Z unitaries, otherwise the problem is generally intractable. Search-based methods, like simulated annealing, empirically generate low resource cost approximate implementations of general multi-qubit unitaries so long as low precision (Hilbert-Schmidt distances of $\varepsilon \geq 10^{-2}$) can be tolerated. These algorithms build up circuits that directly invert target unitaries. We instead leverage search-based methods to first approximately diagonalize a unitary, then perform the inversion analytically. This lets difficult continuous rotations be bypassed and handled in a post-processing step. Our approach improves both the implementation precision and run time of synthesis algorithms by orders of magnitude when evaluated on unitaries from real quantum algorithms. On benchmarks previously synthesizable only with analytical techniques like the Quantum Shannon Decomposition, diagonalization uses an average of 95% fewer non-Clifford gates.

1 Introduction

Recent small-scale demonstrations of error-corrected quantum memory signal significant progress toward the development of Fault-Tolerant (FT) quantum computers [1, 5, 40]. In this setting, programs must be expressed in universal FT gate sets, which usually consist of gates from the Clifford group and at least one non-Clifford gate for universality. The Clifford+T gate set $(H, S, CNOT, T)$ is one such gate set targeted by many quantum compilers [20, 41, 21].

Quantum compilers must use approximate unitary synthesis to ensure circuits are transpiled, or translated, into FT gate sets [23]. Synthesis algorithms must balance *resource efficiency* (e.g. non-Clifford gate count), *approximation accuracy* (error) and *target generality* (width and size of set of unitaries that can be taken as input). Existing approaches can be categorized along the principle *"resource efficiency, high precision, and generality: pick any two!"*.

Our work advances FT synthesis along these three axes by combining analytical and search-based methods. While existing algorithms attempt to directly invert target unitaries, our main insight is to diagonalize instead: we perform a discrete search until target unitaries are (approximately) diagonalized. This process reveals single-qubit rotations that are difficult to compile with search-based multi-qubit synthesis algorithms. Instead of looking for discrete implementations of these continuously parameterized gates, we leverage analytical techniques to implement them. Synthesis-by-diagonalization approximates unitaries with precisions that are orders of magnitude higher than other search-based multi-qubit synthesis algorithms without negatively impacting run time and while maintaining good resource efficiency.

We demonstrate how synthesis-by-diagonalization can implement unitaries that are out of reach for other synthesis algorithms. We retrofitted a simulated annealing-based synthesis algorithm [31] and trained Reinforcement Learning agents to perform synthesis-by-diagonalization. This approach

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 215–230, doi:10.4204/EPTCS.426.8

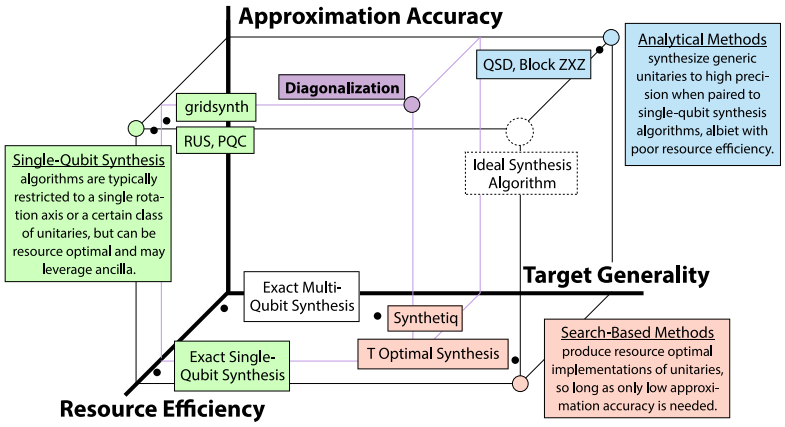


Figure 1: Tradeoffs for synthesis algorithms targeting the Clifford+T gate set. *Approximation accuracy* refers to the precision with which a target program is implemented. In the FT setting, this value must be high in order for program outputs to be meaningful. *Resource efficiency* refers to non-Clifford gate counts. As these gates are orders of magnitude more expensive than Clifford gates, synthesis algorithms should use as few non-Clifford gates as possible. *Target generality* refers to the size of the set of unitaries that can be taken as input. This ranges from exactly synthesizable unitaries and approximations of 1-qubit R_Z rotations, to the most general class of all multi-qubit unitaries.

is general, and other synthesis tools can also be modified to diagonalize rather than invert. We also deploy diagonalization in an end-to-end gate set transpilation workflow and demonstrate utility for actual algorithms. In particular, we observe up to an 18.1% reduction in T count. We believe our synthesis tools can help automate the discovery of gadgets that exploit ancilla to improve resource efficiency.

The remainder of the paper is organized as follows: Section 2 discusses necessary background information relating to synthesis with discrete gate sets. Section 4 explains how matrix diagonalization can be framed in the language of synthesis. Section 5 evaluates diagonalization by synthesizing unitaries taken from partitioned quantum algorithms. Section 6 discusses methods of scaling diagonalization. Finally, Section 7 offers concluding remarks.

2 Background

Fault-Tolerant (FT) quantum computing relies on Quantum Error Correction (QEC) to enable resilient quantum information processing [38]. Different QEC codes admit different gate sets. Gates in the Clifford group are classically simulable and often cheap to implement in FT architectures [25, 29]. For universal quantum computation, the logical gate set must also contain a non-Clifford operation [13]. This work will focus on the approximate compilation of unitaries into the Clifford+T gate set.

The T gate can be executed in a fault-tolerant manner through magic state distillation and injection with Clifford gate corrections [29]. This state distillation process is extremely costly. Estimates show that up to 99% of the resources on a quantum computer could be dedicated to implementing these operations [15, 30]. Minimizing the number of non-Clifford gates for a given implementation of a unitary is, therefore, an essential step in realizing practical FT quantum computation.

Quantum algorithms are often expressed using operations not directly compatible with QEC codes. Compilers require methods of transpiling into compatible gate sets. As multi-qubit unitary synthesis has proven to be a powerful tool for translating between gate sets in the Noisy Intermediate Scale Quantum (NISQ) era [33], it is a natural candidate for ensuring algorithms are suitable for FT machines [46]. This

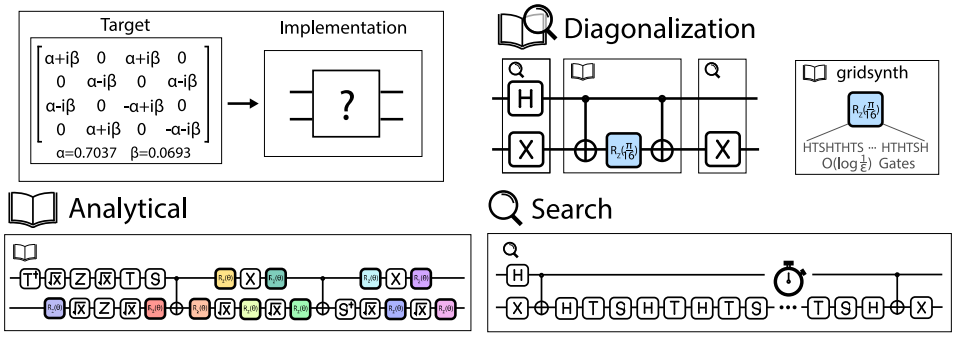


Figure 2: Comparison of analytical, search-based, and diagonalization strategies for Clifford+T synthesis. Analytical decomposition approaches, such as the Quantum Shannon Decomposition, are universally applicable but use many expensive non-Clifford resources. Search-based methods are able to find very low gate count approximations, but are intractable except at low precision. Diagonalization is a hybrid approach that enables both high quality and high precision. Search is used to diagonalize targets, then analytical methods are used to handle the diagonal results. *gridsynth* [36] is used to optimally decompose continuous single-qubit R_Z operations (colored boxes) into Clifford+T gates.

is a vital component of any FT compiler.

2.1 Fault-Tolerant Unitary Synthesis

If every element of an n-qubit unitary is in the ring $\mathbb{Z}[e^{i\pi/4}, 1/2]$, it can be implemented exactly with the Clifford+T gate set [24, 17]. Unitaries appearing in real algorithms are rarely this well structured; they often contain arbitrary angle rotations.

In the single-qubit case, unitaries are decomposed into Clifford $\sqrt{X}$ and non-Clifford $R_Z(\theta)$, so that

$$U = R_Z(\theta_1) \times \sqrt{X} \times R_Z(\theta_2) \times \sqrt{X} \times R_Z(\theta_3). \tag{1}$$

These R_Z rotations can be approximated *optimally* to arbitrary precision using the *gridsynth* algorithm from Ross and Selinger [36]. For some precision ε, optimal ancilla-free translation of R_Z gates into Clifford+T operations requires $O(\log 1/\varepsilon)$ T gates. This is true regardless of the rotation angle, except in select cases such as $\theta \in \{k\pi/4 : k \in \mathbb{Z}\}$. Ancilla-based methods can generate more resource-efficient approximations [26, 7, 6]. Previous work in circuit compilation demonstrates the utility of multi-qubit unitary synthesis [11, 42, 46, 45]. These methods discover approximate circuit implementations using numerical optimization and parameterized (non-FT) gates. Circuits synthesized this way can be straightforwardly transpiled into the Clifford+T gate set by decomposing parameterized single-qubit gates using Equation 1 and *gridsynth*. While these methods often find circuits with fewer two-qubit gates, they often result in circuits containing many R_Z (and therefore T) gates.

The Quantum Shannon Decomposition and its variants are also powerful synthesis algorithms. These analytical methods produce circuit implementations that approach the asymptotic lower bound of $O(4^n)$ CNOT and R_Z gates [37, 12]. Again, this technique results in many non-Clifford gates.

Another option is to deploy a synthesis algorithm that directly operates in the FT gate set and iteratively modifies a circuit, often gate-by-gate until the target unitary is implemented. These *search-based* multi-qubit approaches generate more resource-efficient circuits than the previously described generic analytical methods. Examples of such include simulated annealing [31], Reinforcement Learning approaches [47, 28, 10, 2, 35], and several optimal and heuristic synthesis algorithms [3, 18, 16]. However, solving this problem for optimal non-Clifford gate counts is NP Hard [44].

Method	Qubits	Category	Precision	Unitary Domain
gridsynth [36]	1	Analytical	-	Approx. R_Z
Policy Iter. [2]	1	Search (RL)	10^{-2}	Approx. $\mathbb{U}(2)$
Synthetiq [31]	1-4	Search (SA)	10^{-3}	Approx. $\mathbb{U}(2)$-$\mathbb{U}(16)$
Diagonalization (ours)	1-3	Search	10^{-3} and below	Approx. $\cup$ Diagonalizable $\mathbb{U}(2)$-$\mathbb{U}(8)$
QSD [37]	1+	Analytical	-	$U \in \mathbb{U}(2^n)$

Table 1: Unitary synthesis approaches for the Clifford+T gate set. Among search-based methods, our *diagonalizing* approach can synthesize targets to higher precision. In cases where unitaries can be exactly diagonalized, circuit implementations are produced to arbitrarily high precision. Precision here is measured using Hilbert-Schmidt distance (Eq 3). The *Unitary Domain* column indicates what kinds of unitaries can be handled. *Synthetiq* uses simulated annealing and empirically outperforms other pure search-based tools at low precision. *gridsynth* optimally decomposes 1-qubit R_Z unitaries to any precision. The *QSD* is an analytical method which can be used along with *gridsynth* to synthesize Clifford+T circuits.

Empirically, these state-of-the-art methods find very efficient implementations of multi-qubit unitaries so long as solutions require few (meaning 10s of) gates. Search-based tools are restricted in that they only operate with discrete gate sets, thus they have no direct way of handling continuous rotations such as R_Z gates. As a reference, synthesizing a single R_Z gate to a precision of $\varepsilon \leq 10^{-8}$ requires about 200 individual Clifford+T gates. For unitaries containing these rotations, which are ubiquitous in unitaries taken from benchmarks of interest, search-based tools can only find low-precision implementations for a subset of inputs.

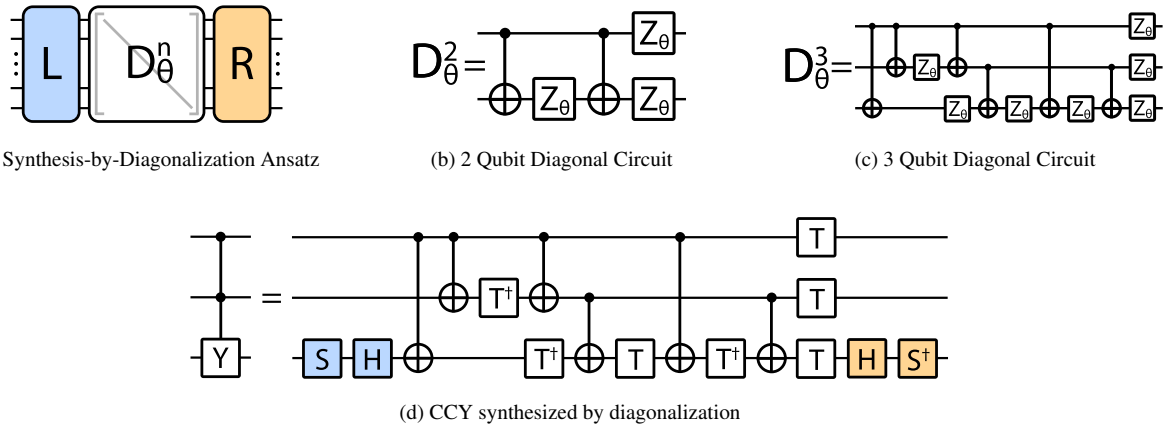
Table 1 compares various unitary synthesis approaches which can be used to target the Clifford+T gate set. Among search-based methods, the simulated annealing tool *Synthetiq* empirically finds better implementations of unitaries than other methods [31]. Even so, Synthetiq fails to produce solutions for high precision implementations of complex unitaries as the space of circuits is too large to search. *Synthesizing complex unitaries to high precision requires a mechanism for handling continuous rotations*; this is possible in both analytical and diagonalization approaches.

Consider the example illustrated in Figure 2. In this case, we want to synthesize the unitary labeled *Target*. Using an analytical synthesis algorithm (e.g., QSD) results in an exponential number of R_Z gates, which are then decomposed into many more T gates. Numerical-optimization methods find much more efficient circuits at the cost of compilation time but still result in far too many T gates. State-of-the-art search-based methods can produce optimal circuits for high-error approximations but have untenable run times as the target precision increases. Ultimately, this lack of precision limits their use in end-to-end compilation (Section 6). Diagonalization combines both the resource-efficiency of search algorithms with the high precision available from analytical methods to practically generate high-quality approximate circuits during the compilation of a wide range of quantum algorithms to an FT gate set.

3 Formalizing Unitary Synthesis

The unitary synthesis is typically framed as finding a circuit which inverts the adjoint (conjugate transpose) of a target matrix. To solve the problem directly by matrix inversion, every n-qubit circuit is described as a sequence of primitive gates taken from a finite set $A \subset \mathbb{U}(2^n)$. Every primitive gate has an associated unitary $a \in A$. A circuit consisting of t gates is associated with a unitary matrix:

$$C_t = a_t \times \cdots \times a_1. \tag{2}$$



(a) Synthesis-by-Diagonalization Ansatz

(b) 2 Qubit Diagonal Circuit

(c) 3 Qubit Diagonal Circuit

(d) CCY synthesized by diagonalization

Figure 3: Diagonalization ansatz and examples. The diagonal matrix D_θ captures up to 2^n-1 continuous degrees of freedom which we implement with $R_Z(\theta)$ and CNOT gates. The L and R subcircuits are made of discrete Clifford+T operations. (b-c) Show how to construct 2 and 3 qubit diagonal circuits. In many cases, not all the R_Z gates in these circuits are needed. (d) A CCY gate only requires two H gates, an S, and an $S^\dagger$ to be diagonalized. A Toffoli is realized if the S gates are removed. Its diagonalization is implemented with only CNOT and $R_Z(\pm\frac{\pi}{4})$ (T or $T^\dagger$ gates).

A circuit represented by C_t implements a target unitary $U_{tar} \in \mathbb{U}(2^n)$ when the distance condition

$$d_{HS}(C_t, U_{tar}) = \sqrt{1 - \frac{1}{4^n}|Tr(C_t U_{tar}^\dagger)|^2} \leq \varepsilon \tag{3}$$

is satisfied. Here $\varepsilon \ll 1$ is a hyperparameter which controls how precisely a target is implemented.

Synthesis methods that invert the adjoint of a target unitary using FT gate sets must find a (likely very long) sequence of discrete gates that satisfies Equation 3. We propose an alternative approach: diagonalization (Figures 2 and 3). The diagonalization process consists of a two-headed search that halts when the target's adjoint is diagonalized (i.e., not fully inverted). Diagonalizing ensures that up to $2^n - 1$ continuous R_Z operations can be handled. Although there is no guarantee of optimality, this approach is able to find high-precision implementations of complex unitaries where search-based inversion methods fail (Figure 4), and uses significantly fewer resources than pure analytical methods (Table 2). In the worst case, diagonalization acts exactly like the underlying search-based algorithm which implements it. When both diagonalization and search-based methods fail, analytical rule-based methods may be necessary.

3.1 Synthesis as a Markov Decision Process

Markov Decision Processes (MDPs) provide a framework for describing stochastic decision problems. Posing quantum circuit synthesis as an MDP has been demonstrated before [47, 28, 10, 2, 35]. We define the MDP as a tuple (S, S_I, S_T, A, r). Here S is a set of states, S_I a set of initial states, S_T a set of terminal states, A a set of actions or gates, and $r : S \times A \to \mathbb{R}$ a reward function indicating when synthesis is done. We use the term state (or sometimes unitary state) to describe an MDP state s_t, not a quantum mechanical state vector or density operator.

4 Diagonalization

Directly synthesizing a unitary by finding a sequence of gates that inverts it is intractable when high precision is needed. This is because many discrete gates are required to implement a single continuous

rotation. We alleviate the demands placed on search-based synthesis algorithms by using them to diagonalize rather than directly invert unitaries. Any search-based synthesis algorithm can be used to diagonalize unitaries. To fit the paradigm of search-based synthesis as described in Section 3, we first define the problem of diagonalization as an MDP. At time $t \in \{0,1,\ldots,T\}$ in the synthesis process, the state $s_t \in S$ is defined by

$$s_t = L_t U_{tar}^{\dagger} R_t \tag{4}$$

where L_t, R_t are each sequences of discrete operations (as in Equation 2). The set of initial states contains the adjoint of target unitaries $S_I = \{U_{tar}^{\dagger}\}$. The set of terminal states is the set of all states which can be approximately inverted by a diagonal unitary matrix. This corresponds to the set of states satisfying

$$d_D(s_T) = d_{HS}\big(s_T, D_\theta\big) \leq \varepsilon \tag{5}$$

where D_θ is a diagonal unitary, and θ is some vector of real rotation angles. It is sufficient that any $\theta \in \mathbb{R}^{2^n-1}$ exists that satisfies this inequality for s_T to be considered a terminal state. Given a terminal state s_T, the corresponding circuit is

$$C_\theta = R_T s_T^{\dagger} L_T = R_T D_\theta^{-1} L_T. \tag{6}$$

Figure 3 illustrates the general form of three-qubit circuits which satisfy this structure. As an example, the CCY gate can be diagonalized by just four Clifford gates (2 H gates, an S and an $S^{\dagger}$ gate).

Synthesis-by-diagonalization can be considered a Singular Value Decomposition, where the left and right singular vector matrices are restricted to unitaries which can be implemented exactly by a discrete gate set (e.g., Clifford+T). In some cases, diagonalization reduces to inversion (meaning $L_T = D_\theta = I$). This implies that any circuit which can be efficiently synthesized by inversion can also be synthesized by diagonalization. *Given the same search algorithm, the set of unitaries that diagonalization is able to synthesize is a strict superset of the unitaries that inversion can synthesize.*

4.1 The Diagonal Distance

Our goal is to determine when a state s_t can be nearly inverted. This means we can satisfy the distance condition in Equation 5 after multiplying by a diagonal unitary.

Theorem 1. *A unitary satisfying* $\max_{i\in[2^n]} \sqrt{\sum_{j\neq i} |u_{ij}|^2} \leq \varepsilon$*, where* u_{ij} *are the unitary's elements, is at most a Hilbert-Schmidt distance of* ε *away from the identity when multiplied by some diagonal unitary.*

Proof. Say the state of the synthesis process is s_t. Each row of s_t is of the form

$$s_t[i] = \begin{bmatrix} u_{i1} & \ldots & u_{ii} & \ldots & u_{i2^n} \end{bmatrix}.$$

Consider a diagonal unitary matrix D such that for all i, $d_{ii}u_{ii} = |u_{ii}|$. It then follows that $tr(D \times s_t) = \sum_i |u_{ii}|$. By the unitarity of s_t, we know $|u_{ii}|^2 = 1 - \sum_{j\neq i} |u_{ij}|^2$. So

$$tr(D \times s_t) = \sum_i |u_{ii}| = \sum_i \sqrt{1 - \sum_{j\neq i} |u_{ij}|^2} \leq \sqrt{4^n(1-\varepsilon^2)}$$

since we assumed that $\sum_{j\neq i} |u_{ij}|^2 \leq \varepsilon$. This satisfies the Hilbert-Schmidt distance condition (Eq 3) because

$$d_{HS}(D \times s_t, I) = \sqrt{1 - \frac{1}{4^n}|tr(D \times s_t)|^2} \leq \sqrt{1 - \frac{1}{4^n}|\sqrt{4^n(1-\varepsilon^2)}|^2} = \varepsilon$$

□

4.2 Synthesizing Diagonal Unitaries

The diagonal operator D_θ captures continuous degrees of freedom that cannot be efficiently handled by the L_t and R_t unitaries. We prepare an ansatz implementing D_θ using specialized algorithms for synthesizing diagonal unitaries [8]. Figure 3 illustrates the circuit structure of generic diagonal unitaries for the 2 and 3 qubit case using only *CNOT* and R_Z gates. We chose to implement R_Z rotations in the Clifford+T gate set using *gridsynth* [36], but alternative techniques can also be used. Some R_Z gates appearing in these ansatzes may not be used in all cases.

5 Experiments

In this section, we compare synthesis-by-diagonalization with two synthesis-by-inversion algorithms. The first is Synthetiq [31], a simulated annealing search-based synthesis algorithm. The second is the Quantum Shannon Decomposition (QSD) [37], an analytical synthesis algorithm.

5.1 Synthesis of Controlled Rotations

To illustrate the advantages of diagonalization compared to direct inversion with search-based methods, we use both to synthesize controlled rotations. These primitive gates are ubiquitous in quantum algorithms, including Shor's algorithm [39] and Hamiltonian simulation circuits [27]. Specifically, we target 100 different random angle $CR_Y(\theta)$ and $CCR_Y(\theta)$ unitaries. These circuits look similar to Figure 3d, but with $R_Z(\theta)$ gates instead of T gates.

We compare simulated annealing based diagonalization to direct inversion in Figure 4. We modified the simulated annealing synthesis tool, Synthetiq [31], so that it can perform synthesis by diagonalization. As mentioned in Section 4, diagonalization is strictly more powerful than inversion; anything which Synthetiq can synthesize by inversion, it can synthesize by diagonalization.

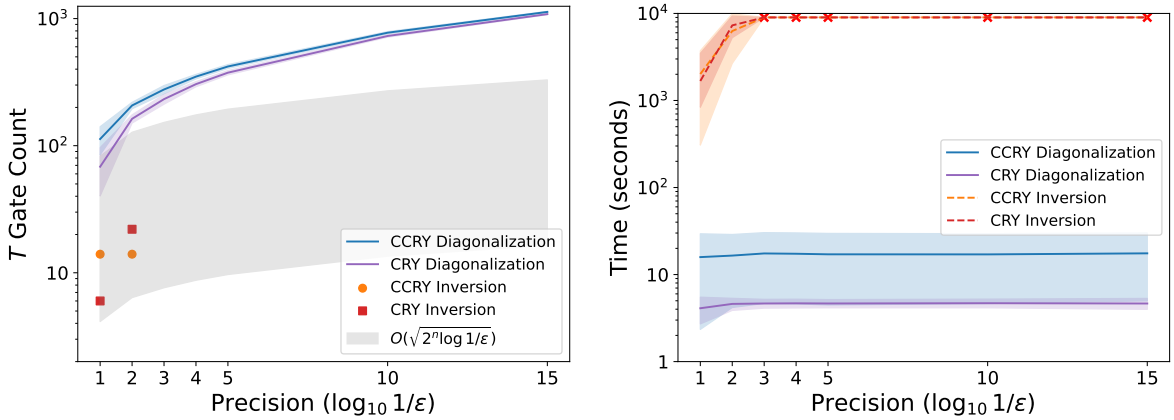


Figure 4: T gate count and run time when synthesizing CCR_Y and CR_Y unitaries with direct-inversion and diagonalization using Synthetiq [31]. At low precision, direct-inversion with search-based synthesis can produce optimal results. As precision increases beyond 10^{-2}, diagonalization finds solutions whereas inversion does not. The grey area indicates different constant value scalings of the optimal T count for diagonal unitaries [19]. Improvements in diagonal synthesis algorithms would allow diagonalization to achieve these bounds. Diagonalization takes less than 20 seconds across all precisions, inversion times out after 2.5 hours (red "x" marks indicate time outs). The success rate of inversion is 4% compared to 100% for diagonalization.

Low T count implementations of some unitaries can be found for very low precision levels when directly inverting unitaries. Given a timeout period of 2.5 hours, and running on 64 AMD EPYC 7720P CPU physical cores, the direct inversion method of synthesis is only able to find solutions in 4/100 cases when $\varepsilon = 10^{-2}$ for both the CCR_Y and CR_Y unitaries. The diagonalization approach produces implementations that contain more T gates, but its run time does not depend strongly on the target precision, and it finds solutions for all 100 angles tested. Improvements in synthesis techniques for diagonal operators (see Equation 7) promise to lower T-counts further.

This experiment highlights how pure search-based methods are largely incapable of synthesizing unitaries which have continuous degrees of freedom. Search-based synthesis algorithms find resource efficient implementations of unitary matrices at low precisions, but fail when higher precision is needed.

5.2 Synthesis of Complex Unitaries from Quantum Algorithms

Diagonalization works well for controlled rotations, but also produces high precision implementations of more complex unitaries which appear in quantum algorithms. In this regime, search-based methods are hopeless (see Figure 4). We therefore compare to the Quantum Shannon Decomposition (QSD) [37].

In addition to the simulated annealing diagonalizer mentioned in Section 5.1, we also trained Reinforcement Learning (RL) agents capable of diagonalizing two- and three-qubit unitaries. We did this because we found inference-based search to be $\approx 1000\times$ faster than simulated annealing. Search-based synthesis with RL has been demonstrated before [47, 28, 10, 2, 35]. To train the diagonalizing agent, we randomly generated separate A and B Clifford+T circuits, then inserted random D_θ operators to form targets in the form $U_{tar} = A \times D_\theta \times B$. Each A and B circuits contained up to 20 random Clifford+T gates.

Throughout these experiments, we require that unitaries be synthesized to a distance of $\varepsilon = 10^{-6}$ (see Equation 3). Higher precision can be attained in most cases (see Section 6). Individual R_Z gates are synthesized using *gridsynth*. Each rotation is synthesized to a distance of $\varepsilon = 10^{-7}$. In the three-qubit diagonalization case, this means that the total Hilbert-Schmidt distance due to R_Z approximation is at

		Heisenberg	HHL	Hubbard	QAOA	QPE	Shor	TFIM	VQE	Mean
2 Qubits	Time per Unitary (s)	0.98 0.93	0.96 0.63	0.97 0.92	0.98 0.86	0.97 0.59	0.96 0.73	0.98 0.52	0.96 0.87	0.97 0.76
	Success Rate	100% 93.7%	100% 37.4%	100% 42.8%	100% 27.1%	100% 29.1%	100% 41.7%	100% 51.0%	100% 20.0%	100% 42.9%
	R_Z Count	7.07 1.0	7.58 1.64	6.34 0.85	8.27 1.06	7.88 0.91	7.31 1.56	6.15 1.0	4.88 0.94	6.94 1.12
	T Count	0.28 (495.2) 0.03 (70.0)	0.29 (530.9) 0.36 (115.2)	0.04 (443.8) 0.0 (59.5)	0.42 (579.3) 0.0 (74.2)	0.22 (551.8) 1.15 (64.9)	0.15 (511.9) 0.1 (109.3)	0.0 (430.5) 0.0 (70.0)	0.02 (341.6) 0.16 (66.0)	0.18 (486.0) 0.23 (78.6)
	Clifford Count	4.66 4.39	4.20 4.01	4.63 3.16	5.15 2.10	4.35 3.50	5.77 4.07	2.06 5.47	3.33 8.31	4.27 4.38
	T Reduction	**85.9%**	**78.4%**	**86.6%**	**87.2%**	**88.5%**	**78.7%**	**83.7%**	**80.7%**	**83.5%**
3 Qubits	Time per Unitary (s)	1.07 1.03	1.06 1.07	1.06 1.12	1.18 1.02	1.07 1.06	1.18 1.12	1.07 1.01	1.05 0.98	1.09 1.05
	Success Rate	100% 27.4%	100% 9.6%	100% 55.6%	100% 11.2%	100% 78.5%	100% 18.0%	100% 9.4%	100% 13.3%	100% 27.9%
	R_Z Count	44.60 1.88	30.42 2.06	36.73 0.87	30.96 2.27	46.89 1.48	43.67 4.21	34.63 1.90	43.57 0.72	38.93 1.92
	T Count	1.46 (3124) 0.1 (132)	0.68 (2130) 2.15 (146)	1.85 (2573) 0.47 (61)	1.22 (2168) 0.0 (159)	1.26 (3284) 3.54 (107)	1.8 (3059) 0.28 (295)	1.1 (2425) 0.17 (133)	1.84 (3052) 0.43 (51)	1.4 (2727) 0.89 (135)
	Clifford Count	33.85 13.06	29.24 15.94	34.48 14.85	30.06 10.22	35.41 14.00	37.95 11.92	29.95 16.76	34.96 16.69	33.24 14.18
	T Reduction	**95.8%**	**93.2%**	**97.6%**	**92.7%**	**96.8%**	**90.4%**	**94.5%**	**98.3%**	**95.1%**

Table 2: Synthesis of 2- and 3-qubit unitaries taken from partitioned circuits (see Figure 5 for an illustration). Numbers indicate average time, success rate, Clifford, and non-Clifford gate counts for unitaries taken from a variety of benchmarks. To avoid bias, gate counts are only reported for trials where both methods succeed. T counts are reported so that the number of T gates introduced by the search based synthesis algorithm appear first, followed by that number plus the number of T gates due to compiling R_Z gates into Clifford+T. Compared to the QSD (top rows), diagonalization (bottom rows) on average reduces the number of R_Z gates by 83.5% (95.1%) for 2- (3-)qubit unitaries. The average reduction is computed as $(T_{QSD} - T_{Diag})/T_{QSD}$. Comparisons are made to the QSD because other synthesis tools fail to find solutions given $\varepsilon < 10^{-3}$.

most 7×10^{-7} [43]. QSD results are optimized by simplifying gate sequences and replacing R_Z gates with Clifford+T gates when possible.

We evaluated the diagonalizing agent's performance on a set of unitaries taken from partitioned quantum algorithms (see Figure 5 for an illustration). This suite of algorithms includes Shor's Algorithm [39], TFIM, Heisenberg, and Hubbard model quantum chemistry simulation circuits [4], trained VQE [32] and QAOA [14] circuits, and QPE circuits [22]. The VQE and QAOA circuits were generated by MQTBench [34]. Each set of unitaries was filtered to ensure that every unitary was unique. There were 22,323 different two-qubit unitaries and 45,202 different three-qubit unitaries. Table 2 summarizes.

We find that in the two-qubit case, diagonalization can find circuits implementing 42.9% of unitaries across all benchmarks. About 79% of these unitaries are already diagonal, and therefore trivial for a diagonalizing agent to synthesize. Most other unitaries contain 2-8 gates, typically H and S gates. For the three-qubit case, realistic unitaries are far more complex: they are more likely to contain unitaries which do not conform to the diagonalization ansatz (see Figure 4a). Approximately 27.9% of three-qubit unitaries from our suite of partitioned circuits could be synthesized by the diagonalizing agent. Of these, approximately 57% were already diagonal. On average, the diagonal operators contained ≈ 4 R_Z gates. The $L_t(\cdot)R_t$ circuits contained an average of about 11 Clifford+T gates.

Compared to the QSD, diagonalization produces solutions with far fewer non-trivial rotation gates. Implementing an R_Z gate requires $O(\log \frac{1}{\varepsilon})$ T gates when synthesized optimally with *gridsynth* [36]. For $\varepsilon = 10^{-7}$ this is approximately 70 T gates per R_Z. Lone T gates are almost negligible compared to the contributions from continuous rotations in the high precision regime. Compared to the QSD, the diagonalizing synthesizer reduces the average number of non-Clifford gates by 83.5% for two qubit unitaries and 95.1% for three qubit unitaries. As the number of qubits n increases, we expect that diagonalization will outperform the QSD because the former uses at most $2^n - 1$ rotations, while the latter typically uses $O(4^n)$. The advantage of diagonalization compared to the QSD grows as $O(2^n)$.

Although diagonalization does not always succeed, the potential savings in non-Clifford gates and the speed with the process runs remain strong arguments for its utility. Unitaries which are (nearly) diagonal are ubiquitous primitives in realistic quantum benchmarks. These common unitaries are well suited for synthesis-by-diagonalization, but entirely out of reach for inversion-based synthesis algorithms.

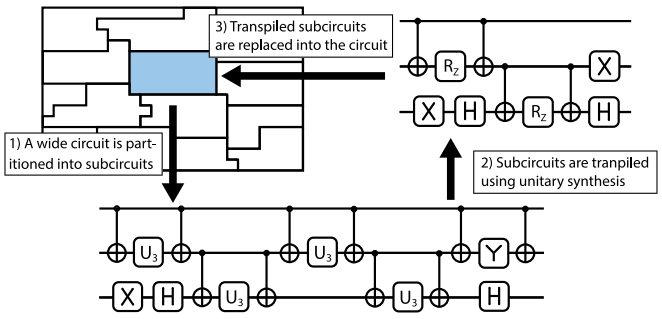


Figure 5: Fault-Tolerant gate set transpilation using unitary synthesis. Quantum algorithms are partitioned into many subcircuits. These subcircuits are transpiled independently using unitary synthesis. The optimized and transpiled subcircuits are then replaced into the original circuit. R_Z gates are handled in a post-processing stage using *gridsynth*.

5.3 End-to-End Circuit Compilation Workflow

Unitary synthesis algorithms are well suited for transpiling quantum circuits into new sets of gates. In this setting, we assume access to a quantum circuit, not just a unitary matrix. Past work targeting NISQ gate sets has demonstrated how multi-qubit synthesis can transpile circuits using fewer gates than when using gate-by-gate replacement rules [46]. This added optimization power comes from replacing gate-level local translations with more globally aware discovered replacements. Here we consider whether the ability to handle more complex unitaries with diagonalization yields similar results in FT transpilation.

We consider a *gate-level* transpilation strategy as a control; circuits are transpiled gate-by-gate (as opposed to subcircuit-by-subcircuit) into the Clifford+T gate set. We compare to this strategy because we already have a gate-level implementation of the algorithm, and the only other method that produces high precision implementations of these unitaries, the QSD, results in an explosion of T gates (Table 2).

A summary of the process used is shown in Figure 5. Our method is:

1. Partition a quantum circuit into 2- or 3-qubit blocks that contain as many gates as possible.
2. For each 2- or 3-qubit unitary, use diagonalization to synthesize a high-precision approximation. At the same time, use replacement rules and *gridsynth* to transpile each partition into Clifford+T gates.
3. If diagonalization fails or produces results with more T gates, use the gate-level transpilation results.

This process ensures that using multi-qubit synthesis for transpilation never performs worse than the gate-level transpilation strategy.

Table 3 summarizes across several quantum algorithms and primitives. The total approximation error across the entire circuit is about $\varepsilon_{\text{total}} \approx 10^{-6}$. This value is an upper bound that is found by summing the individual approximation errors for each partition and each *gridsynth* transpiled R_Z gate [43]. Transpilation via diagonalization results in T gate savings for these algorithms compared to gate-level transpilation. We see the highest T gate count reduction (18.1%) for the 16 qubit multiplier circuit.

		Add	HHL	Mult	QAE	QFT	QPE	Shor
	Qubits	17	6	16	50	32	30	16
By Gate	R_Z Gates	251	243	1,079	715	1,080	321	816
	T Gates	23,562	22,571	100,678	65,779	101,480	29,296	74,797
	$\varepsilon_{\text{total}}$	2.5×10^{-7}	2.4×10^{-7}	1.1×10^{-6}	6.7×10^{-7}	1.1×10^{-6}	3.2×10^{-7}	8.2×10^{-7}
2Q Blocks	R_Z Gates	240	241	1,079	683	1,080	310	816
	T Gates	21,924	22,201	98,516	62,405	98,640	28,290	74,790
	$\varepsilon_{\text{total}}$	4.5×10^{-7}	2.9×10^{-7}	1.2×10^{-6}	4.8×10^{-6}	4.4×10^{-6}	1.1×10^{-6}	8.2×10^{-7}
	% Diagonalized	94.0%	72.4%	92.7%	45.1%	91.0%	21.4%	26.2%
	Improvement	7.0%	1.6%	2.1%	5.1%	2.8%	3.4%	0.0%
3Q Blocks	R_Z Gates	216	240	898	686	949	316	816
	T Gates	19,912	22,265	82,500	62,667	87,248	28,836	74,720
	$\varepsilon_{\text{total}}$	4.2×10^{-7}	2.8×10^{-7}	1.2×10^{-6}	4.2×10^{-6}	3.1×10^{-6}	9.7×10^{-7}	8.2×10^{-7}
	% Diagonalized	85.1%	14.3%	73.5%	40.6%	85.3%	14.4%	12.5%
	Improvement	15.5%	1.4%	18.1%	4.7%	14.0%	1.6%	0.1%

Table 3: Transpilation results. The *By Gate* strategy indicates gate-level transpilation into Clifford+T gates. The *2Q* and *3Q Block* strategies indicate circuits partitioned into subcircuits of that size and synthesized by diagonalization. In scenarios where diagonalization fails, subcircuits are transpiled gate-by-gate. Each subcircuit is synthesized to a distance of $\varepsilon = 10^{-8}$. Individual R_Z gates are synthesized to $\varepsilon = 10^{-9}$, resulting in at least 90 T gates per R_Z. The total approximation error in each circuit on the order of $\varepsilon_{\text{total}} \approx 10^{-6}$. Reported *Improvement* is percent decrease in T gate counts.

Other algorithmic primitives such as the 17 qubit adder circuit, and the 32 qubit approximate QFT see similarly large reductions. The HHL, QAE, and QPE circuits see more moderate decreases in T count.

How well diagonalization transpiles circuits is highly dependent on the circuit partitioning algorithm. For example, the 16-qubit transpiled implementation of Shor's algorithm sees little improvement compared to the gate-level strategy. Using two-qubit blocks results in a success rate of 26% for this circuit. In the three-qubit case only 12% of partitioned subcircuits are successfully diagonalized. Shor's algorithm consists of many repeated copies QFT and inverse QFT modules, which our data indicate is a class of circuit that can be simplified greatly by diagonalization. For two- and three-qubit partitions, 91% and 85% of subcircuits taken from the example 32 qubit QFT circuit are successfully transpiled by diagonalizing. The performance of the transpilation strategy is therefore highly likely to be dependent on the partitioning algorithm used. Partitioning strategies which are informed by circuit structure are likely to improve results.

Combined with circuit partitioning, unitary synthesis enables transpilation to take place on a less localized scale. This more global view of a circuit's function enables synthesis to outperform simple gate-level methods when paired with approximation in the FT setting.

6 Discussion

FT synthesis algorithms must be able to approximate unitaries to high levels of precision. In the worst case, the total approximation error in a transpiled circuit is the sum of each individual gate's and partition's approximation error [43]. If the circuits shown in Section 5.3 (see Table 3) had been transpiled by a synthesis algorithm capable of only finding solutions with precision $\varepsilon = 10^{-3}$, the average total approximation error of the transpiled circuits would be upper bounded by $\varepsilon_{\text{total}} = 0.56$ (ranging from 0.05 to the max error of 1.0). This much approximation error is unlikely to lead to meaningful algorithm outputs. The outlook for coarse synthesis is even worse for wider circuits which contain more gates.

Because diagonalization allows for $2^n - 1$ continuous rotations to be handled analytically, much higher levels of precision are possible with this approach. In fact, we can consider the class of *exactly diagonalizable* unitaries (analogous to exactly synthesizable unitaries), where $U = RD_\theta^{-1}L$ and each entry of $L, R \in \mathbb{Z}[e^{i\pi/4}, \frac{1}{2}]$. These exactly diagonalizable unitaries can be approximated to arbitrarily high levels of precision using synthesis by diagonalization, so long as L and R can be found. Common examples of these unitaries include controlled rotation gates (Section 5.1).

Most of the unitaries diagonalization finds solutions for fit into the category of exact diagonalizability. For this reason, values more precise than $\varepsilon = 10^{-6}$-10^{-8} (which are shown in this paper) can be attained. These values are orders of magnitude (10^3-$10^5\times$) more precise than previous search-based methods (Table 1) and are enough to enable high precision transpilation of complete algorithms (Table 3). Determining the exact precision needed to ensure circuits produce meaningful results is an open area of research. How techniques such as unitary mixing [9] can be used in this setting to boost precision with ensembles of coarse R_Z implementations is worth exploring. Unitary mixing specifically enables quadratic improvements in precision, meaning this technique can be used to boost precision from 10^{-6} to 10^{-12}.

Our synthesis approach is composable and extensible. As most FT synthesis work has focused on the Clifford+T gate set, there may be unexplored optimization opportunity when considering alternate gate sets such as Clifford+$\sqrt{T}$ and Clifford+V. Search-based synthesis tools are well suited to begin answering these questions, as synthesis in alternative gate sets is simply a matter of modifying what gates can be applied.

Recent work [19] has proved that a diagonal unitary D_θ can be approximated to a diamond distance threshold of $\varepsilon_\diamond$, where the T count scales as

$$\text{T-count}(D_\theta) = \Theta(\sqrt{2^n \log \frac{1}{\varepsilon_\diamond}} + \log \frac{1}{\varepsilon_\diamond}), \tag{7}$$

but an efficient algorithm achieving this bound has not been found. Our approach uses $O(2^n \log \frac{1}{\varepsilon})$ T gates to synthesize diagonal unitaries. Improvements in techniques for synthesizing diagonal unitaries can therefore greatly improve the T-count of our synthesis-by-diagonalization approach. This improvement in resource efficiency is illustrated by the grey region shown in Figure 4.

This work showcases the benefit of augmenting analytical decomposition methods with search-based multi-qubit methods. We expect that other analytical-search-based hybrid approaches that offer further improvements are both possible and practical.

6.1 The Utility of Diagonalization for Compilation Tasks

By augmenting search-based compilation with generalized analytical decomposition, diagonalization greatly expands the domain of unitaries which can be transpiled and increases the precision to which they are transpiled. However, diagonalization also enables even more powerful compilation strategies.

Ancilla qubits and projective measurements exposes more opportunity for optimization when implementing operations in FT gate sets. Repeat Until Success schemes [7, 6] leverage these resources to reduce the number of non-Clifford gates needed to implement R_Z rotations compared to optimal ancilla-free synthesis. These techniques rely on synthesizing unitaries with a particular structure (the *Jack of Daggers* structure [7]). We believe diagonalization and other powerful synthesis techniques will help discover more structures which systematically reduce non-Clifford gate counts when using ancilla.

By loosening the synthesis objective from inversion to diagonalization, our tool exposes opportunity for resource-efficient implementations of continuous rotations. As shown in [19] and mentioned in Section 7, further improvements can be made in the diagonal synthesis procedure. Improvements here would automatically benefit synthesis-by-diagonalization.

7 Conclusion

We have demonstrated a novel approach to high precision multi-qubit unitary synthesis targeting fault-tolerant gate sets. By diagonalizing unitaries instead of directly inverting them, complex continuous degrees of freedom can be bypassed then handled efficiently with mathematical decomposition methods. This enables us to synthesize a broader scope of unitaries than other synthesis methods which directly invert continuous rotations, a task which is intractable for high levels of precision. The diagonalization process is general and other synthesis tools can be retrofitted to diagonalize rather than invert unitaries.

The effectiveness of our diagonalizing approach is demonstrated by synthesizing unitaries taken from a suite of partitioned quantum algorithms to very high precision. In this regime, only resource inefficient analytical methods are also able to find solutions. Compared to the Quantum Shannon Decomposition, our approach reduces the number of expensive T gates by 83.5% in two-qubit unitaries and 95.1% in three-qubit unitaries. Our approach can be used to transpile future-term algorithms to fault-tolerant gate sets with very low approximation error. Using diagonalization results in up to a 18.1% reduction in non-Clifford gates compared to gate-by-gate transpilation. The diagonalizing approach is fast and able to find low gate count implementations of meaningful unitaries, making it a promising technique for use in future fault-tolerant quantum compilers.

References

[1] Rajeev Acharya, Dmitry A. Abanin, Laleh Aghababaie-Beni, Igor Aleiner, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, Ryan Babbush, Dave Bacon, Brian Ballard, Joseph C. Bardin, Johannes Bausch, Andreas Bengtsson, Alexander Bilmes, Sam Blackwell, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, David A. Browne, Brett Buchea, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Anthony Cabrera, Juan Campero, Hung-Shen Chang, Yu Chen, Zijun Chen, Ben Chiaro, Desmond Chik, Charina Chou, Jahan Claes, Agnetta Y. Cleland, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Sayan Das, Alex Davies, Laura De Lorenzo, Dripto M. Debroy, Sean Demura, Michel Devoret, Agustin Di Paolo, Paul Donohoe, Ilya Drozdov, Andrew Dunsworth, Clint Earle, Thomas Edlich, Alec Eickbusch, Aviv Moshe Elbag, Mahmoud Elzouka, Catherine Erickson, Lara Faoro, Edward Farhi, Vinicius S. Ferreira, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, Suhas Ganjam, Gonzalo Garcia, Robert Gasca, Élie Genois, William Giang, Craig Gidney, Dar Gilboa, Raja Gosula, Alejandro Grajales Dau, Dietrich Graumann, Alex Greene, Jonathan A. Gross, Steve Habegger, John Hall, Michael C. Hamilton, Monica Hansen, Matthew P. Harrigan, Sean D. Harrington, Francisco J. H. Heras, Stephen Heslin, Paula Heu, Oscar Higgott, Gordon Hill, Jeremy Hilton, George Holland, Sabrina Hong, Hsin-Yuan Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Stephen Jordan, Chaitali Joshi, Pavol Juhas, Dvir Kafri, Hui Kang, Amir H. Karamlou, Kostyantyn Kechedzhi, Julian Kelly, Trupti Khaire, Tanuj Khattar, Mostafa Khezri, Seon Kim, Paul V. Klimov, Andrey R. Klots, Bryce Kobrin, Pushmeet Kohli, Alexander N. Korotkov, Fedor Kostritsa, Robin Kothari, Borislav Kozlovskii, John Mark Kreikebaum, Vladislav D. Kurilovich, Nathan Lacroix, David Landhuis, Tiano Lange-Dei, Brandon W. Langley, Pavel Laptev, Kim-Ming Lau, Loïck Le Guevel, Justin Ledford, Joonho Lee, Kenny Lee, Yuri D. Lensky, Shannon Leon, Brian J. Lester, Wing Yan Li, Yin Li, Alexander T. Lill, Wayne Liu, William P. Livingston, Aditya Locharla, Erik Lucero, Daniel Lundahl, Aaron Lunt, Sid Madhuk, Fionn D. Malone, Ashley Maloney, Salvatore Mandrà, James Manyika, Leigh S. Martin, Orion Martin, Steven Martin, Cameron Maxfield, Jarrod R. McClean, Matt McEwen, Seneca Meeks, Anthony Megrant, Xiao Mi, Kevin C. Miao, Amanda Mieszala, Reza Molavi, Sebastian Molina, Shirin Montazeri, Alexis Morvan, Ramis Movassagh, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Chia-Hung Ni, Murphy Yuezhen Niu, Thomas E. O'Brien, William D. Oliver, Alex Opremcak, Kristoffer Ottosson, Andre Petukhov, Alex Pizzuto, John Platt, Rebecca Potter, Orion Pritchard, Leonid P. Pryadko, Chris Quintana, Ganesh Ramachandran, Matthew J. Reagor, John Redding, David M. Rhodes, Gabrielle Roberts, Eliott Rosenberg, Emma Rosenfeld, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Andrew W. Senior, Michael J. Shearn, Aaron Shorter, Noah Shutty, Vladimir Shvarts, Shraddha Singh, Volodymyr Sivak, Jindra Skruzny, Spencer Small, Vadim Smelyanskiy, W. Clarke Smith, Rolando D. Somma, Sofia Springer, George Sterling, Doug Strain, Jordan Suchard, Aaron Szasz, Alex Sztein, Douglas Thor, Alfredo Torres, M. Mert Torunbalci, Abeer Vaishnav, Justin Vargas, Sergey Vdovichev, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Steven Waltman, Shannon X. Wang, Brayden Ware, Kate Weber, Travis Weidel, Theodore White, Kristi Wong, Bryan W. K. Woo, Cheng Xing, Z. Jamie Yao, Ping Yeh, Bicheng Ying, Juhwan Yoo, Noureldin Yosri, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, Nicholas Zobrist & Google Quantum AI and Collaborators (2024): *Quantum error correction below the surface code threshold*. Nature, doi:10.1038/s41586-024-08449-y.

[2] M. Sohaib Alam, Noah F. Berthusen & Peter P. Orth (2023): *Quantum logic gate synthesis as a Markov decision process*. npj Quantum Information 9(1), doi:10.1038/s41534-023-00766-w. Available at `https://www.osti.gov/biblio/2287671`.

[3] Matthew Amy, Dmitri Maslov, Michele Mosca & Martin Roetteler (2013): *A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 32(6), pp. 818–830, doi:10.1109/TCAD.2013.2244643. Available at `http://arxiv.org/abs/1206.0758`. ArXiv:1206.0758 [quant-ph].

[4] Lindsay Bassman, Connor Powers & Wibe A. De Jong (2022): *ArQTiC: A Full-Stack Software Package for Simulating Materials on Quantum Computers*. *ACM Transactions on Quantum Computing* 3(3), doi:10.1145/3511715.

[5] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2023): *Logical quantum processor based on reconfigurable atom arrays*. *Nature* 626(7997), p. 58–65, doi:10.1038/s41586-023-06927-3.

[6] Alex Bocharov, Martin Roetteler & Krysta M. Svore (2015): *Efficient synthesis of probabilistic quantum circuits with fallback*. *Physical Review A* 91(5), p. 052317, doi:10.1103/PhysRevA.91.052317. Available at `http://arxiv.org/abs/1409.3552`. ArXiv:1409.3552 [quant-ph].

[7] Alex Bocharov, Martin Roetteler & Krysta M. Svore (2015): *Efficient synthesis of universal Repeat-Until-Success circuits*. *Physical Review Letters* 114(8), p. 080502, doi:10.1103/PhysRevLett.114.080502. Available at `http://arxiv.org/abs/1404.5320`. ArXiv:1404.5320 [quant-ph].

[8] Stephen S. Bullock & Igor L. Markov (2004): *Asymptotically optimal circuits for arbitrary n-qubit diagonal comutations*. *Quantum Info. Comput.* 4(1), p. 27–47.

[9] Earl Campbell (2017): *Shorter gate sequences for quantum computing by mixing unitaries*. *Physical Review A* 95(4), doi:10.1103/physreva.95.042306.

[10] Qiuhao Chen, Yuxuan Du, Yuliang Jiao, Xiliang Lu, Xingyao Wu & Qi Zhao (2024): *Efficient and practical quantum compiler towards multi-qubit systems with deep reinforcement learning*. *Quantum Science and Technology* 9(4), p. 045002, doi:10.1088/2058-9565/ad420a. Available at `https://dx.doi.org/10.1088/2058-9565/ad420a`.

[11] Marc G. Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi & Costin Iancu (2020): *Towards Optimal Topology Aware Quantum Circuit Synthesis*. In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 223–234, doi:10.1109/QCE49297.2020.00036.

[12] A. De Vos & S. De Baerdemacker (2016): *Block-ZXZ synthesis of an arbitrary quantum circuit*. *Phys. Rev. A* 94, p. 052317, doi:10.1103/PhysRevA.94.052317. Available at `https://link.aps.org/doi/10.1103/PhysRevA.94.052317`.

[13] Bryan Eastin & Emanuel Knill (2009): *Restrictions on Transversal Encoded Quantum Gate Sets*. *Physical Review Letters* 102(11), doi:10.1103/physrevlett.102.110502.

[14] Edward Farhi, Jeffrey Goldstone & Sam Gutmann (2014): *A Quantum Approximate Optimization Algorithm*. arXiv:1411.4028.

[15] Austin G. Fowler, Matteo Mariantoni, John M. Martinis & Andrew N. Cleland (2012): *Surface codes: Towards practical large-scale quantum computation*. *Physical Review A* 86(3), doi:10.1103/physreva.86.032324.

[16] Vlad Gheorghiu, Michele Mosca & Priyanka Mukhopadhyay (2022): *T-count and T-depth of any multi-qubit unitary*. *npj Quantum Information* 8(1), pp. 1–10, doi:10.1038/s41534-022-00651-y. Available at `https://www.nature.com/articles/s41534-022-00651-y`.

[17] Brett Giles & Peter Selinger (2013): *Exact synthesis of multiqubit Clifford+T circuits*. *Phys. Rev. A* 87, p. 032332, doi:10.1103/PhysRevA.87.032332. Available at `https://link.aps.org/doi/10.1103/PhysRevA.87.032332`.

[18] David Gosset, Vadym Kliuchnikov, Michele Mosca & Vincent Russo (2013): *An algorithm for the T-count*, doi:10.48550/arXiv.1308.4134. Available at `http://arxiv.org/abs/1308.4134`. ArXiv:1308.4134 [quant-ph].

[19] David Gosset, Robin Kothari & Kewen Wu (2024): *Quantum state preparation with optimal T-count*. arXiv:2411.04790.

[20] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: a scalable quantum programming language*. *ACM SIGPLAN Notices* 48(6), p. 333–342, doi:10.1145/2499370.2462177.

[21] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. *Electronic Proceedings in Theoretical Computer Science* 318, p. 229–241, doi:10.4204/eptcs.318.14.

[22] A. Yu. Kitaev (1995): *Quantum measurements and the Abelian Stabilizer Problem*. arXiv:quant-ph/9511026.

[23] A Yu Kitaev (1997): *Quantum computations: algorithms and error correction*. *Russian Mathematical Surveys* 52(6), p. 1191, doi:10.1070/RM1997v052n06ABEH002155.

[24] Vadym Kliuchnikov, Dmitri Maslov & Michele Mosca (2013): *Fast and efficient exact synthesis of single-qubit unitaries generated by clifford and T gates*. *Quantum Info. Comput.* 13(7–8), p. 607–630.

[25] E. Knill (2004): *Fault-Tolerant Postselected Quantum Computation: Schemes*. arXiv:quant-ph/0402171.

[26] Andrew J. Landahl & Chris Cesare (2013): *Complex instruction set computing architecture for performing accurate quantum Z rotations with less magic*, doi:10.48550/arXiv.1302.3240. Available at `http://arxiv.org/abs/1302.3240`. ArXiv:1302.3240 [quant-ph].

[27] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding & Yuan Xie (2022): *Paulihedral: a generalized block-wise compiler optimization framework for Quantum simulation kernels*. In: *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, Association for Computing Machinery, New York, NY, USA, p. 554–569, doi:10.1145/3503222.3507715.

[28] Lorenzo Moro, Matteo G. A. Paris, Marcello Restelli & Enrico Prati (2021): *Quantum compiling by deep reinforcement learning*. *Communications Physics* 4(1), p. 178, doi:10.1038/s42005-021-00684-3.

[29] Michael A. Nielsen & Isaac L. Chuang (2011): *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

[30] Joe O'Gorman & Earl T. Campbell (2017): *Quantum computation with realistic magic-state factories*. *Physical Review A* 95(3), doi:10.1103/physreva.95.032338.

[31] Anouk Paradis, Jasper Dekoninck, Benjamin Bichsel & Martin Vechev (2024): *Synthetiq: Fast and Versatile Quantum Circuit Synthesis*. *Proc. ACM Program. Lang.* 8(OOPSLA1), doi:10.1145/3649813.

[32] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik & Jeremy L. O'Brien (2014): *A variational eigenvalue solver on a photonic quantum processor*. *Nature Communications* 5(1), doi:10.1038/ncomms5213.

[33] John Preskill (2018): *Quantum Computing in the NISQ era and beyond*. *Quantum* 2, p. 79, doi:10.22331/q-2018-08-06-79.

[34] Nils Quetschlich, Lukas Burgholzer & Robert Wille (2023): *MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing*. *Quantum*, doi:10.22331/q-2023-07-20-1062. arXiv:2204.13719. MQT Bench is available at `https://www.cda.cit.tum.de/mqtbench/`.

[35] Sebastian Rietsch, Abhishek Y. Dubey, Christian Ufrecht, Maniraman Periyasamy, Axel Plinge, Christopher Mutschler & Daniel D. Scherer (2024): *Unitary Synthesis of Clifford+T Circuits with Reinforcement Learning*. In: *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 01, pp. 824–835, doi:10.1109/QCE60285.2024.00102.

[36] Neil J. Ross & Peter Selinger (2016): *Optimal ancilla-free Clifford+T approximation of z-rotations*. *Quantum Info. Comput.* 16(11–12), p. 901–953.

[37] Vivek V Shende, Stephen S Bullock & Igor L Markov (2006): *Synthesis of quantum-logic circuits*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(6), pp. 1000–1010, doi:10.1109/TCAD.2005.855930.

[38] Peter W. Shor (1995): *Scheme for reducing decoherence in quantum computer memory*. *Phys. Rev. A* 52, pp. R2493–R2496, doi:10.1103/PhysRevA.52.R2493. Available at `https://link.aps.org/doi/10.1103/PhysRevA.52.R2493`.

[39] Peter W. Shor (1997): *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. *SIAM Journal on Computing* 26(5), p. 1484–1509, doi:10.1137/s0097539795293172.

[40] M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, A. Chernoguzov, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, T. M. Gatterman, L. Grans-Samuelsson, D. Hayes, N. Hewitt, J. Johansen, D. Lucchetti, M. Mills, S. A. Moses, B. Neyenhuis, A. Paz, J. Pino, P. Siegfried, J. Strabley, A. Sundaram, D. Tom, S. J. Wernli, M. Zanner, R. P. Stutz & K. M. Svore (2024): *Demonstration of logical qubits and repeated error correction with better-than-physical error rates*. arXiv:2404.02280.

[41] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *t—ket⟩: a retargetable compiler for NISQ devices*. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92.

[42] Ethan Smith, Marc Grau Davis, Jeffrey Larson, Ed Younis, Lindsay Bassman Oftelie, Wim Lavrijsen & Costin Iancu (2023): *LEAP: Scaling Numerical Optimization Based Synthesis Using an Incremental Approach*. *ACM Transactions on Quantum Computing* 4(1), doi:10.1145/3548693.

[43] Bo-Ying Wang & Fuzhen Zhang (1994): *A Trace Inequality for Unitary Matrices*. *The American Mathematical Monthly* 101(5), pp. 453–455, doi:10.1080/00029890.1994.11996973. Available at `http://www.jstor.org/stable/2974909`.

[44] John van de Wetering & Matt Amy (2024): *Optimising quantum circuits is generally hard*. arXiv:2310.05958.

[45] Xin-Chuan Wu, Marc Grau Davis, Frederic T. Chong & Costin Iancu (2021): *Reoptimization of Quantum Circuits via Hierarchical Synthesis*. In: *2021 International Conference on Rebooting Computing (ICRC)*, pp. 35–46, doi:10.1109/ICRC53822.2021.00016.

[46] Ed Younis & Costin Iancu (2022): *Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation*. In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 465–475, doi:10.1109/QCE53715.2022.00068.

[47] Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang & Dong-Ling Deng (2020): *Topological Quantum Compiling with Reinforcement Learning*. *Physical Review Letters* 125(17), doi:10.1103/physrevlett.125.170501.

Contributions to the Theory of Clifford-Cyclotomic Circuits

Linh Dinh & Neil J. Ross
Dalhousie University

Let n be a positive integer divisible by 8. The Clifford-cyclotomic gate set $\mathscr{G}_n$ consists of the Clifford gates, together with a z-rotation of order n. It is easy to show that, if a circuit over $\mathscr{G}_n$ represents a unitary matrix U, then the entries of U must lie in $\mathscr{R}_n$, the smallest subring of $\mathbb{C}$ containing $1/2$ and $\exp(2\pi i/n)$. The converse implication, that every unitary U with entries in $\mathscr{R}_n$ can be represented by a circuit over $\mathscr{G}_n$, is harder to show, but it was recently proved to be true when $n = 2^k$. In that case, $k-2$ ancillas suffice to synthesize a circuit for U, which is known to be minimal for $k = 3$, but not for larger values of k. In the present paper, we make two contributions to the theory of Clifford-cyclotomic circuits. Firstly, we improve the existing synthesis algorithm by showing that, when $n = 2^k$ and $k \geq 4$, only $k-3$ ancillas are needed to synthesize a circuit for U, which is minimal for $k = 4$. Secondly, we extend the existing synthesis algorithm to the case of $n = 3 \cdot 2^k$ with $k \geq 3$.

1 Introduction

1.1 Background

Let n be a positive integer divisible by 8. The **Clifford-cyclotomic gate set of degree** n, which we denote by $\mathscr{G}_n$, consists of the usual **Clifford gates**

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \qquad \text{and} \qquad CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

together with the z-rotation of order n

$$T_n = \begin{bmatrix} 1 & 0 \\ 0 & \zeta_n \end{bmatrix},$$

where ζ_n is the **primitive n-th root of unity** $\zeta_n = e^{2\pi i/n}$. For any n divisible by 8, $\mathscr{G}_n$ is universal for quantum computation. The gate sets $\mathscr{G}_8$ and $\mathscr{G}_{16}$ are also known as the **Clifford+T** and **Clifford+$\sqrt{T}$** gate sets, respectively. Clifford-cyclotomic circuits are important to the design of quantum algorithms [15], the theory of fault-tolerant quantum computation [5, 10], and the study of quantum complexity [14]. Because of this, they have received significant attention in the literature [2, 3, 7, 8, 9, 11, 13].

An important question in the theory of Clifford-cyclotomic circuits is that of precisely characterizing the matrices that can be exactly represented by a circuit over $\mathscr{G}_n$. Let $\mathscr{R}_n$ be the smallest subring of $\mathbb{C}$ containing $1/2$ and ζ_n. It is easy to see that, if a circuit over $\mathscr{G}_n$ represents a unitary matrix U, then the entries of U must lie in $\mathscr{R}_n$. The converse implication, that every unitary U with entries in $\mathscr{R}_n$ can be represented by a circuit over $\mathscr{G}_n$, is harder to show. Indeed, until recently, it was only known to be true for the Clifford+T gate set $\mathscr{G}_8$, as well as for a handful of adjacent gate sets [3, 8]. Recently, however, it was shown that this converse is true whenever n is a power of 2 [2]. In that case, $\log(n) - 2$ ancillas suffice to synthesize a circuit for U, which is known to be minimal for $n = 8$, but not for larger powers of 2. This recent exact synthesis result naturally raises two questions. Firstly, when n is a power of 2, can the

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 231–246, doi:10.4204/EPTCS.426.9

number of ancillas needed to synthesize a Clifford-cyclotomic circuit of degree n be made smaller than $\log(n)-2$? Secondly, can one prove an exact synthesis result for Clifford-cyclotomic circuits whose degree is not a power of 2? In the present paper, we contribute to the theory of Clifford-cyclotomic circuits by answering both questions positively.

1.2 Contributions

Let m be a positive integer.

We prove that any 2^m-dimensional unitary U with entries in $\mathscr{R}_{16}$ can be exactly represented by an m-qubit circuit over $\mathscr{G}_{16}$ using at most 1 ancilla, which is minimal. We establish this result through a study of the effect of catalytic embeddings [1] on the determinant which may be of independent interest. We then use this result about circuits over $\mathscr{G}_{16}$ to save an ancilla in synthesizing circuits over $\mathscr{G}_{2^k}$, for $k \geq 4$.

We also prove, inspired by the results of [4], that any 2^m-dimensional unitary U with entries in $\mathscr{R}_{3\cdot 2^k}$ with $k \geq 3$ can be exactly represented by an m-qubit circuit over $\mathscr{G}_{3\cdot 2^k}$, thereby establishing a number-theoretic characterization for Clifford-cyclotomic circuits whose degree is not a power of 2.

2 Rings

We now introduce the rings that will be important in what follows and we discuss some of their properties. For further details, we encourage the reader to consult [6, 17].

We assume that rings have a multiplicative identity. If R is a ring and $u \in R$, we write $R/(u)$ for the **quotient of R by the ideal** (u). Two elements v and v' of the ring R are **congruent modulo** u, if their difference is a multiple of u, that is, if $v - v' \in (u)$. In that case, we write $v \equiv_u v'$, or $v \equiv v' \pmod{u}$. The relation $\equiv_u$ is an equivalence relation on R and the elements of $R/(u)$ are precisely the equivalence classes of elements of R under the relation $\equiv_u$. We sometimes refer to these equivalence classes as **residues**.

2.1 Cyclotomic integers

We write ζ_n for the **primitive n-th root of unity** $\zeta_n = e^{2\pi i/n}$. The **ring of cyclotomic integers** $\mathbb{Z}[\zeta_n]$ is the smallest subring of $\mathbb{C}$ that contains ζ_n. Since $\zeta_n^\dagger = \zeta_n^{n-1}$, the ring $\mathbb{Z}[\zeta_n]$ is closed under complex conjugation.

Let φ denote **Euler's totient function**, so that $\varphi(n)$ counts the integers in $\{1,\ldots,n\}$ that are relatively prime to n. The ring $\mathbb{Z}[\zeta_n]$ can be characterized as

$$\mathbb{Z}[\zeta_n] = \left\{ a_0 + a_1\zeta_n + a_2\zeta_n^2 + \cdots + a_{\varphi(n)-1}\zeta_n^{\varphi(n)-1} \mid a_0,\ldots,a_{\varphi(n)-1} \in \mathbb{Z} \right\}. \tag{1}$$

Every element $u \in \mathbb{Z}[\zeta_n]$ can be uniquely expressed as a $\mathbb{Z}$-linear combination of powers of ζ_n as in Equation (1). That is, we have

$$a_0 + a_1\zeta_n + \cdots + a_{\varphi(n)-1}\zeta_n^{\varphi(n)-1} = a'_0 + a'_1\zeta_n + \cdots + a'_{\varphi(n)-1}\zeta_n^{\varphi(n)-1}$$

if and only if $a_j = a'_j$ for $0 \leq j \leq \varphi(n)-1$.

We will be interested in two families of rings of cyclotomic integers: the one corresponding to $n = 2^k$, and the one corresponding to $n = 3\cdot 2^k$. For $k \leq k'$, we have $\mathbb{Z}[\zeta_{2^k}] \subseteq \mathbb{Z}[\zeta_{2^{k'}}]$ (resp. $\mathbb{Z}[\zeta_{3\cdot 2^k}] \subseteq \mathbb{Z}[\zeta_{3\cdot 2^{k'}}]$).

Moreover, for $k \geq 1$, every element $u \in \mathbb{Z}[\zeta_{2^{k+1}}]$ (resp. $u \in \mathbb{Z}[\zeta_{3\cdot 2^{k+1}}]$) can be uniquely written as $u = a + b\zeta_{2^{k+1}}$ (resp. $u = a + b\zeta_{3\cdot 2^{k+1}}$), with $a, b \in \mathbb{Z}[\zeta_{2^k}]$ (resp. $a, b \in \mathbb{Z}[\zeta_{3\cdot 2^k}]$).

We define the ring $\mathscr{R}_n$ as the smallest subring of $\mathbb{C}$ that contains $1/2$ and ζ_n. The ring $\mathscr{R}_n$ can be characterized as in Equation (1). Indeed, we have

$$\mathscr{R}_n = \left\{ a_0 + a_1\zeta_n + a_2\zeta_n^2 + \cdots + a_{\varphi(n)-1}\zeta_n^{\varphi(n)-1} \mid a_0, \ldots, a_{\varphi(n)-1} \in \mathbb{D} \right\}, \tag{2}$$

where $\mathbb{D} = \{a/2^\ell \mid a \in \mathbb{Z} \text{ and } \ell \in \mathbb{N}\}$ is the ring of **dyadic fractions**. Every element $u \in \mathscr{R}_n$ can be uniquely expressed as in Equation (2).

If n is divisible by 8, then $n = 8d$, so that $\zeta_n^{2d} = \zeta_4 = i$, $\zeta_n^d = \zeta_8$, and $\zeta_n^d + \zeta_n^{-d} = \zeta_8 + \zeta_8^\dagger = \sqrt{2}$. Hence, in that case, we have $i \in \mathscr{R}_n$ and $1/\sqrt{2} \in \mathscr{R}_n$, which implies that the entries of the gates H, S, CX, and T_n belong to $\mathscr{R}_n$. Thus, whenever n is divisible by 8, any matrix that can be represented by a circuit over $\mathscr{G}_n$ has entries in $\mathscr{R}_n$.

2.2 The ring $\mathbb{Z}[\zeta_{12}]$

The ring $\mathbb{Z}[\zeta_{12}]$ will play an important role in Section 7. We now record some of its relevant properties. We know from Equation (1) that

$$\mathbb{Z}[\zeta_{12}] = \left\{ a_0 + a_1\zeta_{12} + a_2\zeta_{12}^2 + a_3\zeta_{12}^3 \mid a_0, a_1, a_2, a_3 \in \mathbb{Z} \right\}.$$

We define $\delta \in \mathbb{Z}[\zeta_{12}]$ as $\delta = 1 + \zeta_{12}^3 = 1 + i$. The cyclotomic integer δ is prime in $\mathbb{Z}[\zeta_{12}]$ and the prime factorization of 2 in $\mathbb{Z}[\zeta_{12}]$ is given by

$$2 = (1+i)^2(-i) = \delta^2(-i). \tag{3}$$

Now consider an element $u \in \mathscr{R}_{12}$. By Equations (2) and (3), we can write u as $u = u'/\delta^\ell$, with $u' \in \mathbb{Z}[\zeta_{12}]$ and $\ell \in \mathbb{N}$. The smallest such ℓ is called the **least denominator exponent of** u and is denoted $\mathrm{lde}(u)$. Equivalently, $\mathrm{lde}(u)$ is the smallest $\ell \in \mathbb{N}$ such that $\delta^\ell u \in \mathbb{Z}[\zeta_{12}]$. More generally, if M is a matrix (or a vector) with entries in $\mathscr{R}_{12}$, then $\mathrm{lde}(M)$ is the smallest ℓ such that $\delta^\ell M$ is a matrix (or a vector) over $\mathbb{Z}[\zeta_{12}]$.

Proposition 2.1. *We have:*

- $\mathbb{Z}[\zeta_{12}]/(2) = \{a_0 + a_1\zeta_{12} + a_2\zeta_{12}^2 + a_3\zeta_{12}^3 \mid a_0, a_1, a_2, a_3 \in \{0,1\}\}$, *and*
- $\mathbb{Z}[\zeta_{12}]/(\delta) = \{0, 1, \zeta_{12}, \zeta_{12}^2\}$.

Proof. Let $u = a_0 + a_1\zeta_{12} + a_2\zeta_{12}^2 + a_3\zeta_{12}^3 \in \mathbb{Z}[\zeta_{12}]$. Since $2 \equiv 0 \pmod 2$, the coefficients a_0, a_1, a_2, and a_3 can be chosen in $\{0,1\}$, which establishes the first item in the proposition. For the second item, notice that, since $\delta = 1 + \zeta_{12}^3$, we have $1 + \zeta_{12}^3 \equiv 0 \pmod \delta$ so that $\zeta_{12}^3 \equiv -1 \equiv 1 \pmod \delta$. This, together with the fact that $2 \equiv 0 \pmod \delta$, implies that any $u \in \mathbb{Z}[\zeta_{12}]/(\delta)$ can be written as $u = a_0 + a_1\zeta_{12} + a_2\zeta_{12}^2$ with $a_0, a_1, a_2 \in \{0,1\}$. Since we have the cyclotomic relation $\zeta_{12}^4 - \zeta_{12}^2 + 1 = 0$, we get $\zeta_{12}^2 \equiv \zeta_{12} + 1 \pmod \delta$, from which the second item in the proposition then follows. □

The **quadratic integer ring** $\mathbb{Z}[\sqrt{3}]$ is the smallest subring of $\mathbb{C}$ containing $\mathbb{Z}$ and $\sqrt{3}$. The elements of $\mathbb{Z}[\sqrt{3}]$ can be characterized as $\mathbb{Z}[\sqrt{3}] = \{a + b\sqrt{3} \mid a, b \in \mathbb{Z}\}$ and every element of $\mathbb{Z}[\sqrt{3}]$ can be uniquely written as such a $\mathbb{Z}$-linear combination of 1 and $\sqrt{3}$. Because $\sqrt{3} = \zeta_{12} + \zeta_{12}^\dagger$, we have $\mathbb{Z}[\sqrt{3}] \subseteq$

$u \pmod{\delta}$	$u \pmod 2$	$u^\dagger u \pmod 2$
0	0	0
0	$1+\zeta_{12}^3$	0
0	$1+\zeta_{12}+\zeta_{12}^2$	0
0	$\zeta_{12}+\zeta_{12}^2+\zeta_{12}^3$	0
1	1	1
1	ζ_{12}^3	1
1	$\zeta_{12}+\zeta_{12}^2 \equiv_2 \zeta_{12}(1+\zeta_{12})$	$\sqrt{3}$
1	$1+\zeta_{12}+\zeta_{12}^2+\zeta_{12}^3 \equiv_2 \zeta_{12}^4(1+\zeta_{12})$	$\sqrt{3}$
ζ_{12}	ζ_{12}	1
ζ_{12}	$1+\zeta_{12}^2 \equiv_2 \zeta_{12}^4$	1
ζ_{12}	$\zeta_{12}^2+\zeta_{12}^3 \equiv_2 \zeta_{12}^2(1+\zeta_{12})$	$\sqrt{3}$
ζ_{12}	$1+\zeta_{12}+\zeta_{12}^3 \equiv_2 \zeta_{12}^5(1+\zeta_{12})$	$\sqrt{3}$
ζ_{12}^2	ζ_{12}^2	1
ζ_{12}^2	$1+\zeta_{12}$	$\sqrt{3}$
ζ_{12}^2	$1+\zeta_{12}^2+\zeta_{12}^3 \equiv_2 \zeta_{12}^3(1+\zeta_{12})$	$\sqrt{3}$
ζ_{12}^2	$\zeta_{12}+\zeta_{12}^3 \equiv_2 \zeta_{12}^5$	1

Table 1: The possible residues for $u \in \mathbb{Z}[\zeta_{12}]$ modulo δ and 2, as well as those for $u^\dagger u$ modulo 2.

$\mathbb{Z}[\zeta_{12}]$. Since $\zeta_{12}^6 = -1$ and $\zeta_{12}^4 - \zeta_{12}^2 + 1 = 0$, we have $\zeta_{12}^\dagger = \zeta_{12} - \zeta_{12}^3$. It can then be verified by direct computation that if $u = a + b\zeta_{12} + c\zeta_{12}^2 + d\zeta_{12}^3 \in \mathbb{Z}[\zeta_{12}]$, then

$$u^\dagger u = ((a^2+c^2+ac)+(b^2+d^2+bd))+(ab+bc+cd)\sqrt{3}. \tag{4}$$

In particular, the Euclidean norm of $u \in \mathbb{Z}[\zeta_{12}]$ belongs to $\mathbb{Z}[\sqrt{3}]$, i.e., if $u \in \mathbb{Z}[\zeta_{12}]$, then $u^\dagger u \in \mathbb{Z}[\sqrt{3}]$.

Lemma 2.2. *We have:*

- *if $u \in \mathbb{Z}[\zeta_{12}]$ and $u \equiv_\delta 0$, then $u^\dagger u \equiv_2 0$,*
- *if $u \in \mathbb{Z}[\zeta_{12}]$ and $u \not\equiv_\delta 0$, then $u^\dagger u \equiv_2 1$ or $u^\dagger u \equiv_2 \sqrt{3}$, and*
- *if $u, v \in \mathbb{Z}[\zeta_{12}]$ and $u^\dagger u \equiv_2 v^\dagger v$, then $u \equiv_2 \zeta_{12}^m v$ for some integer m.*

Proof. By inspection of Table 1, which lists the possible residues of $u \in \mathbb{Z}[\zeta_{12}]$ modulo δ and 2, as well as the possible residues of $u^\dagger u$ modulo 2. The calculations leading to the construction of Table 1 can be verified using the congruences from the proof of Proposition 2.1, in addition to the following facts: $\delta^\dagger \delta = 2$, $(1+\zeta_{12})^\dagger(1+\zeta_{12}) = 2+\sqrt{3} \equiv_2 \sqrt{3}$, and $\zeta_{12}^4 \equiv_2 \zeta_{12}^2 + 1$. □

3 Matrices and circuits

Let R be a commutative ring. We write $\mathrm{M}(R)$ for the collection of all square matrices with entries in R, and $\mathrm{M}_m(R)$ for the ring of $m \times m$ matrices in R. When R is a subring of $\mathbb{C}$ that is closed under complex conjugation, we write $\mathrm{U}(R)$ for the collection of all unitary matrices with entries in R, and $\mathrm{U}_m(R)$ for the group of $m \times m$ unitary matrices with entries in R.

We now introduce certain matrices which will be useful in Section 7. Let $c \in \mathbb{C}$. For $0 \leq j \leq m-1$, the **one-level operator of type** c is the $m \times m$ matrix $c_{[j]}$ defined as

$$c_{[j]} = \begin{array}{c} \\ \vdots \\ j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & j & \cdots \end{array} \\ \begin{bmatrix} I & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & I \end{bmatrix} \end{array}$$

Similarly, let $M \in \mathrm{M}_2(\mathbb{C})$. For $0 \leq j < j' \leq m-1$, the **two-level operator of type** M is the $m \times m$ matrix $M_{[j,j']}$ defined as

$$M_{[j,j']} = \begin{array}{c} \\ \vdots \\ j \\ \\ j' \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \cdots & j & \cdots & j' & \cdots \end{array} \\ \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & M_{1,1} & 0 & M_{1,2} & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & M_{2,1} & 0 & M_{2,2} & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \end{array}$$

The one-level operator $c_{[j]}$ acts on an m-dimensional vector $\mathbf{u}$ by scaling its j-th component by c and leaving the remaining components unchanged. The two-level operator $M_{[j,j']}$ similarly acts as M on the j-th and j'-th components of $\mathbf{u}$ and leaves the remaining components unchanged. Note that if $|c| = 1$ then $c_{[j]}$ is unitary, and that if $M \in \mathrm{U}_2(\mathbb{C})$, then $M_{[j,j']}$ is unitary.

We close this section by recalling the existence of some well-known circuit constructions which will be useful in what follows. As mentioned in Section 1, when n is divisible by 8, the gate set $\mathscr{G}_n$ subsumes the Clifford+T gate set. Hence, any matrix that can be represented by a Clifford+T circuit can also be represented by a circuit over $\mathscr{G}_n$. As a consequence, the one-level operators of type ζ_n and the two-level operators of type X and H can be represented exactly over the gate set $\mathscr{G}_n$ using a single ancilla.

Theorem 3.1 (Giles & Selinger)**.** *The one- and two-level operators of type ζ_n, X, and H can each be exactly represented by a circuit over $\mathscr{G}_n$ using at most one ancilla.*

Proof. Circuit constructions for the one- and two-level operators of type ζ_8, X, and H using a single ancilla can be found in [8, Section 5]. To obtain a circuit for the one-level operator of type ζ_n, it suffices to replace T with T_n where appropriate. □

4 Determinants

The **determinant** is an important function on matrices. We will be interested in the determinant of certain block matrices.

Let R be a commutative ring and let M be an $m \times m$ matrix with entries in R. The **determinant of** M **with respect to** R, denoted by $\det_R(M)$, is defined as

$$\det_R(M) = \sum_{\sigma \in S_m} \mathrm{sgn}(\sigma) \prod_{j=1}^{m} M_{j,\sigma(j)},$$

where S_m is the symmetric group of degree m and $\mathrm{sgn}(\sigma)$ is the sign of the permutation $\sigma \in S_m$. If R is a subring of some ring R' and M is a matrix with entries in R, we have $\det_R(M) = \det_{R'}(M)$. For simplicity,

when the ring R in which the determinant is to be computed is clear from context, we sometimes write $\det(M)$ rather than $\det_R(M)$. The determinant is multiplicative, in the sense that for matrices M and M' over R of compatible dimensions, we have $\det_R(MM') = \det_R(M)\det_R(M')$. Moreover, if M is a complex unitary matrix, then $|\det_{\mathbb{C}}(M)| = 1$.

If $\phi : R \to R'$ is a ring homomorphism, then the entrywise application of ϕ is a ring homomorphism $\mathrm{M}_m(R) \to \mathrm{M}_m(R')$. By a slight abuse of notation, we use the symbol ϕ to denote both the homomorphism $R \to R'$ and its entrywise extension $\mathrm{M}(R) \to \mathrm{M}(R')$. Now consider a matrix M over R. Since ϕ is a ring homomorphism and the determinant is a polynomial in the matrix entries, we have

$$\phi(\det_R(M)) = \phi\left(\sum_{\sigma\in S_m} \operatorname{sgn}(\sigma)\prod_{j=1}^{m} M_{j,\sigma(j)}\right) = \sum_{\sigma\in S_m} \operatorname{sgn}(\sigma)\prod_{j=1}^{m} \phi\left(M\right)_{j,\sigma(j)} = \det_{R'}(\phi(M)). \tag{5}$$

Let R be a commutative ring and let R' be a commutative subring of $\mathrm{M}_m(R)$. Now consider a matrix M in $\mathrm{M}_{m'}(R')$. The matrix M is an $m' \times m'$ block matrix whose blocks are $m \times m$ matrices over R. We can therefore compute the determinant of M with respect to R', and then the determinant of the resulting matrix with respect to R. Alternatively, we can "open" the blocks and think of M as an $mm' \times mm'$ matrix over R to directly compute the determinant of M with respect to R. The following theorem, whose proof can be found in [16, Theorem 1], states that these two quantities are equal.

Theorem 4.1 (Silvester)**.** *Let R be a commutative ring, let R' be a commutative subring of $\mathrm{M}_m(R)$, and let $M \in \mathrm{M}_{m'}(R')$. Then*

$$\det_R M = \det_R(\det_{R'} M).$$

It will be convenient for us to consider a variant of Theorem 4.1 where R' is not a subring of $\mathrm{M}_m(R)$ but simply related to one.

Corollary 4.2. *Let R and R' be two commutative rings, let $\phi : R' \to \mathrm{M}_m(R)$ be a ring homomorphism, and let $M \in \mathrm{M}_{m'}(R')$. Then*

$$\det_R(\phi(M)) = \det_R(\phi(\det_{R'}(M))).$$

Proof. Let $Q = \phi[R']$ be the direct image of R' under ϕ. Then Q is a commutative subring of $\mathrm{M}_m(R)$, since R' is commutative. Hence, by Theorem 4.1 and Equation (5),

$$\det_R(\phi(M)) = \det_R(\det_Q(\phi(M))) = \det_R(\phi(\det_{R'}(M))),$$

as desired. □

5 Catalytic embeddings

We now introduce **catalytic embeddings** [1, 2, 9]. Let $\mathscr{U}$ and $\mathscr{V}$ be two sets of unitary matrices. An *m-dimensional catalytic embedding* from $\mathscr{U}$ into $\mathscr{V}$ is a pair $(\phi, \mathbf{c})$ where $\phi : \mathscr{U} \to \mathscr{V}$ is a function and $\mathbf{c} \in \mathbb{C}^m$ is a unit vector such that

1. If $U \in \mathscr{U}$ has dimension d, then $\phi(U) \in \mathscr{V}$ has dimension md, and
2. For any $\mathbf{u} \in \mathbb{C}^d$, $\phi(U)(\mathbf{u} \otimes \mathbf{c}) = (U\mathbf{u}) \otimes \mathbf{c}$.

We often write $(\phi, \mathbf{c}) : \mathscr{U} \to \mathscr{V}$ to indicate that $(\phi, \mathbf{c})$ is a catalytic embedding from $\mathscr{U}$ to $\mathscr{V}$. The composition of an m-dimensional catalytic embedding $(\phi, \mathbf{c}) : \mathscr{U} \to \mathscr{V}$ and an m'-dimensional catalytic

embedding $(\phi', \mathbf{c}') : \mathscr{V} \to \mathscr{W}$ is the mm'-dimensional catalytic embedding $(\phi' \circ \phi, \mathbf{c} \otimes \mathbf{c}') : \mathscr{U} \to \mathscr{W}$. The catalytic embedding $(\mathrm{id}_{\mathscr{U}}, 1) : \mathscr{U} \to \mathscr{U}$ acts as the identity for this notion of composition.

We now discuss two specific catalytic embeddings which will be important in the rest of this paper. Let $k \geq 2$ and define the matrix Λ_k and the vector $\mathbf{c}_k$ by

$$\Lambda_k = \begin{bmatrix} 0 & 1 \\ \zeta_{2^{k-1}} & 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{c}_k = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \zeta_{2^k} \end{bmatrix}.$$

We can use Λ_k and $\mathbf{c}_k$ to define a catalytic embedding, following [1, 2]. Consider a matrix $M \in \mathrm{M}(\mathscr{R}_{2^k})$. Then M can be uniquely written as $M = A + B\zeta_{2^k}$, for some $A, B \in \mathrm{M}(\mathscr{R}_{2^{k-1}})$, so that we can construct the matrix $A \otimes I_2 + B \otimes \Lambda_k \in \mathrm{M}(\mathscr{R}_{2^{k-1}})$. It can be shown that the assignment

$$A + B\zeta_{2^k} \longmapsto A \otimes I_2 + B \otimes \Lambda_k \tag{6}$$

defines, for every m, a ring homomorphism $\phi_k : \mathrm{M}_m(\mathscr{R}_{2^k}) \to \mathrm{M}_{2m}(\mathscr{R}_{2^{k-1}})$. Moreover, we have $\phi_k(M^\dagger) = \phi_k(M)^\dagger$ for every M, so that ϕ_k restricts to a group homomorphism $\mathrm{U}_m(\mathscr{R}_{2^k}) \to \mathrm{U}_{2m}(\mathscr{R}_{2^{k-1}})$. Now let $\mathbf{u}$ be an arbitrary vector. Then

$$\begin{aligned}
\phi_k(U)(\mathbf{u} \otimes \mathbf{c}_k) &= (A \otimes I + B \otimes \Lambda_k)(\mathbf{u} \otimes \mathbf{c}_k) \\
&= (A \otimes I)(\mathbf{u} \otimes \mathbf{c}_k) + (B \otimes \Lambda_k)(\mathbf{u} \otimes \mathbf{c}_k) \\
&= A\mathbf{u} \otimes I\mathbf{c}_k + B\mathbf{u} \otimes \Lambda_k \mathbf{c}_k \\
&= A\mathbf{u} \otimes \mathbf{c}_k + B\mathbf{u} \otimes \zeta_{2^k}\mathbf{c}_k \\
&= A\mathbf{u} \otimes \mathbf{c}_k + B\zeta_{2^k}\mathbf{u} \otimes \mathbf{c}_k \\
&= (A\mathbf{u} + B\zeta_{2^k}\mathbf{u}) \otimes \mathbf{c}_k \\
&= (U\mathbf{u}) \otimes \mathbf{c}_k.
\end{aligned}$$

The pair $(\phi_k, \mathbf{c}_k)$ is therefore a catalytic embedding. For future reference, we record this fact in the proposition below.

Proposition 5.1. *Let $k \geq 2$. Then $(\phi_k, \mathbf{c}_k)$ is a 2-dimensional catalytic embedding from $\mathrm{U}(\mathscr{R}_{2^k})$ to $\mathrm{U}(\mathscr{R}_{2^{k-1}})$.*

By identifying $\mathrm{M}_1(\mathscr{R}_{2^k})$ with $\mathscr{R}_{2^k}$, we can think of the function ϕ_k, when restricted to $\mathscr{R}_{2^k}$ as a ring homomorphism $\mathscr{R}_{2^k} \to \mathrm{M}_2(\mathscr{R}_{2^{k-1}})$. The function ϕ_k as defined by Equation (6) is then the entrywise extension of ϕ_k from $\mathscr{R}_{2^k}$ to $\mathrm{M}(\mathscr{R}_{2^k})$. We can therefore use Corollary 4.2 to get, for $U \in \mathrm{U}(\mathscr{R}_{2^k})$, an expression for the determinant of $\phi_k(U)$.

Corollary 5.2. *Let $k \geq 2$ and let $U \in \mathrm{U}_n(\mathscr{R}_{2^k})$. Then we have*

$$\det_{\mathscr{R}_{2^{k-1}}}(\phi_k(U)) = \det_{\mathscr{R}_{2^{k-1}}}(\phi_k(\det_{\mathscr{R}_{2^k}}(U))).$$

In other words, Corollary 5.2 states that, for $U \in \mathrm{U}(\mathscr{R}_{2^k})$, to compute the determinant of $\phi_k(U)$ over $\mathscr{R}_{2^{k-1}}$, one can first compute the determinant u of U over $\mathscr{R}_{2^k}$, and then compute the determinant of $\phi_k(u)$ over $\mathscr{R}_{2^{k-1}}$.

Remark 5.3. The function $\det_{\mathscr{R}_{2^{k-1}}} \circ \phi_k : \mathscr{R}_{2^k} \to \mathscr{R}_{2^{k-1}}$ is known in number theory as the **relative norm** of the field extension $\mathbb{Q}(\zeta_{2^k})/\mathbb{Q}(\zeta_{2^{k-1}})$, and is often denoted by $\mathrm{N}_{\mathbb{Q}(\zeta_{2^k})/\mathbb{Q}(\zeta_{2^{k-1}})}$. Corollary 5.2 therefore states that the determinant (over $\mathscr{R}_{2^{k-1}}$) of $\phi_k(U)$ is the relative norm of the determinant (over $\mathscr{R}_{2^k}$) of U, i.e., $\det_{\mathscr{R}_{2^{k-1}}} \circ \phi_k = \mathrm{N}_{\mathbb{Q}(\zeta_{2^k})/\mathbb{Q}(\zeta_{2^{k-1}})} \circ \det_{\mathscr{R}_{2^k}}$.

We close this section by introducing a second catalytic embedding. The construction is essentially the same as in the definition of $(\phi_k, \mathbf{c}_k)$. We define the matrix Γ_k and the vector $\mathbf{d}_k$ by

$$\Gamma_k = \begin{bmatrix} 0 & 1 \\ \zeta_{3\cdot 2^{k-1}} & 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{d}_k = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \zeta_{3\cdot 2^k} \end{bmatrix}.$$

We then define the function $\psi_k : \mathrm{M}(\mathscr{R}_{3\cdot 2^k}) \to \mathrm{M}(\mathscr{R}_{3\cdot 2^{k-1}})$ by

$$\psi_k(A + B\zeta_{2^k}) = A \otimes I_2 + B \otimes \Gamma_k.$$

Reasoning as above, we obtain the proposition below.

Proposition 5.4. *Let $k \geq 2$. Then $(\psi_k, \mathbf{d}_k)$ is a 2-dimensional catalytic embedding from $\mathrm{U}(\mathscr{R}_{3\cdot 2^k})$ to $\mathrm{U}(\mathscr{R}_{3\cdot 2^{k-1}})$.*

An analogue of Corollary 5.2 holds for $(\psi_k, \mathbf{d}_k)$, but we omit it here, since it will not be useful for our purposes.

6 Clifford-cyclotomic circuits of degree 2^k

We now turn to the exact synthesis of circuits for matrices with entries in the ring $\mathscr{R}_{2^k}$. We will take advantage of the following result, which was established in [8, Lemma 7].

Theorem 6.1 (Giles & Selinger)**.** *If U is a $2^m \times 2^m$ matrix with entries in $\mathscr{R}_8$ and $\det(U) = 1$, then U can be exactly represented by an ancilla-free m-qubit circuit over $\mathscr{G}_8$.*

Let U be a unitary matrix with entries in $\mathscr{R}_{16}$. Then the determinant of U over $\mathscr{R}_{16}$ is an element of $\mathscr{R}_{16}$ of norm 1. The next lemma shows that this determinant must be a power of ζ_{16}. The proof is relegated to Appendix A.

Lemma 6.2. *Let $u \in \mathscr{R}_{16}$ be such that $|u| = 1$. Then $u = \zeta_{16}^{\ell}$ for some $0 \leq \ell \leq 15$.*

In order to save an ancilla in synthesizing circuits over $\mathscr{G}_{16}$ our strategy is the following. Consider an m-qubit unitary U with entries in $\mathscr{R}_{16}$. By Lemma 6.2, the determinant of U is a power of ζ_{16}. Hence, multiplying U by an appropriate power of the one-level operator of type ζ_{16} if needed, we can assume without loss of generality that U has determinant 1. By Corollary 5.2, $\phi_4(U)$ is then an $(m+1)$-qubit unitary of determinant 1 with entries in $\mathscr{R}_8$ and it can thus be represented by an $(m+1)$-qubit circuit by Theorem 6.1.

Theorem 6.3. *A $2^m \times 2^m$ matrix U can be exactly represented by an m-qubit circuit over $\mathscr{G}_{16}$ if and only if $U \in \mathrm{U}_{2^m}(\mathscr{R}_{16})$. Furthermore, a single ancilla suffices to synthesize a circuit for U.*

Proof. The left-to-right implication follows immediately from the fact that all the elements of $\mathscr{G}_{16}$ belong to $\mathrm{U}(\mathscr{R}_{16})$. For the right-to-left implication, let $U \in \mathrm{U}_{2^m}(\mathscr{R}_{16})$. Since U is unitary, $\det_{\mathscr{R}_{16}}(U)$ is an element of $\mathscr{R}_{16}$ of norm 1. Thus, by Lemma 6.2, we have that $\det_{\mathscr{R}_{16}}(U) = \zeta_{16}^{\ell}$ for some $0 \leq \ell \leq 15$. Now let P be the fully-controlled phase gate

$$P = \mathrm{diag}(1, 1, \ldots, 1, \zeta_{16}^{\ell}),$$

and let $V = P^{\dagger}U$. Note that $\det_{\mathscr{R}_{16}}(V) = \det_{\mathscr{R}_{16}}(P^{\dagger}) \det_{\mathscr{R}_{16}}(U) = 1$. Now consider the catalytic embedding $(\phi_4, \mathbf{c}_4)$ defined in Section 5. By Corollary 5.2, we have

$$\det_{\mathscr{R}_8}(\phi_4(V)) = \det_{\mathscr{R}_8}(\phi_4(\det_{\mathscr{R}_{16}}(V))) = \det_{\mathscr{R}_8}(I_2) = 1.$$

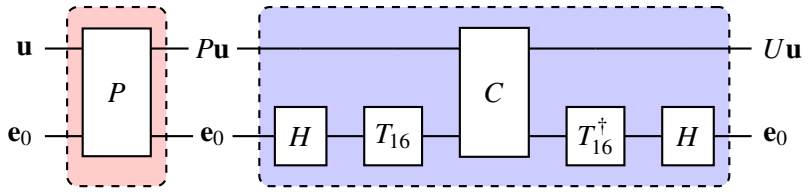


Figure 1: The circuit constructed in the proof of Theorem 6.3.

Hence, $\phi_4(V) \in \mathrm{U}_{2^{m+1}}(\mathscr{R}_8)$ and $\det_{\mathscr{R}_8}(\phi_4(V)) = 1$ so that, by Theorem 6.1, $\phi_4(V)$ can be represented by an ancilla-free circuit C over $\mathscr{G}_8$. Now let D be the circuit $D = (I_m \otimes (T_{16}H))^\dagger \circ C \circ (I_m \otimes (T_{16}H))$ over $\mathscr{G}_{16}$. Then $T_{16}H\mathbf{e}_0 = \mathbf{c}_4$, so that, for an arbitrary $\mathbf{u}$, we have:

$$\begin{aligned} D(\mathbf{u} \otimes \mathbf{e}_0) &= (I_m \otimes (T_{16}H))^\dagger \circ C \circ (I_m \otimes (T_{16}H))(\mathbf{u} \otimes \mathbf{e}_0) \\ &= (I_m \otimes (T_{16}H))^\dagger \circ C(\mathbf{u} \otimes \mathbf{c}_4) \\ &= (I_m \otimes (T_{16}H))^\dagger \circ \phi_4(V)(\mathbf{u} \otimes \mathbf{c}_4) \\ &= (I_m \otimes (T_{16}H))^\dagger((V\mathbf{u}) \otimes \mathbf{c}_4) \\ &= (V\mathbf{u}) \otimes \mathbf{e}_0. \end{aligned}$$

Hence D exactly represents V over $\mathscr{G}_{16}$ using a single ancilla. Moreover, P can also be represented by a circuit over $\mathscr{G}_{16}$ using a single ancilla. Indeed, this follows from Theorem 3.1, since P is (a power of) a one-level operator of type ζ_{16}. Thus, the unitary U can be represented using a single ancilla as well. □

The circuit constructed in the proof of Theorem 6.3 is depicted in Figure 1. We note that, if U is a unitary with entries in $\mathscr{R}_{16}$ and the dimension of U is larger than 4, then an ancilla is necessary to synthesize a matrix for a U. As a result, the construction of Theorem 6.3 uses the minimal number of ancillas (in the worst case).

Corollary 6.4. *Let $k \geq 4$. A $2^m \times 2^m$ matrix U can be exactly represented by an m-qubit circuit over $\mathscr{G}_{2^k}$ if and only if $U \in U_{2^m}(\mathscr{R}_{2^k})$. Furthermore, $k-3$ ancillas suffice to synthesize a circuit for U.*

Proof. By induction, using Theorem 6.3 as the base case, and a construction similar to the one given in Figure 1 in the inductive step. □

7 Clifford-cyclotomic circuits of degree $3 \cdot 2^k$

We now turn to the exact synthesis of circuits for matrices with entries in the ring $\mathscr{R}_{3\cdot 2^k}$. Our strategy is to first prove that every matrix with entries in $\mathscr{R}_{24}$ can be represented by a circuit over $\mathscr{G}_{24}$, and to then use an inductive argument like the one used in proving Corollary 6.4. In order to establish the result for $n = 24$, we start by showing that every unitary matrix with entries in $\mathscr{R}_{12}$ can be factored as a product of convenient generators.

7.1 Generating $\mathrm{U}_n(\mathscr{R}_{12})$

To generate $\mathrm{U}_n(\mathscr{R}_{12})$, we use one-level operators of type ζ_{12} and two-level operators of type X and H', where

$$H' = \zeta_8 H = \frac{\delta}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Note that $H' \in \mathrm{U}(\mathscr{R}_{12})$. We follow [3, 8] and first show that a unit vector with entries in $\mathscr{R}_{12}$ can be mapped to a standard basis vector using one- and two-level operators of type ζ_{12}, X, and H'. To this end, we proceed by induction on the least denominator exponent of the vector.

Lemma 7.1. *Let a and b be integers, at least one of which is nonzero. Then $a^2+b^2+ab \geq 1$, and equality is achieved exactly when (a,b) is one of $(\pm 1, 0)$, $(0,\pm 1)$, or $(\pm 1, \mp 1)$.*

Proof. First notice that if $a \neq 0$ and $b = 0$, then $a^2+b^2+ab = a^2 \geq 1$, since $a \in \mathbb{Z}$. A similar argument applies when $a = 0$ and $b \neq 0$. Equality in the first of these cases happens exactly when $a = \pm 1$ and $b = 0$, while equality in the second of these cases happens exactly when $a = 0$ and $b = \pm 1$. Now suppose that a and b are both nonzero, and assume, without loss of generality, that $|a| \geq |b|$. We then have $a^2 \geq |ab|$, so that $a^2 + ab \geq 0$. Hence,

$$a^2 + ab + b^2 \geq b^2 \geq 1,$$

since $b \in \mathbb{Z}$. Equality in this case happens exactly when $b^2 = 1$ and $a^2 + ab = 0$ which, in turn, happens exactly when $b = \pm 1$ and $a = \mp 1$. □

Lemma 7.2. *If $\boldsymbol{u}$ is an m-dimensional unit vector with entries in $\mathscr{R}_{12}$ and $\mathrm{lde}(\boldsymbol{u}) = 0$, then, for any $0 \leq j \leq m-1$, there exists a sequence $G_1, \ldots, G_q$ of one- and two-level operators of type ζ_{12}, X, and H' such that $G_1 \cdots G_q \boldsymbol{u} = \boldsymbol{e}_j$.*

Proof. It suffices to show that $\mathbf{u} = \zeta_{12}^{\ell} \mathbf{e}_{j'}$ for some integers j' and ℓ, since, if $\mathbf{u}$ is of that form, then it can be mapped to $\mathbf{e}_j$ by applying the appropriate operators of type ζ_{12} and X. Because $\mathbf{u}$ is a unit vector, we have $\mathbf{u}^\dagger \mathbf{u} = 1$. Hence, using Equation (4), we get

$$1 = \mathbf{u}^\dagger \mathbf{u} = \sum_j u_j^\dagger u_j = \sum_j ((a_j^2 + c_j^2 + a_j c_j) + (b_j^2 + d_j^2 + b_j d_j)) + (a_j b_j + b_j c_j + c_j d_j)\sqrt{3}.$$

Since every element $\mathbb{Z}[\sqrt{3}]$ can be uniquely expressed as an integer linear combination of 1 and $\sqrt{3}$, the equation above implies the equations below.

$$\sum_j ((a_j^2 + c_j^2 + a_j c_j) + (b_j^2 + d_j^2 + b_j d_j)) = 1 \tag{7}$$

$$\sum_j (a_j b_j + b_j c_j + c_j d_j) = 0 \tag{8}$$

It follows from Lemma 7.1 that Equation (7) can only be satisfied if there is exactly one index j such that either $a_j^2 + c_j^2 + a_j c_j = 1$ or $b_j^2 + d_j^2 + b_j d_j = 1$, but not both. By Lemma 7.1, this happens precisely when (a_j, c_j) is one of $(\pm 1, 0)$, $(0, \pm 1)$, or $(\pm 1, \mp 1)$, and (b_j, d_j) is $(0,0)$, or when (b_j, d_j) is one of $(\pm 1, 0)$, $(0, \pm 1)$, or $(\pm 1, \mp 1)$, and (a_j, c_j) is $(0,0)$. These 12 solutions all satisfy Equation (8) and correspond exactly to the possible powers of ζ_{12}. □

Lemma 7.3. *If $u, v \in \mathbb{Z}[\zeta_{12}]$ are such that $u^\dagger u \equiv_2 v^\dagger v$, then there exists ℓ such that*

$$H' T_{12}^{\ell} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u' \\ v' \end{bmatrix}$$

for some $u', v' \in \mathbb{Z}[\zeta_{12}]$ such that $u' \equiv v' \equiv 0 \pmod{\delta}$.

Proof. Let u and v be as stated. Then, by Lemma 2.2, we have $u \equiv_2 \zeta_{12}^{\ell} v$ for some ℓ. Equivalently, $u \pm \zeta_{12}^{\ell} v \equiv 0 \pmod 2$, so that $u + \zeta_{12}^{\ell} v = 2w$ and $u - \zeta_{12}^{\ell} v = 2w'$ for some $w, w' \in \mathbb{Z}[\zeta_{12}]$. We then get

$$H' T_{12}^{\ell} \begin{bmatrix} u \\ v \end{bmatrix} = H' \begin{bmatrix} u \\ \zeta_{12}^{\ell} v \end{bmatrix} = \frac{\delta}{2} \begin{bmatrix} u + \zeta_{12}^{\ell} v \\ u - \zeta_{12}^{\ell} v \end{bmatrix} = \frac{\delta}{2} \begin{bmatrix} 2w \\ 2w' \end{bmatrix} = \delta \begin{bmatrix} w \\ w' \end{bmatrix},$$

which completes the proof. □

Lemma 7.4. *If $\boldsymbol{u}$ is an m-dimensional unit vector with entries in $\mathscr{R}_{12}$ and $\mathrm{lde}(\boldsymbol{u}) \geq 1$, then there exists a sequence $G_1, \ldots G_q$ of one- and two-level operators of type ζ_{12}, X, and H' such that $\mathrm{lde}(G_1 \cdots G_q \boldsymbol{u}) < \mathrm{lde}(\boldsymbol{u})$.*

Proof. Let $k = \mathrm{lde}(\mathbf{u})$ and let $\mathbf{v} = \delta^k \mathbf{u} \in \mathbb{Z}[\zeta_{12}]$. We know from Lemma 2.2 that, for $v \in \mathbb{Z}[\zeta_{12}]$, if $v \not\equiv_\delta 0$, then $v^\dagger v \equiv_2 1$ or $v^\dagger v \equiv_2 \sqrt{3}$. We can therefore write $\mathbf{v}^\dagger \mathbf{v}$ as

$$\mathbf{v}^\dagger \mathbf{v} = \sum_j v_j^\dagger v_j = \sum_{v_j \equiv_\delta 0} v_j^\dagger v_j + \sum_{v_j \not\equiv_\delta 0} v_j^\dagger v_j = \sum_{v_j \equiv_\delta 0} v_j^\dagger v_j + \sum_{v_j^\dagger v_j \equiv_2 1} v_j^\dagger v_j + \sum_{v_j^\dagger v_j \equiv_2 \sqrt{3}} v_j^\dagger v_j. \tag{9}$$

Since $\mathbf{u}$ is a unit vector and $\delta^\dagger \delta = 2$, we have $\mathbf{v}^\dagger \mathbf{v} = \mathbf{u}^\dagger \mathbf{u} (\delta^\dagger \delta)^k = 2^k$. Because $k \geq 1$, this implies that $\mathbf{v}^\dagger \mathbf{v} \equiv_2 0$. Hence, by Lemma 2.2, taking Equation (9) modulo 2 yields

$$0 \equiv_2 \mathbf{v}^\dagger \mathbf{v} \equiv_2 a + b\sqrt{3},$$

where a is the number of v_j such that $v_j^\dagger v_j \equiv_2 1$, and b be the number of v_j such that $v_j^\dagger v_j \equiv_2 \sqrt{3}$. It then follows that we must have $a \equiv_2 b \equiv_2 0$. That is, both a and b are even integers. We can therefore group the entries of $\mathbf{v}$ that are not congruent to 0 modulo δ into pairs $(v_j, v_{j'})$ such that $v_j^\dagger v_j \equiv_2 v_{j'}^\dagger v_{j'}$. Applying Lemma 7.3 to every such pair reduces the least denominator exponent of $\mathbf{u}$. □

Lemma 7.5. *If $\boldsymbol{u}$ is an m-dimensional unit vector with entries in $\mathscr{R}_{12}$, then, for any $0 \leq j \leq m-1$, there exists a sequence $G_1, \ldots, G_q$ of one- and two-level operators of type ζ_{12}, X, and H' such that $G_1 \cdots G_q \boldsymbol{u} = \boldsymbol{e}_j$.*

Proof. By induction on $\mathrm{lde}(\mathbf{u})$. If $\mathrm{lde}(\mathbf{u}) = 0$, then the result follows from Lemma 7.2. If $\mathrm{lde}(\mathbf{u}) \geq 1$, then, by Lemma 7.4, there exists a sequence $G_1, \ldots G_q$ of two-level operators of type ζ_{12}, X, and H' such that $\mathrm{lde}(G_1 \cdots G_q \mathbf{u}) < \mathrm{lde}(\mathbf{u})$. Now let $\mathbf{u}' = G_1 \cdots G_q \mathbf{u}$. By the induction hypothesis, there exists a sequence $G_1', \ldots, G_{q'}'$ of one- and two-level operators of type ζ_{12}, X, and H such that $G_1' \cdots G_{\ell'}' \mathbf{u}' = \mathbf{e}_j$. We therefore have

$$\mathbf{e}_j = G_1' \cdots G_{q'}' \mathbf{u}' = G_1' \cdots G_{q'}' \cdot G_1 \cdots G_q \mathbf{u},$$

which completes the proof. □

Theorem 7.6. *A matrix U belongs to $\mathrm{U}(\mathscr{R}_{12})$ if and only if U can be expressed as a product of one- and two-level operators of type ζ_{12}, X, and H'.*

Proof. The right-to-left direction follows immediately from the fact that one- and two-level operators of type ζ_{12}, X, and H' are unitaries with entries in $\mathscr{R}_{12}$. We now prove the left-to-right direction. Let $U \in \mathrm{U}(\mathscr{R}_{12})$ and let $\mathbf{u}$ be the first column of U. Then $\mathbf{u}$ is a unit vector with entries in $\mathscr{R}_{12}$. Hence, by

Lemma 7.5, there exists a sequence $G_1,\ldots,G_q$ of one- and two-level operators of type ζ_{12}, X, and H' such that $G_1\cdots G_q\mathbf{u}=\mathbf{e}_1$. Since U and $G_1,\ldots,G_q$ are unitaries with entries in $\mathscr{R}_{12}$, we have

$$G_1\cdots G_qU=\left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & U' & \\ 0 & & & \end{array}\right],$$

for some smaller $U'\in \mathrm{U}(\mathscr{R}_{12})$. Repeating this process inductively, we ultimately obtain a sequence $F_1,\ldots,F_{q'}$ of one- and two-level operators of type ζ_{12}, X and H' such that $F_1\cdots F_{q'}U=I$. Multiplying by $(F_1\cdots F_{q'})^{-1}$ on both sides then yields the desired decomposition. □

7.2 Exact synthesis

Theorem 7.7. *A $2^m\times 2^m$ matrix U can be exactly represented by an m-qubit circuit over $\mathscr{G}_{24}$ if and only if $U\in \mathrm{U}_{2^m}(\mathscr{R}_{24})$. Furthermore, 2 ancillas suffice to synthesize a circuit for U.*

Proof. The left-to-right follows immediately from the fact that elements of $\mathscr{G}_{24}$ belong to $\mathrm{U}(\mathscr{R}_{24})$. Now let $U\in \mathrm{U}_{2^m}(\mathscr{R}_{24})$ and let $(\psi_3,\mathbf{d}_3):\mathrm{U}(\mathscr{R}_{24})\to \mathrm{U}(\mathscr{R}_{12})$ be the catalytic embedding defined in Section 5. Then $\psi_3(U)\in \mathrm{U}_{2^{m+1}}(\mathscr{R}_{12})$ and can be represented as a product of one- and two-level operators of type ζ_{12}, X, and H'. By Theorem 3.1, these 1- and 2-level operators can each be represented by a circuit over $\mathscr{G}_{24}$ using a single ancilla. Hence, there is a circuit C over $\mathscr{G}_{24}$ that represents $\psi_3(U)$. Using an additional ancilla and reasoning as in Theorem 6.3, we obtain a circuit over $\mathscr{G}_{24}$ for U that uses 2 ancillas. □

We can now use Theorem 7.7 to obtain an exact synthesis result for Clifford-cyclotomic gate sets of degree $3\cdot 2^k$, with $k\geq 3$. The proof is very similar to that of Corollary 6.4, so we omit it here.

Corollary 7.8. *Let $k\geq 3$. A $2^m\times 2^m$ matrix U can be exactly represented by an m-qubit circuit over $\mathscr{G}_{3\cdot 2^k}$ if and only if $U\in U_{2^m}(\mathscr{R}_{3\cdot 2^k})$. Furthermore, $k-1$ ancillas suffice to synthesize a circuit for U.*

8 Conclusion

We now know that, for $n=2^k$ and $n=3\cdot 2^k$, m-qubit circuits with ancillas over $\mathscr{G}_n$ correspond precisely to matrices in $\mathrm{U}_{2^m}(\mathscr{R}_n)$. Yet, many questions in the theory of Clifford-cyclotomic circuits remain unanswered. Two natural open problems are the following.

1. For which values of n does the correspondence between circuits over $\mathscr{G}_n$ and matrices in $\mathrm{U}(\mathscr{R}_n)$ hold?
2. For the values of n for which the correspondence holds, what is the smallest number of ancillas required to synthesize circuits in the worst case?

A natural initial step in addressing the first of these open problems is to consider values of n of the form $p\cdot 2^k$, for p a prime. While it stands to reason that our results might generalize to such cases, it gets progressively harder to analyze the relevant residues; this indicates that a novel approach may be needed for such generalizations.

Acknowledgements

We thank the QPL reviewers for their very thoughtful comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The circuit in Section 6 was drawn using the Quantikz package [12].

References

[1] Matthew Amy, Matthew Crawford, Andrew N. Glaudell, Melissa L. Macasieb, Samuel S. Mendelson & Neil J. Ross (2023): *Catalytic embeddings of quantum circuits*, doi:10.48550/arXiv.2305.07720. Preprint. Available from arXiv:2305.07720.

[2] Matthew Amy, Andrew N. Glaudell, Shaun Kelso, William Maxwell, Samuel S. Mendelson & Neil J. Ross (2024): *Exact synthesis of multiqubit Clifford-cyclotomic circuits*. In Torben Ægidius Mogensen & Łukasz Mikulski, editors: *Proceedings of the Reversible Computation, RC 2024, Torún, Poland*, Springer Nature Switzerland, Cham, pp. 238–245, doi:10.1007/978-3-031-62076-8_15. Available from arXiv:2311.07741.

[3] Matthew Amy, Andrew N. Glaudell & Neil J. Ross (2020): *Number-theoretic characterizations of some restricted Clifford+T circuits*. *Quantum* 4, p. 252, doi:10.22331/q-2020-04-06-252. Available from arXiv:1908.06076.

[4] Alex Bocharov, Martin Roetteler & Krysta M. Svore (2015): *Efficient synthesis of probabilistic quantum circuits with fallback*. *Physical Review A* 91, p. 052317, doi:10.1103/PhysRevA.91.052317. Available from arXiv:1409.3552.

[5] Guillaume Duclos-Cianci & David Poulin (2015): *Reducing the quantum-computing overhead with complex gate distillation*. *Physical Review A* 91(4), p. 042315, doi:10.1103/PhysRevA.91.042315. Available from arXiv:1403.5280.

[6] David S. Dummit & Richard M. Foote (2004): *Abstract Algebra*, 3 ed. edition. Wiley, New Delhi.

[7] Simon Forest, David Gosset, Vadym Kliuchnikov & David McKinnon (2015): *Exact synthesis of single-qubit unitaries over Clifford-cyclotomic gate sets*. *Journal of Mathematical Physics* 56(8), p. 082201, doi:10.1063/1.4927100. Available from arXiv:1501.04944.

[8] Brett Giles & Peter Selinger (2013): *Exact synthesis of multiqubit Clifford+T circuits*. *Physical Review A* 87(3), p. 032332, doi:10.1103/PhysRevA.87.032332. Also available from arXiv:1212.0506.

[9] Andrew N. Glaudell, Neil J. Ross, John van de Wetering & Lia Yeh (2024): *Exact synthesis of multiqutrit Clifford-cyclotomic circuits*. In Alejandro Díaz-Caro & Vladimir Zamdzhiev, editors: *Proceedings of the 21st International Conference on Quantum Physics and Logic, QPL 2024, Buenos Aires, Argentina, EPTCS* 406, pp. 44–62, doi:10.4204/EPTCS.406.2. Available from arXiv:2405.08136.

[10] Daniel Gottesman & Isaac L. Chuang (1999): *Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations*. *Nature* 402(6760), pp. 390–393, doi:10.1038/46503.

[11] Colin Ingalls, Bruce W. Jordan, Allan Keeton, Adam Logan & Yevgeny Zaytman (2021): *The Clifford-cyclotomic group and Euler–Poincaré characteristics*. *Canadian Mathematical Bulletin* 64(3), pp. 651–666, doi:10.4153/S0008439520000727. Available from arXiv:1903.09497.

[12] Alastair Kay (2018): *Tutorial on the Quantikz package*, doi:10.48550/arXiv.1809.03842. Preprint. Available from arXiv:1809.03842.

[13] Vadym Kliuchnikov, Dmitri Maslov & Michele Mosca (2013): *Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates*. *Quantum Information & Computation* 13(7-8), pp. 607–630, doi:10.26421/QIC13.7-8-4. Available from arXiv:1206.5236.

[14] Dorian Rudolph (2024): *Towards a universal gate set for* QMA_1, doi:10.48550/arXiv.2411.02681. Preprint. Available from arXiv:2411.02681.

[15] Peter W. Shor (1997): *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. *SIAM Journal on Computing* 26(5), pp. 1484–1509, doi:10.1137/S0097539795293172. Available from arXiv:quant-ph/9508027.

[16] John R. Silvester (2000): *Determinants of block matrices*. *The Mathematical Gazette* 84(501), pp. 460–467, doi:10.2307/3620776.

[17] Lawrence C. Washington (1982): *Introduction to Cyclotomic Fields*. Springer New York, NY, doi:10.1007/978-1-4612-1934-7.

A A Proof of Lemma 6.2

Let R be a ring. A **unit** of R is an element of R that admits a multiplicative inverse. A ring R is an **integral domain** if $uv \neq 0$, for all nonzero elements $u, v \in R$. An element $u \in R$ is an **associate** of an element $v \in R$ if there exists a unit $w \in R$ such that $v = wu$. A nonunit, nonzero element $u \in R$ is called **prime** if u divides v or u divides w, whenever u divides vw. An ideal $I \subsetneq R$ is called **prime** if $uv \in I$ implies $u \in I$ or $v \in I$.

We start by recalling two well known facts, before establishing a property of roots of unity.

Proposition A.1. *Let R be a commutative ring with identity and let $u, v \in R$. Then u and v are associates if and only if $u \mid v$ and $v \mid u$.*

Proposition A.2. *Let R be a commutative ring with identity and let $u \in R$. Then u is prime if and only if the ideal generated by u is prime.*

Lemma A.3. *Let $a, b \in \mathbb{N}$ be such that $\gcd(a,b) = 1$. Then $1 - \zeta_a^b$ and $1 - \zeta_a$ are associates.*

Proof. We have

$$1 - \zeta_a^b = (1 - \zeta_a)(1 + \zeta_a + \zeta_a^2 + \ldots + \zeta_a^{b-1}).$$

Hence, $1 - \zeta_a \mid 1 - \zeta_a^b$. Now since $\gcd(a,b) = 1$, there exist $c, d \in \mathbb{Z}$ such that $ac + bd = 1$. Then $1 - \zeta_a = 1 - \zeta_a^{ac+bd} = 1 - (\zeta_a^b)^d$, and thus

$$1 - \zeta_a = 1 - (\zeta_a^b)^d = (1 - \zeta_a^b)(1 + \zeta_a^b + (\zeta_a^b)^2 + \ldots + (\zeta_a^b)^{d-1}).$$

Hence, $1 - \zeta_a^b \mid 1 - \zeta_a$. Thus, $1 - \zeta_a$ and $1 - \zeta_a^b$ are associates by Proposition A.1. □

We are now in a position to prove Lemma 6.2, whose statement we reproduce below.

Lemma. *Let $u \in \mathscr{R}_{16}$ be such that $|u| = 1$. Then $u = \zeta_{16}^{\ell}$ for some $0 \leq \ell \leq 15$.*

Proof. Let $\chi = 1 - \zeta_{16} \in \mathscr{R}_{16}$. Then χ is a prime element in $\mathbb{Z}[\zeta_{16}]$ and, by Proposition A.2, $\langle \chi \rangle$ is a prime ideal of $\mathscr{R}_{16}$. By Lemma A.3, we can decompose 2 in $\mathbb{Z}[\zeta_{16}]$ as

$$\begin{aligned}
2 &= (1+i)(1-i) \\
&= (1-i)^2 u_1 \\
&= (1-\zeta_8)^2 (1+\zeta_8)^2 u_1 \\
&= (1-\zeta_8)^4 u_2 u_1 \\
&= (1-\zeta_{16})^4 (1+\zeta_{16})^4 u_2 u_1 \\
&= (1-\zeta_{16})^8 u_3 u_2 u_1 \\
&= \chi^8 u_3 u_2 u_1,
\end{aligned}$$

where u_1, u_2, and u_3 are units in $\mathbb{Z}[i]$, $\mathbb{Z}[\zeta_8]$, and $\mathbb{Z}[\zeta_{16}]$, respectively. Hence, any $u \in \mathscr{R}_{16}$, can be written as $u = v/\chi^{\ell}$ with $\ell \in \mathbb{N}$ and $v \in \mathbb{Z}[\zeta_{16}]$. Now let $u \in \mathscr{R}_{16}$ be such that $|u| = 1$ and write u as $u = v/\chi^{\ell}$ with ℓ minimal. Observe that $\chi^{\dagger} = 1 - \zeta_{16}^{\dagger} = -\zeta_{16}^{\dagger}\chi$. We hence have

$$1 = |u|^2 = u^{\dagger}u = \frac{v^{\dagger}v}{(-\zeta_{16}^{\dagger})^{\ell}\chi^{2\ell}},$$

so that $v^\dagger v = \chi^{2\ell} \cdot (-\zeta_{16}^\dagger)^\ell$. If $\ell > 0$, we have $v^\dagger v \in \langle \chi \rangle$. Since χ is prime, this implies that we must have either $v \in \langle \chi \rangle$ or $v^\dagger \in \langle \chi \rangle$. But $v^\dagger \in \langle \chi \rangle$ would imply $v \in \langle \chi \rangle$ since χ and $\chi^\dagger$ are associates. Hence, if $\ell > 0$, then $v \in \langle \chi \rangle$, which contradicts the minimality of ℓ. It must thus be the case that $\ell = 0$, so that u is in fact an element of $\mathbb{Z}[\zeta_{16}]$. Writing u as $u = \sum_{j=0}^{7} a_j \zeta_{16}^j$, with $a_j \in \mathbb{Z}$, we then get

$$\begin{aligned} 1 &= u^\dagger u \\ &= \sum_{j=0}^{7} a_j^2 + 2Re(\zeta_{16})((\sum_{j=0}^{6} a_j a_{j+1}) - a_0 a_7) + 2Re(\zeta_{16}^2)((\sum_{j=0}^{5} a_j a_{j+2}) - a_0 a_6 - a_1 a_7) \\ &+ 2Re(\zeta_{16}^3)((\sum_{j=0}^{4} a_j a_{j+3}) - a_0 a_5 - a_1 a_6 - a_2 a_7). \end{aligned}$$

The above equation holds when exactly one $a_j = \pm 1$ and the rest are zero. Hence $u = \zeta_{16}^\ell$ for some $0 \leq \ell \leq 15$, as desired. □

Fast Classical Simulation of Quantum Circuits via Parametric Rewriting in the ZX-Calculus

Matthew Sutcliffe
Department of Computer Science
University of Oxford
Oxford, UK
matthew.sutcliffe@cs.ox.ac.uk

Aleks Kissinger
Department of Computer Science
University of Oxford
Oxford, UK
aleks.kissinger@cs.ox.ac.uk

The ZX-calculus is an algebraic formalism that allows quantum computations to be simplified via a small number of simple graphical rewrite rules. Recently, it was shown that, when combined with a family of "sum-over-Cliffords" techniques, the ZX-calculus provides a powerful tool for classical simulation of quantum circuits. However, for several important classical simulation tasks, such as computing the probabilities associated with many measurement outcomes of a single quantum circuit, this technique results in reductions over many very similar diagrams, where much of the same computational work is repeated. In this paper, we show that the majority of this work can be shared across branches, by developing reduction strategies that can be run parametrically on diagrams with boolean free parameters. As parameters only need to be fixed after the bulk of the simplification work is already done, we show that it is possible to perform the final stage of classical simulation quickly utilising a high degree of GPU parallelism. Using these methods, we demonstrate an average speedup factor of 78.3 ± 10.2 for certain classical simulation tasks vs. the non-parametric approach.

1 Introduction

The ZX-calculus [12, 38] is a useful tool for expressing a broad range of quantum computations, including quantum circuits, as a type of labelled graph called a *ZX-diagram*, and subsequently reducing it to a simpler form using a handful of graph rewriting rules. Recently, it has been applied extensively in optimisation [15, 14, 5, 6, 16, 26, 27] and classical simulation [22, 23, 39, 10, 11, 9, 24, 36, 25, 3, 2, 34] of quantum circuits.

The latter application makes use of the ZX-calculus to simplify circuits as much as possible, before relying on the *sum-over-Cliffords* method of *stabiliser decomposition* introduced by Bravyi, Smith, and Smolin [8] as well as Bravyi et al [7]. This approach amounts to decomposing a non-Clifford quantum circuit with measurements into an exponentially large sum of efficiently reducible Clifford terms, in order to deduce the probability amplitude of particular measurement outcomes. Applying ZX-calculus simplification at each step of the decomposition, as proposed by Kissinger and van de Wetering [22], makes this process more efficient.

In this paper, we identify that when multiple measurement amplitudes or probabilities are needed for the same circuit (such as when sampling the output distribution many times or computing marginal probabilities), the entire procedure needs to be repeated from the start to compute each amplitude independently. We address this issue by adapting the rewriting rules and simplification procedure of the ZX-calculus to support boolean free parameters.

Remarkably, the graph-theoretic simplification of such diagrams can be done in a way that is agnostic to the value of these parameters, which enables us to delay fixing them until near the end of the calculation. Ultimately, this allows —- once the circuit has been reduced for the generalised case —-

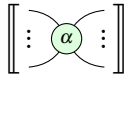
A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 247–269, doi:10.4204/EPTCS.426.10

for the respective results of different choices of free parameters to be calculated very rapidly, by simply evaluating a real-valued polynomial with boolean free variables. Furthermore, we show that such polynomials can be represented in such a way that these evaluations can be calculated in parallel on a GPU. Consequently, as the following sections will show, after the initial generalised reduction, every subsequent choice of parameters can be evaluated in a small fraction of the time that would be achieved by repeating the whole process.

2 Background

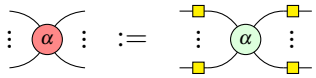
2.1 ZX-calculus

Quantum algorithms are commonly expressed graphically in the form of quantum circuit diagrams [29], which show the sequence of gates acting upon the qubits. A useful alternative notation is that of the ZX-calculus, wherein quantum computations are expressed as *ZX-diagrams* [13]. These are undirected labelled graphs composed of Z- and X- '*spiders*'. Semantically, Z-spiders can be interpreted "generalised Z-phase gates", which are $2^n \times 2^m$ matrices with a 1 in the top-left corner, $e^{i\alpha}$ in the bottom-right, and 0s elsewhere:



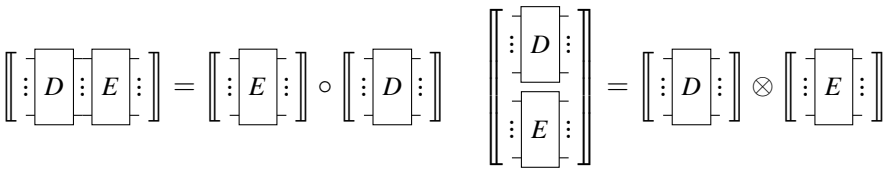
$$:= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & & e^{i\alpha} \end{pmatrix}$$

Similarly, X-spiders are generalised X-phase gates. One way to define them is by conjugating Z-spiders by Hadamard gates:



$$\text{where} \qquad [\![\text{—□—}]\!] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

More complex ZX-diagrams can be interpreted much like quantum circuits in terms of composition and tensor products of their components:



and crossing wires are interpreted as "swap" gates: $\text{SWAP}(v \otimes w) = w \otimes v$. Whereas quantum circuits always describe unitary matrices, ZX-diagrams can in general be non-unitary. In particular, a ZX-diagram with m inputs and n outputs describes a $2^n \times 2^m$ matrix. If it has no inputs or outputs, it describes a $2^0 \times 2^0 = 1 \times 1$ matrix, i.e. a scalar. An important property of ZX-diagrams is that isomorphic graphs semantically describe the same linear map. This is sometimes stated as a "meta-rule" of ZX-calculus: *only connectivity matters*.

The ZX-calculus comes equipped with a number of *rewrite rules* that describe how certain structures within a ZX-diagram may be re-expressed as an equivalent (generally simplified) structure. Figure 1 shows the elementary rewrite rules. The rules in Figure 1 are sufficient to efficiently reduce any *Clifford* ZX-diagrams (those diagrams whose angles are restricted to integer multiples of $\frac{\pi}{2}$ [4]) with no open inputs or output to a scalar.

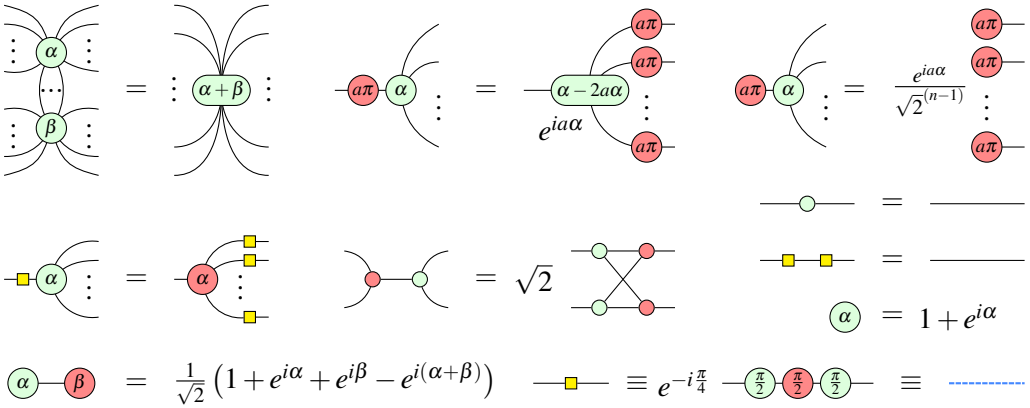


Figure 1: A set of the basic rewrite rules and scalar relations of ZX-calculus, where Greek letters denote arbitrary real variables, $[0,2\pi)$, and Latin letters arbitrary boolean variables, $\{0,1\}$. Note that the rules still apply if all the spider colours are inverted.

When extending to the Clifford+T gateset, which is sufficient to approximately express any quantum map [4, 28, 37], this efficiency is lost. When simplifying ZX-diagrams containing T-like gates (corresponding to spiders with phases of *odd* multiples of $\frac{\pi}{4}$), some of these T-spiders may be liable to fuse and cancel, or be otherwise transferred from the diagram to the scalar factor, but many will typically hinder further simplification. When reaching such apparent dead-ends, however, one may continue the diagram's reduction by decomposing the T-gates into sums of Clifford terms.

The BSS decomposition, for instance (introduced in [8]) allows sets of 6 T-spiders to be exchanged for sums of 7 Cliffords. As per [22], this can be expressed in ZX-calculus form as in Figure 2. Consequently, utilising such a decomposition to remove T-gates, and hence allowing the circuit to be further reduced, results in an exponential increase in the number of terms to compute (and correspondingly the runtime) versus the number of T-gates, t, in the initial simplified circuit. In the case of the BSS decomposition, this complexity is given by $7^{t/6} \approx 2^{0.47t}$. Hence, for a general decomposition complexity, $2^{\alpha t}$, BSS has an $\alpha \approx 0.47$ (where lower values of α represent more efficient decompositions). The current state of the art T-decompositions can achieve $\alpha \approx 0.396$ [31].

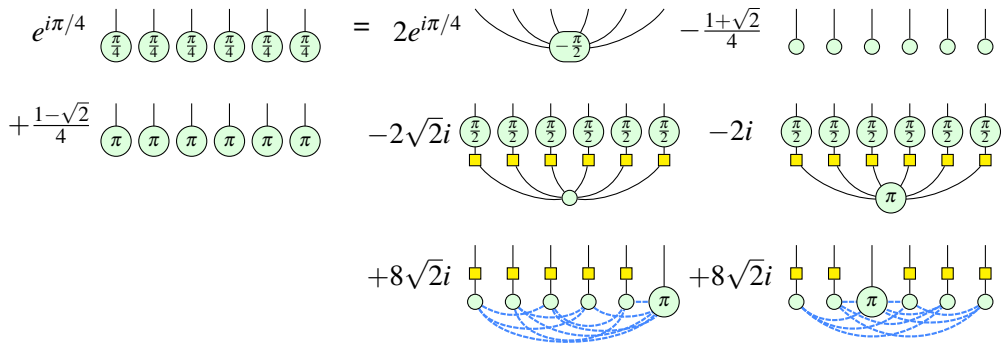


Figure 2: The BSS decomposition [8], relating a set of 6 T-gates to a sum of 7 Clifford terms, expressed in ZX-calculus terms as per [22].

Furthermore, as explored in [22], at every stage of applying the decomposition to reduce the T-count

(number of T-gates), the rewrite rules of ZX-calculus may be applied in order to, where possible, remove any T-gates that are now liable for removal. Given this extra step, the α values may be seen to represent, fairly naïvely, an estimate of the *upper-bound* of the number of terms that a Clifford+T circuit will decompose to.

2.2 Classical simulation

A very prominent task in the field is that of classically simulating quantum circuits, in order to, among other things, verify quantum algorithms and hardware. More specifically, this can be divided into *strong* and *weak* simulation, whereby the former is to deduce arbitrary (potentially marginal) probabilities associated with specific measurement outcomes, while the latter is to produce sample outputs of a circuit in accordance with its output probability distribution.

As the name suggests, being able to perform strong simulation implies we can do weak simulation. However, this relies crucially on having access to marginal probabilities. Suppose one wishes to sample a bitstring $(a_1,\ldots,a_n)$ from the probability distribution associated with measuring the output of a quantum circuit. One can first compute the marginal probability $P(x_1 = 0)$, then set the first bit of the output $a_1 := 0$ with probability $P(x_1 = 0)$ and otherwise set $a_1 := 1$. We can compute the marginal probability $P(x_1 = a_1, x_2 = 0)$ and set the next bit $a_2 := 0$ with probability $P(x_1 = a_1, x_2 = 0)/P(x_1 = a_1)$ and set $a_2 := 1$ otherwise, and so on. Using the product rule, we see that the resulting bitstring will be sampled according to the distribution $P(x_1,\ldots,x_n)$.

A circuit may be strongly simulated by plugging a fixed input state into the inputs and the conjugate-transpose of the state associated with a particular measurement outcome into the outputs. The resulting closed diagram can then be reduced to a single complex number λ via the means outlined in section 2.1. The quantity $|\lambda|^2$ then corresponds to the probability associated with that particular measurement outcome computed according to the Born rule of quantum theory. One may also deduce marginal probabilities, concerning measurement outcomes where some qubit outputs $(a_1,\ldots,a_k)$ are set, while not caring about the others $(b_1,\ldots,b_m)$. Two means of denoting this in ZX-calculus (namely the '*summing*' and '*doubling*' approaches) are shown in Figure 3 [22]. As to which of these two options is more efficient varies from circuit to circuit, depending on a few factors, including the number of qubits and how much cancellation is facilitated by partially composing against its own adjoint.

Both methods for computing marginal probabilities potentially incur an exponential overhead. The 'summing' approach requires summing over 2^m distinct terms, whereas the 'doubling' approach takes advantage of properties of the Born rule and ZX-diagrams to eliminate the sum, but doubles the size of the diagram, which for a Clifford+T circuit will double its T-count.

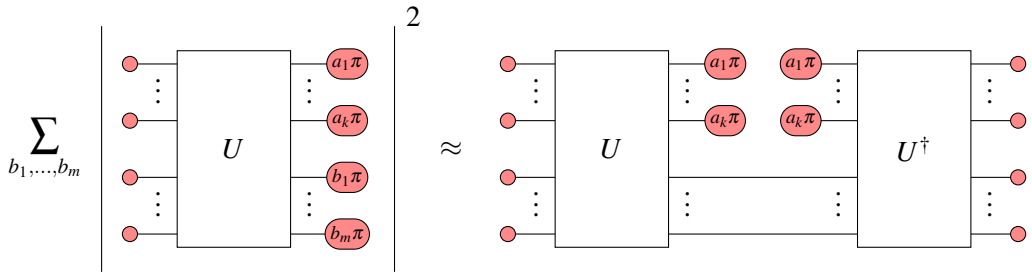


Figure 3: Two means of denoting as a ZX-diagram (up to a scalar factor) the probability that a circuit, U, will produce the qubit outcomes $a_1,\ldots,a_k$, irrespective of the qubit outcomes $b_1,\ldots,b_m$ [22]. Note that $k+m=n$, for a circuit of n total qubits.

2.3 GPU parallelism

A GPU is a specialised hardware component optimised for computing '*Single Instruction Multiple Data*' (*SIMD*) tasks very efficiently. Equipped with many thousands of parallel *threads*, a GPU is adept at performing simple functions, relying on basic arithmetic, to many units of data simultaneously [30]. Importantly, as the instructions are executed in lock-step across the threads[1], branching code (such as produced by 'If' statements) should be avoided, and as each thread is much less powerful than a CPU core, the GPU functions (or '*kernels*') should rely only upon simple arithmetic.

Note that there is a runtime overhead involved in physically transferring the data across hardware (from the CPU to the GPU and back). If the number of data elements to process exceeds the number of available parallel threads, then this data transfer may be '*pipelined*'. This means sending an initial batch of data and beginning its processing while the next batch is in transit. When applicable, this may render the data transfer time negligible and avoid running out of memory space on the GPU.

3 Methods

3.1 Parameterising ZX-calculus

Computing a k-qubit marginal probability of a quantum circuit via the summation method (left-hand side of Figure 3) involves reducing 2^k almost identical ZX-diagrams to scalars and summing the results. For circuits of relatively high T-counts, t, each such reduction may be very slow, requiring the computation of up to $2^{\alpha t}$ stabiliser terms, with α given by the decomposition efficiency.

Consequently, devising a means of reducing such similar ZX-diagrams as one, rather than reducing each independently, would be of great utility. Specifically, such sets of ZX-diagrams are structurally identical and differing only insofar as some spiders vary by a phase of $\pm\pi$. We may call such sets of ZX-diagrams '*parametrically symmetric*'. Any parametrically symmetric set may therefore be expressed as a single parameterised ZX-diagram, with every spider phase restricted to $\alpha \in \mathbb{R}$ or:

$$\Phi = (\mathfrak{p}_1 \oplus \mathfrak{p}_2 \oplus \ldots \oplus \mathfrak{p}_n)\pi + \alpha \tag{1}$$

where $\oplus$ denotes the XOR operator (addition modulo 2), $\alpha \in \mathbb{R}$, and $\mathfrak{p}_i \; \forall i = 1, 2, \ldots, n$ for $n \geq 1$ are uninstantiated boolean parameters, such that $\mathfrak{p}_i$ is unfixed but may later be instantiated to $\mathfrak{p}_i \to p_i$, where $p_i \in \mathbb{B}$. As such, while the parameter takes no fixed value, its *image* (set of possible values to which it may later be instantiated) is known: $\text{Image}(\mathfrak{p}_i) = \{0, 1\}$. Given this formalism, any parameterised phase, which necessarily takes the form of Equation 1, is such that:

$$\text{Image}(\Phi) = \{\alpha, \alpha + \pi\} \tag{2}$$

given a particular $\alpha \in \mathbb{R}$. Every phase, therefore, is either fixed and known, $\alpha \in \mathbb{R}$, or parameterised such that it may later be instantiated to α or $\alpha + \pi$.[2] Given this restriction, the parameterised phases may be said to be '*polarised*' and such ZX-diagrams said to be '*polar-parameterised*'.

Hence, any parametrically symmetric set of ZX-diagrams, expressed as a single polar-parameterised ZX-diagram, may be simplified as a single entity via the polar-parameterised versions of the rewriting rules summarised in Figure 4. Each is derivable from their non-parametric counterparts. Polar-parameterisations of additional rewriting rules, derivable from this fundamental set, are also included

[1] This is a slight oversimplification, as different '*warps*' of 32 threads may execute out of sync.

[2] Appendix A provides further justification for this restriction.

in Appendix B. Lastly, note that every phase is implicitly modulo 2π and $\text{Image}(\Phi) \subseteq \{0,\pi\}$, for instance, means the phase Φ may either be fixed at 0 or π or indeed polar-parameterised as $\Phi = (\mathfrak{p}_1 \oplus \mathfrak{p}_2 \oplus \ldots \oplus \mathfrak{p}_n)\pi$ such that $\text{Image}(\Phi) = \{0,\pi\}$.

Close inspection of these rules will reveal that, for Clifford ZX-diagrams, they represent perfect generalisations[3] of the non-parameterised rules of Figure 1. However, the same is not quite true of the broader Clifford+T gateset. Specifically, the π-commutation rule cannot be fully parameterised while maintaining generality. Particular note should be made of the fact that the π-commutation rule has been divided into three distinct polar-parameterised rules. These represent the three distinct cases in which (from Figure 1) the mapping $\Psi \to \Psi - 2\Psi\Phi$ is expressible as a mapping from one polar-parameterised phase to another. Notably, parameterising this rule for T-like phases is not possible without breaking the parametric symmetry. This is outlined in Lemma 1. As a result, constant (i.e. non-parameterised) Clifford+T ZX-diagrams are able to undergo further simplification than their equivalent polar-parameterised ZX-diagrams. This means the latter results in more stabiliser terms than the former. However, as will be seen, the difference in practice is generally very small.

Lemma 1. *Commuting a polarised phase Φ, where $\text{Image}(\Phi) = \{0,\pi\}$, through a T-like spider (parameterised or otherwise) breaks parametric symmetry.*

Proof. π-commutation with a polar-parameterised phase Φ, where $\text{Image}(\Phi) = \{0,\pi\}$, performs the following mapping to a separate phase: $\Psi \to \Psi - 2\Psi\Phi$. If Ψ is T-like (whether fixed or polar-parameterised), such as $\Psi = \frac{\pi}{4}$, then this mapping produces a phase of $\frac{\pi}{4} - \frac{\pi}{2}\Phi$. With a fractional coefficient to the polarised parameter, this phase is no longer polarised:

$$\text{Image}\left(\frac{\pi}{4} - \frac{\pi}{2}\Phi\right) = \left\{\frac{\pi}{4}, -\frac{\pi}{4}\right\} \neq \{\alpha, \alpha + \pi\}$$

for any $\alpha \in \mathbb{R}$. Hence, this gives rise to phases which are parameterised but not polarised, which, by definition, breaks the parametric symmetry. Lemma 2 in Appendix A illustrates the problems that would arise if such phases were permitted. □

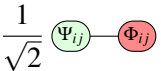
3.2 Parametric scalar expressions

Reducing a polar-parameterised Clifford+T ZX-diagram via the rules of Figure 4 and stabiliser decomposition results in a parametric scalar expression, S, of the form:

$$S = \sum_{i=1}^{m} \left[C_i \prod_{j=1}^{n_i} S_{ij} \right] \tag{3}$$

Here, m is the number of stabiliser terms, $m \leq 2^{\alpha t}$, given an initial T-count t. $C_i \in \mathbb{C}$ is then a global constant factor of term i and n_{ij} the number of parametric *subterms* within term i. Lastly, S_{ij} denotes the j^{th} subterm within the i^{th} term.

It can be shown, as in Appendix C, that every subterm may be expressed in the form:

$$S_{ij} \;=\; 1 + e^{i\Psi_{ij}} + e^{i\Phi_{ij}} - e^{i(\Psi_{ij}+\Phi_{ij})} \;=\; \frac{1}{\sqrt{2}}$$  $$\forall i, j \tag{4}$$

[3]Strictly speaking, the identity removal rule is an exception. Since this rule requires a specific phase of 0, it is not possible to generalise this to polar-parameterised phases. However, this is not a problem in practice as such instances may be pushed to one side via polar-parameterised π-commutation.

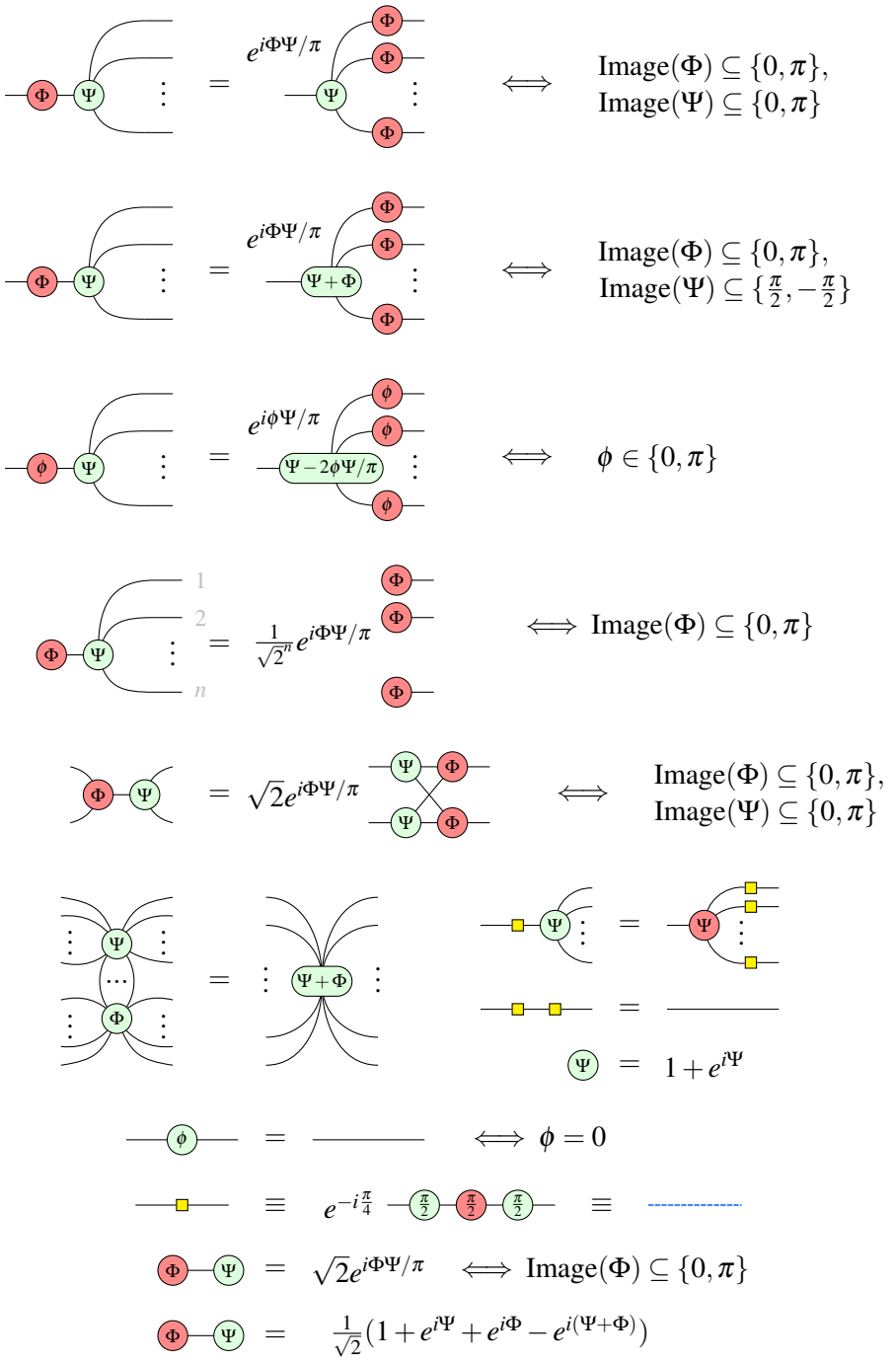


Figure 4: The complete set of polar-parameterised rewriting rules, where Ψ and Φ are parameterised or fixed phases such that Image$(\Psi) \subseteq \{\alpha, \alpha+\pi\}$ and Image$(\Phi) \subseteq \{\beta, \beta+\pi\}$, given $\alpha, \beta \in \mathbb{R}$. Meanwhile, $\phi \in \mathbb{R}$ denotes a strictly fixed (non-parametric) phase.

where:

$$\begin{aligned}\Psi_{ij} &= \alpha_{ij} + \pi \bigoplus_{\mathfrak{p}\in P_{ij}^{\psi}} \mathfrak{p} \\ \Phi_{ij} &= \beta_{ij} + \pi \bigoplus_{\mathfrak{p}\in P_{ij}^{\phi}} \mathfrak{p}\end{aligned} \tag{5}$$

with $\alpha_{ij}, \beta_{ij} \in \mathbb{R}\ \forall i, j$ and:

$$P_{ij}^{\psi}, P_{ij}^{\phi} \subseteq \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_n\} \quad \forall i, j \tag{6}$$

Here, $\{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_n\}$ denotes the full set of uninstantiated boolean parameters from the initial parameterised ZX-diagram.

To summarise so far, any parametrically symmetric set of ZX-diagrams may be expressed as a single polar-parameterised ZX-diagram and reduced to a parameterised scalar expression. This expression may then be evaluated for any particular bitstring input, $\{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_n\} \to \{p_1, p_2, \ldots, p_n\}$, where $p_i \in \mathbb{B}\ \forall i$, to deduce the scalar equivalent to having reduced the corresponding ZX-diagram. As a result, this method involves just a single instance of ZX-calculus reduction to scalar via stabiliser decomposition, followed by 2^n *evaluations* of the resulting parametric scalar to deduce all possible outcomes. This is as opposed to the traditional method, which would instead require 2^n full (and potentially very slow) reductions to scalar of independent Clifford+T ZX-diagrams.

In practice, little (if any) speedup is achieved at this stage, since a single evaluation of the (potentially very large) parameterised scalar tends to be approximately as slow as simply reducing the corresponding ZX-diagram to scalar via stabiliser decomposition. However, the benefit lies in the fact that the parametric scalar expression always adheres to a strict consistent format and requires only very simple arithmetic to evaluate. As a result, it lends itself well to GPU-parallelism.

3.3 GPU-parallelised evaluation

To prepare it for the GPU, the parametric scalar attained from reducing a polar-parametric ZX-diagram may be expressed in a very primitive data structure. Specifically, each subterm, S_{ij}, may be recorded as two bitstrings to denote which parameters among the global set are included within its P_{ij}^{ψ} and P_{ij}^{ϕ}, together with two integers a, b in the range $a, b \in [0, 7]$ to denote $\alpha_{ij} = a\frac{\pi}{4}$ and $\beta_{ij} = b\frac{\pi}{4}$. Each subterm may thus be expressed as a successive row with the following headers, where, for instance, $\mathfrak{p}_l^{\psi} = 1$ if $\mathfrak{p}_l \in P_{ij}^{\psi}$ and $\mathfrak{p}_l^{\psi} = 1$ otherwise:

$*$	$4\alpha/\pi$	$\mathfrak{p}_1^{\psi}$	$\mathfrak{p}_2^{\psi}$	$\mathfrak{p}_3^{\psi}$	$\cdots$	$4\beta/\pi$	$\mathfrak{p}_1^{\phi}$	$\mathfrak{p}_2^{\phi}$	$\mathfrak{p}_3^{\phi}$	$\cdots$

To optimise the indexing speed, it helps to artificially enforce each term, i, to share a consistent number, n_i, of subterms: $n_i = \max_i(n_i)$. This can be achieved simply by padding 'dummy' subterms into each term, with an additional bit, '*', to record whether each subterm is genuine (1) or a dummy (0). Additionally, the data should be stored in column-major order for a more efficient data access pattern, as detailed in Appendix D.

With the data now prepared into a primitive data structure, it may be evaluated with the aid of GPU parallelism. For any particular bitstring, $\{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_n\} \to \{p_1, p_2, \ldots, p_n\}$ where $p_i \in \mathbb{B}\ \forall i$, each subterm may then be evaluated in parallel on the GPU with only simple arithmetic, consistent across all threads. The process of evaluating each subterm is as follows:

1. If the row is marked as a dummy (i.e. the '$*$' column is 0) then skip to step 7. In such cases, the row can essentially be ignored and its subterm result set immediately to 1 without needing to compute any of the following calculations or steps. (This means the processing of dummy rows essentially amounts to simply doing nothing and waiting for the non-dummy threads to be processed.)
2. Substitute into each subterm expression (i.e. each row), in parallel, the chosen parameter values. This amounts to a simple bitwise multiplication (or AND operation):

1	1	p_1	p_2	p_3	$\cdots$	1	p_1	p_2	p_3	$\cdots$
				$\times$						
$*$	$4\alpha/\pi$	$\mathfrak{p}_1^\psi$	$\mathfrak{p}_2^\psi$	$\mathfrak{p}_3^\psi$	$\cdots$	$4\beta/\pi$	$\mathfrak{p}_1^\phi$	$\mathfrak{p}_2^\phi$	$\mathfrak{p}_3^\phi$	$\cdots$
				$=$						
$*$	$4\alpha/\pi$	$\mathfrak{p}_1^\psi p_1$	$\mathfrak{p}_2^\psi p_2$	$\mathfrak{p}_3^\psi p_3$	$\cdots$	$4\beta/\pi$	$\mathfrak{p}_1^\phi p_1$	$\mathfrak{p}_2^\phi p_2$	$\mathfrak{p}_3^\phi p_3$	$\cdots$

3. Calculate the XOR strings within $\Psi = (\mathfrak{p}_1^\psi p_1 \oplus \mathfrak{p}_2^\psi p_2 \oplus \ldots \oplus \mathfrak{p}_n^\psi p_n)\pi + \alpha$ and $\Phi = (\mathfrak{p}_1^\phi p_1 \oplus \mathfrak{p}_2^\phi p_2 \oplus \ldots \oplus \mathfrak{p}_n^\phi p_n)\pi + \beta$. That is, reduce $(\mathfrak{p}_1^\psi p_1 \oplus \mathfrak{p}_2^\psi p_2 \oplus \ldots \oplus \mathfrak{p}_n^\psi p_n) \to x$ and $(\mathfrak{p}_1^\phi p_1 \oplus \mathfrak{p}_2^\phi p_2 \oplus \ldots \oplus \mathfrak{p}_n^\phi p_n) \to y$, where $x, y \in \mathbb{B}$. This can be computed for each row in parallel, relying only on basic arithmetic:

$*$	$4\alpha/\pi$	$\mathfrak{p}_1^\psi p_1$	$\mathfrak{p}_2^\psi p_2$	$\mathfrak{p}_3^\psi p_3$	$\cdots$	$4\beta/\pi$	$\mathfrak{p}_1^\phi p_1$	$\mathfrak{p}_2^\phi p_2$	$\mathfrak{p}_3^\phi p_3$	$\cdots$
				$\downarrow$						
$*$	$4\alpha/\pi$		x			$4\beta/\pi$		y		

4. Given $\Psi = x\pi + \alpha \mod 2\pi$ and $\Phi = y\pi + \beta \mod 2\pi$, calculate $\frac{4\Psi}{\pi} = 4x + \frac{4\alpha}{\pi} \mod 8$ and $\frac{4\Phi}{\pi} = 4y + \frac{4\beta}{\pi} \mod 8$, such that $\frac{4\Psi}{\pi}, \frac{4\Phi}{\pi} \in \{0, 1, 2, \ldots, 7\}$. This too can be calculated for each row in parallel, using only simple operations on integers:

$*$	$4\alpha/\pi$	x	$4\beta/\pi$	y
		$\downarrow$		
$*$	$4\Psi/\pi$		$4\Phi/\pi$	

5. The next step is to calculate $e^{i\Psi}$ and $e^{i\Phi}$. However, computing complex exponentials, such as these, is computationally costly. So, to avoid this, a lookup table may be used instead, mapping $4\Psi/\pi \in \{0, 1, 2, \ldots, 7\}$ to $e^{i\Psi} \equiv A_\psi + B_\psi\sqrt{2} + i(C_\psi + D_\psi\sqrt{2})$, where this form is used to record the complex numbers as a set of simple fractional numbers.

   The same method may be used to compute $e^{i\Phi} \equiv A_\phi + B_\phi\sqrt{2} + i(C_\phi + D_\phi\sqrt{2})$ from $4\Phi/\pi$, and indeed to compute $e^{i(\Psi+\Phi)} \equiv A_{\psi+\phi} + B_{\psi+\phi}\sqrt{2} + i(C_{\psi+\phi} + D_{\psi+\phi}\sqrt{2})$ from $\frac{4\Psi}{\pi} + \frac{4\Phi}{\pi} \mod 8$. As ever, these calculations can be computed for each row in parallel:

$*$	$4\Psi/\pi$	$4\Phi/\pi$

$\downarrow$

$*$	A_ψ	B_ψ	C_ψ	D_ψ	A_ϕ	B_ϕ	C_ϕ	D_ϕ	$A_{\psi+\phi}$	$B_{\psi+\phi}$	$C_{\psi+\phi}$	$D_{\psi+\phi}$

6. Lastly, having computed $e^{i\Psi}$, $e^{i\Phi}$, and $e^{i(\Psi+\Phi)}$ for each row, ij, each final subterm result may be deduced in parallel by calculating $s_{ij} = 1 + e^{i\Psi} + e^{i\Phi} - e^{i(\Psi+\Phi)}$. The result, in each case, may be stored as four simple fractional numbers via the form $s_{ij} \equiv A + B\sqrt{2} + i(C + D\sqrt{2})$. Note that adding two such complex numbers, $e^{i\Psi} + e^{i\Phi}$, in this form amounts to adding the like terms, as outlined in Appendix E.

   Now, each row will store four simple numbers to record its subterm scalar:

$*$	A_ψ	B_ψ	C_ψ	D_ψ	A_ϕ	B_ϕ	C_ϕ	D_ϕ	$A_{\psi+\phi}$	$B_{\psi+\phi}$	$C_{\psi+\phi}$	$D_{\psi+\phi}$

$\downarrow$

$*$	$A_{1,\psi,\phi,-(\psi+\phi)}$	$B_{1,\psi,\phi,-(\psi+\phi)}$	$C_{1,\psi,\phi,-(\psi+\phi)}$	$D_{1,\psi,\phi,-(\psi+\phi)}$

7. Hiding the subscript labels here for brevity, the culmination of these steps is a matrix wherein each row now records four simple numbers which collectively denote a single subterm value:

A	B	C	D

   If the subterm was flagged as a dummy then one may jump straight to this step with $A = 1$, $B = 0$, $C = 0$, $D = 0$.

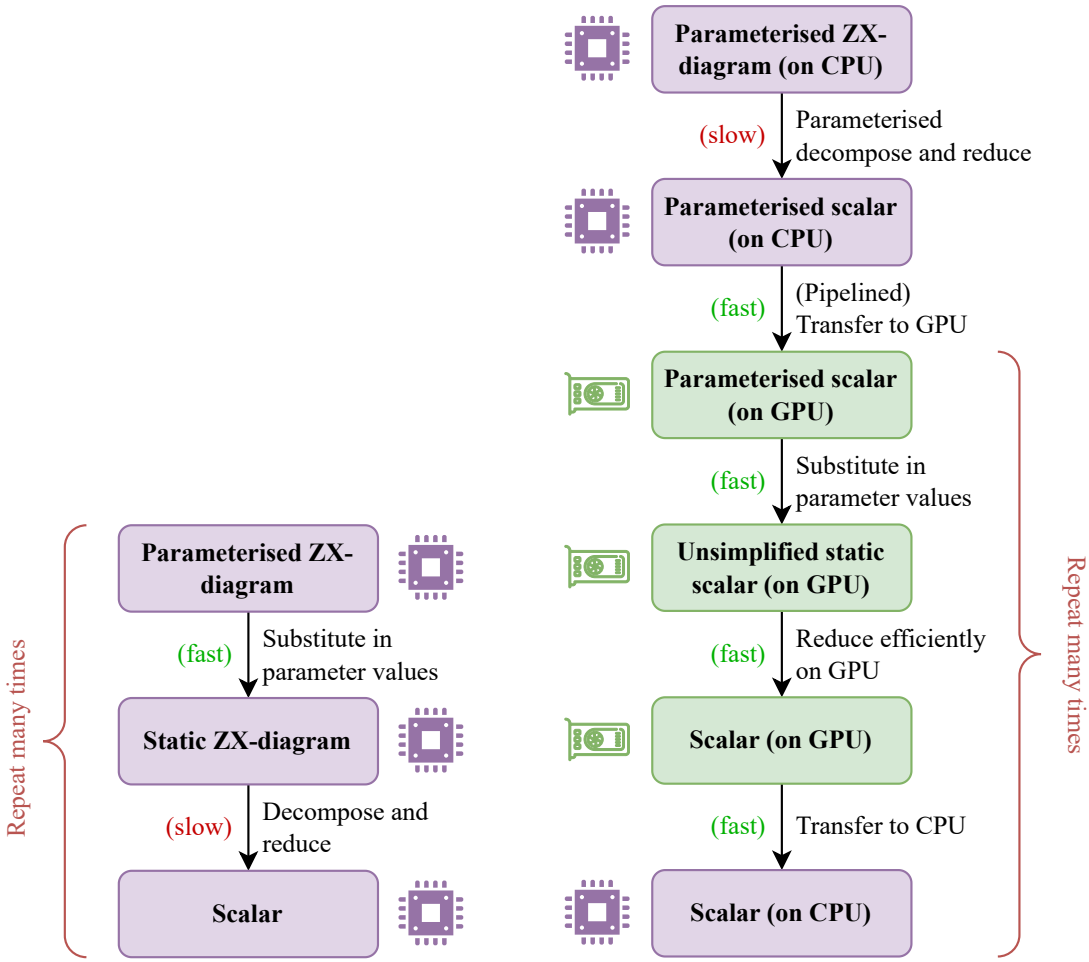
Once every subterm has been fully reduced, it remains to multiply together all the subterms within each term, and then to sum together these results, as per Equation 3. These two steps may make further use of GPU parallelism, utilising the algorithm outlined in Appendix E. This whole approach may be used for repeated strong simulation or applied directly to the left-hand side of Figure 3 for computing marginal probabilities. It may also be applied to repeated weak simulation as per Appendix F.

4 Results

An illustrative summary of this new parametric and GPU-parallelised method, as compared to the conventional approach of [22], is shown in Figure 5.

To measure the effectiveness of this approach, it was benchmarked against the conventional method of Kissinger and van de Wetering [22], for strongly simulating randomly generated Clifford+T circuits of various T-counts and parameter counts. Specifically, we compare against the Quizx [20] implementation of the conventional method, which is a Rust-based port of PyZX [19, 21], designed for speed and performance. Note that all experiments were run on relatively modest commercial hardware with the following specs: *6-core 2.69GHz Intel i5-11400H CPU and an 8GB NVIDIA GeForce GTX 1650 GPU, plus 8GB SODIMM RAM*.

For each circuit, we measure the time taken by both methods to strongly simulate various instances (i.e. evaluate its scalar for particular parameter bitstrings). The speedup factor of the new method versus



(a) The conventional CPU-based approach, introduced in [22].

(b) The parallelised GPU-based approach outlined in this chapter.

Figure 5: A comparison of the two procedures for repeated evaluation of a parameterised ZX-diagram for various sets of parameter values. The boxes show the state of the data at each step, connected by arrows indicating the processes that update these data, together with a qualitative note of the speed of each process. Purple boxes represent data being on the CPU ( ) and green boxes data on the GPU ( ).

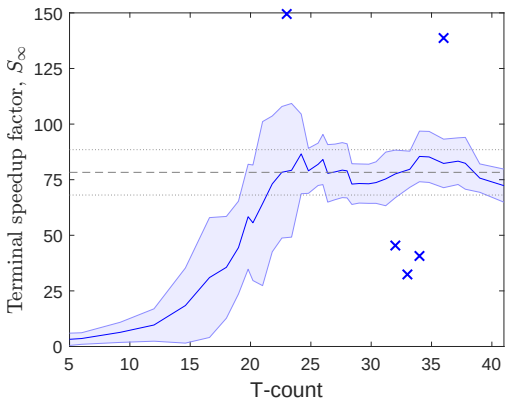
the old is then given by the ratio of these two runtimes. The speedup factor given a particular number N of evaluations is labelled S_N.

The new method carries an initial overhead runtime to reduce the original parameterised ZX-diagram to a GPU-ready parameterised scalar, though every subsequent evaluation may be computed very rapidly. Consequently, as an increasing number of evaluations N are taken, the initial overhead time becomes increasingly negligible and hence the speedup factor increases. However, as the number of parallel threads available on the GPU is finite, each experiment has a maximum '*terminal*' speedup factor, which is approached asymptotically as $N \to \infty$.

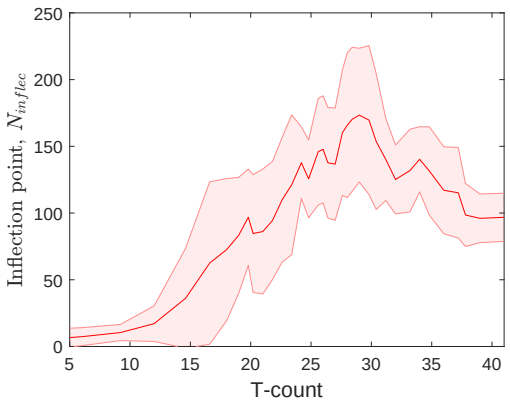
Both theoretically and experimentally, the speedup factor S_N thus follows a sigmoid curve (as pic-

tured in Appendix G) versus the number of evaluations N when plotted on a log-linear scale. With this in mind, this sigmoid curve for any particular circuit may be described by two metrics, namely:

- the **terminal speedup factor**, S_∞, which measures the theoretical maximum runtime improvement for the given parameterised ZX-diagram, and
- the **inflection point**, which measures how many evaluations are required for the speedup factor to reach half of its theoretical maximum, and which effectively quantifies the rate at which the speedup factor increases against the number of evaluations taken.



(a) The terminal speedup factors, S_∞, versus Quizx, plotted against T-count.



(b) The inflection points, N_{inflec}, plotted against T-count.

Figure 6: The measured (a) terminal speedup factors and (b) inflection points of various randomly generated parameterised ZX-diagrams versus the conventional Quizx implementation [22], plotted against the initial T-counts (after Clifford simplification). These plots show a (5-point window) moving average of the measured results, with error margins given by the standard deviation. Extreme outlier measurements have been excluded and indicated with an 'x', and Figure (a) includes a dashed grey line indicating the average S_∞ (with standard deviation error margins) across all (non-outlier) measurements of T-counts ≥ 25, where S_∞ appears to plateau.

Figure 6 shows how these metrics varied against the initial (post Clifford simplification) T-counts of the various circuits. Evidently, for trivially small T-counts, little improvement is observed. This is unsurprising as such circuits decompose into very few stabiliser terms — too few to take full advantage of GPU parallelism (especially if the number of terms is fewer than the number of available parallel threads).

Nevertheless, beyond the trivially small cases the measured terminal speedup factors appears to plateau, indicating that the full GPU capacity is being utilised. In this region (T-counts ≥ 25), we observe an average terminal speedup factor of $S_\infty = 78.3 \pm 10.2$, with rare but extreme outliers as high as 149.5 and as low as 32.4. The lower outliers are believed to represent particularly unlucky ZX-diagrams for which the problem described in Lemma 1 was more prominent. In other words, in very unfortunate (though rare) cases, non-parametric Clifford+T ZX-diagrams were able to undergo much more simplification and produce significantly fewer stabiliser terms than their parametric counterparts. The very high outliers, meanwhile, are likely due to cases which simplified unusually neatly, producing very few subterms per term, easing the computational cost.

Evidently, therefore, this method is generally very effective at speeding up the problem of strong simulation by utilising GPU hardware, particularly when a larger number of evaluations (i.e. simulating

more elements in a parametrically symmetric set of ZX-diagrams), as is very common in weak simulation tasks, where the number of evaluations is essentially exponential against the qubit count. From the inflection point measurements of Figure 6b, it is clear that hundreds of measurements is sufficient to reach at least half of the full potential speedup in each case, with thousands then being sufficient to surpass 90% of the terminal speedup factor. This means in practical situations speedup factors very near the $S_\infty = 78.3 \pm 10.2$ mark are indeed viable.

All the relevant code is available at `https://github.com/mjsutcliffe99/ParamZX` [32].

5 Conclusions and Future Directions

There are many applications in ZX-calculus where it is useful to compute (and often sum) many instances of the reduced scalar of a particular circuit, for various boolean input/output bitstrings. This paper highlights the redundancy inherent in such cases by demonstrating how the quantum circuit reduction strategies of ZX-calculus can be parameterised, with appropriate considerations, to allow circuits to be reduced while maintaining arbitrary boolean inputs/outputs. In effect this means that instead of reducing a given circuit n times to compute its scalar for n different input/output bitstrings, one can instead reduce it just once (while maintaining generality), and efficiently *evaluate* the resulting parameterised expression n times using GPU hardware. Ultimately, it was shown that, applied to classical simulation, this led to an average speedup factor of 78 ± 10.2.

While not implemented within the scope of this paper, there are a number of small techniques that could be employed to further optimise the method outlined here. For instance, after a given circuit has been reduced to a parameterised scalar, there may often be many simplifications that could be applied to minimise the number of subterms involved. For example, node-type subterms that share the same set of parameters but whose constant terms differ by π may be cancelled pairwise in place of a constant, such as $(1+e^{i\pi(\frac{2}{4}+(a\oplus b))})(1+e^{i\pi(\frac{6}{4}+(a\oplus b))}) = (1+e^{i\pi\frac{2}{4}})(1+e^{i\pi\frac{6}{4}}) = (1+i)(1-i) = 2$. A special case of this is in cancelling such pairs whose constants are 0 and π respectively, as this reduces overall to 0, hence reducing the entire scalar term to 0, negating any need to compute it further. This would increase the initial overhead time, and by extension the inflection point, but would reduce the number of subterms to compute and hence improve the terminal speedup factor.

Furthermore, this paper primarily focused on applying the method outlined within to the use-case of classical simulation via the summing and/or doubling approaches. However, the method is a very general one, applicable to many related applications such as breaking simulation tasks into disjoint components by vertex cutting [10, 33, 35] and performing weak simulation via the metropolis method described in [7].

References

[1] Scott Aaronson & Daniel Gottesman (2004): *Improved Simulation of Stabilizer Circuits*. *CoRR* quant-ph/0406196, doi:10.1103/PhysRevA.70.052328.

[2] Wira Azmoon Ahmad (2024): *Efficient Heuristics for Classical Simulation of Quantum Circuits Using ZX-Calculus*. Master's thesis, University of Oxford. Available at `https://www.cs.ox.ac.uk/people/aleks.kissinger/theses/ahmad-thesis.pdf`.

[3] Wira Azmoon Ahmad & Matthew Sutcliffe (2024): *Dynamic T-decomposition for classical simulation of quantum circuits*. *arXiv preprint arXiv:2412.17182*, doi:10.48550/arXiv.2412.17182. arXiv:2412.17182.

[4] Miriam Backens (2014): *The ZX-calculus is complete for stabilizer quantum mechanics*. New Journal of Physics 16(9), p. 093021, doi:10.1088/1367-2630/16/9/093021.

[5] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Fast and Effective Techniques for T-Count Reduction via Spider Nest Identities*. In Steven T. Flammia, editor: *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, Leibniz International Proceedings in Informatics (LIPIcs) 158, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 11:1–11:23, doi:10.4230/LIPIcs.TQC.2020.11.

[6] Agustín Borgna, Simon Perdrix & Benoît Valiron (2021): *Hybrid quantum-classical circuit simplification with the ZX-calculus*. In Hakjoo Oh, editor: Programming Languages and Systems, Springer International Publishing, Cham, pp. 121–139, doi:10.1007/978-3-030-89051-3_8.

[7] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset & Mark Howard (2019): *Simulation of quantum circuits by low-rank stabilizer decompositions*. Quantum 3, p. 181, doi:10.22331/q-2019-09-02-181.

[8] Sergey Bravyi, Graeme Smith & John A. Smolin (2016): *Trading classical and quantum computational resources*. Physical Review X 6(2), p. 021043, doi:10.1103/PhysRevX.6.021043.

[9] Tristan Cam & Simon Martiel (2023): *Speeding up quantum circuits simulation using ZX-Calculus*. arXiv preprint arXiv:2305.02669, doi:10.48550/arXiv.2305.02669.

[10] Julien Codsi (2022): *Cutting-Edge Graphical Stabiliser Decompositions for Classical Simulation of Quantum Circuits*. Master's thesis, University of Oxford.

[11] Julien Codsi & John van de Wetering (2022): *Classically Simulating Quantum Supremacy IQP Circuits trough a Random Graph Approach*. arXiv preprint arXiv:2212.08609, doi:10.48550/arXiv.2212.08609.

[12] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. New Journal of Physics 13, p. 043016, doi:10.1088/1367-2630/13/4/043016.

[13] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes*. Cambridge University Press, doi:10.1017/9781316219317.

[14] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons & Seyon Sivarajah (2020): *Phase Gadget Synthesis for Shallow Circuits*. In Bob Coecke & Matthew Leifer, editors: Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019, Electronic Proceedings in Theoretical Computer Science 318, Open Publishing Association, pp. 213–228, doi:10.4204/EPTCS.318.13.

[15] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. Quantum 4, p. 279, doi:10.22331/q-2020-06-04-279.

[16] Stefano Gogioso & Richie Yeung (2023): *Annealing Optimisation of Mixed ZX Phase Circuits*. In Stefano Gogioso & Matty Hoban, editors: Proceedings 19th International Conference on Quantum Physics and Logic, Wolfson College, Oxford, UK, 27 June - 1 July 2022, Electronic Proceedings in Theoretical Computer Science 394, Open Publishing Association, pp. 415–431, doi:10.4204/EPTCS.394.20.

[17] Calum Holker (2023): *Causal flow preserving optimisation of quantum circuits in the ZX-calculus*. arXiv preprint arXiv:2312.02793, doi:10.48550/arXiv.2312.02793.

[18] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18, ACM, New York, NY, USA, pp. 559–568, doi:10.1145/3209108.3209131.

[19] Aleks Kissinger & John van de Wetering: *PyZX*. Available at `https://github.com/Quantomatic/pyzx`.

[20] Aleks Kissinger & John van de Wetering: *Quizx*. Available at `https://github.com/Quantomatic/quizx`.

[21] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. In Bob Coecke & Matthew Leifer, editors: Proceedings 16th International Conference on Quantum Physics

and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019, *Electronic Proceedings in Theoretical Computer Science* 318, Open Publishing Association, pp. 229–241, doi:10.4204/EPTCS.318.14.

[22] Aleks Kissinger & John van de Wetering (2022): *Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions*. *Quantum Science and Technology* 7(4), p. 044001, doi:10.1088/2058-9565/ac5d20.

[23] Aleks Kissinger, John van de Wetering & Renaud Vilmart (2022): *Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions*. In François Le Gall & Tomoyuki Morimae, editors: *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 232, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 5:1–5:13, doi:10.4230/LIPIcs.TQC.2022.5.

[24] Mark Koch, Richie Yeung & Quanlong Wang (2023): *Speedy Contraction of ZX Diagrams with Triangles via Stabiliser Decompositions*. *arXiv preprint arXiv:2307.01803*, doi:10.48550/arXiv.2307.01803.

[25] Alexander Koziell-Pipe, Richie Yeung & Matthew Sutcliffe (2024): *Towards Faster Quantum Circuit Simulation Using Graph Decompositions, GNNs and Reinforcement Learning*. In: *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*. Available at `https://openreview.net/forum?id=54060pbCKY`.

[26] Tommy McElvanney & Miriam Backens (2023): *Flow-preserving ZX-calculus Rewrite Rules for Optimisation and Obfuscation*. In Shane Mansfield, Benoit Valîron & Vladimir Zamdzhiev, editors: *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, *Electronic Proceedings in Theoretical Computer Science* 384, Open Publishing Association, pp. 203–219, doi:10.4204/EPTCS.384.12.

[27] Maximilian Nägele & Florian Marquardt (2023): *Optimizing ZX-Diagrams with Deep Reinforcement Learning*. *arXiv preprint arXiv:2311.18588*, doi:10.1088/2632-2153/ad76f7.

[28] Kang Feng Ng & Quanlong Wang (2018): *Completeness of the ZX-calculus for Pure Qubit Clifford+T Quantum Mechanics*. *arXiv:1801.07993*, doi:10.48550/arXiv.1801.07993.

[29] Michael A. Nielsen & Isaac L. Chuang (2010): *Quantum Computation and Quantum Information*. Cambridge University Press, doi:10.1017/CBO9780511976667.

[30] NVIDIA, Péter Vingelmann & Frank H.P. Fitzek (2020): *CUDA, release: 10.2.89*. Available at `https://developer.nvidia.com/cuda-toolkit`.

[31] Hammam Qassim, Hakop Pashayan & David Gosset (2021): *Improved upper bounds on the stabilizer rank of magic states*. *Quantum* 5, p. 606, doi:10.22331/q-2021-12-20-606.

[32] Matthew Sutcliffe: *ParamZX*. Available at `https://github.com/mjsutcliffe99/ParamZX`.

[33] Matthew Sutcliffe (2024): *Smarter k-Partitioning of ZX-Diagrams for Improved Quantum Circuit Simulation*. *arXiv preprint arXiv:2409.00828*, doi:10.48550/arXiv.2409.00828. arXiv:2409.00828.

[34] Matthew Sutcliffe (2025): *Classically simulating quantum circuits with ZX-calculus*. *[Preprint]*.

[35] Matthew Sutcliffe (2025): *Novel Methods for Classical Simulation of Quantum Circuits via ZX-Calculus*. Ph.D. thesis, University of Oxford, Oxford, United Kingdom.

[36] Matthew Sutcliffe & Aleks Kissinger (2024): *Procedurally Optimised ZX-Diagram Cutting for Efficient T-Decomposition in Classical Simulation*. *Electronic Proceedings in Theoretical Computer Science* 406, p. 63–78, doi:10.4204/eptcs.406.3.

[37] Quanlong Wang (2018): *Completeness of the ZX-calculus*. Ph.D. thesis, Wolfson College, University of Oxford. Available at `https://arxiv.org/pdf/2209.14894.pdf`.

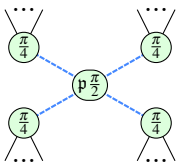
[38] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. doi:10.48550/arXiv.2012.13966.

[39] Robert Wille, Lukas Burgholzer, Stefan Hillmich, Thomas Grurl, Alexander Ploier & Tom Peham (2022): *The Basis of Design Tools for Quantum Computing: Arrays, Decision Diagrams, Tensor Networks, and ZX-Calculus*. In: *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, Association for Computing Machinery, New York, NY, USA, p. 1367–1370, doi:10.1145/3489517.3530627.

A Preserving parametric symmetry

Lemma 2. *Reducing a parametrically symmetric set of ZX-diagrams as a single polar-parameterised ZX-diagram requires every phase at every step to be either fixed or polarised.*

Proof. As a proof by example, consider the following parameterised ZX-diagram containing a non-polarised phase $\mathfrak{p}\frac{\pi}{2}$, where $\mathfrak{p}$ is an uninstantiated boolean parameter:



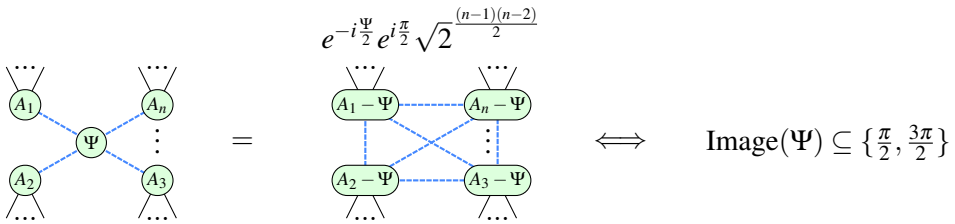
Here, the central parameterised phase is not polarised:

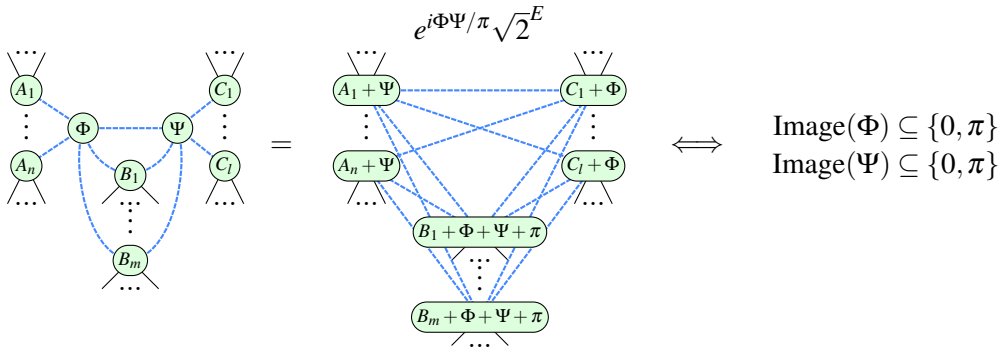
$$\text{Image}\left(\mathfrak{p}\frac{\pi}{2}\right) = \left\{0, \frac{\pi}{2}\right\} \neq \{\alpha, \alpha+\pi\}$$

for any $\alpha \in \mathbb{R}$. Consequently, in this case it is ambiguous whether the local complementation rule applies, as it would do so when $\mathfrak{p} \to 1$ but not when $\mathfrak{p} \to 0$. In other words, the two possibilities of $\mathfrak{p} \to 0$ and $\mathfrak{p} \to 1$ would lead to two diverging branches, meaning the parametrically symmetric pair would no longer be expressible as a single parameterised ZX-diagram. □

B Additional polar-parameterised rewrite rules

In additional to the generalised rules given in Figure 4, some higher-level rules (derivable from the basic set) may also be generalised for polar-parameterised ZX-diagrams. Specifically, these are parametric versions of the *local complementation* and *pivoting* rules [38]:





The derivations of these parametric rules, including those of Figure 4, follow from their non-parametric counterparts and are left as an exercise for the reader.

C Equivalence of subterm types

From the parameterised rewriting rules and scalar relations presented in Figure 4 and Appendix B, one may observe that there are in fact four different types of parameterised subterms that may arise. These are summarised in table 1.

Name	Form	Origin(s)
Node	$(1+e^{i\Psi})$	
Phase-pair	$\left(1+e^{i\Psi}+e^{i\Phi}-e^{i(\Psi+\Phi)}\right)$	
Half-π	$e^{i\Psi/2}$	Appendix B
π-pair	$e^{i\Psi\Phi/\pi}$	Figure 4 & Appendix B

Table 1: The different types of subterms that may arise from reducing parameterised ZX-diagrams, where Ψ and Φ are polar-parameterised phases obeying the form of Equation 1.

In fact, it can be shown that these may all reduce to a single unique subterm type, namely that labelled '*phase-pair*'. Lemmas 3 to 5 formalise this observation.

Lemma 3. *Node-type subterms can be reduced to phase-pair subterms:*

$$\left(1+e^{i\Psi}\right) = C\left(1+e^{i\Psi'}+e^{i\Phi}-e^{i(\Psi'+\Phi)}\right) \tag{7}$$

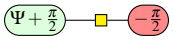
where $\Psi' = \Psi + \frac{\pi}{2}$ *and* $C = \frac{\sqrt{2}}{4}(1-i)$.

Proof.

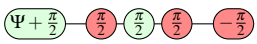
$$\left(1+e^{i\Psi}\right) = \tag{8}$$

$=$

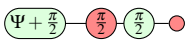
$=$



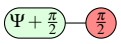
$= e^{-i\frac{\pi}{4}}$



$= e^{-i\frac{\pi}{4}}$



$= \frac{1}{\sqrt{2}}e^{-i\frac{\pi}{4}}$



$= \frac{1}{\sqrt{2}}e^{-i\frac{\pi}{4}} \cdot \frac{1}{\sqrt{2}}\left(1+e^{i(\Psi+\frac{\pi}{2})}+e^{i\Phi}-e^{i(\Psi+\frac{\pi}{2}+\Phi)}\right)$

$= \frac{\sqrt{2}}{4}(1-i)\left(1+e^{i\Psi'}+e^{i\Phi}-e^{i(\Psi'+\Phi)}\right)$

□

Lemma 4. *π-pair subterms can be reduced to phase-pair subterms:*

$$e^{i\Psi\Phi/\pi} \to C\left(1 + e^{i\Psi} + e^{i\Phi} - e^{i(\Psi+\Phi)}\right) \tag{9}$$

where $C = \frac{1}{2}$.

Proof. There are five sources from which π-pair subterms, $e^{i\Psi\Phi/\pi}$, may arise, being:

- parameterised state copy (Figure 4),
- parameterised pivoting (Appendix B),
- parameterised π-commutation (Figure 4),
- parameterised bialgebra (Figure 4), and
- parameterised special case phase-pair (Figure 4).

In each case, one may observe that $\text{Image}(\Phi) \subseteq \{0, \pi\}$. Under this restriction, the relation holds:

$$e^{i\Psi\Phi/\pi} = \frac{1}{2}\left(1 + e^{i\Psi} + e^{i\Phi} - e^{i(\Psi+\Phi)}\right) \tag{10}$$

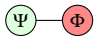
Explicitly, if $\Phi = 0$:

$$\begin{aligned} e^{0} &= \frac{1}{2}\left(1 + e^{i\Psi} + e^{0} - e^{i(\Psi+0)}\right) \\ 1 &= \frac{1}{2}(1+1) \\ 1 &= 1 \\ \therefore \text{LHS} &= \text{RHS} \end{aligned} \tag{11}$$

Likewise, if $\Phi = \pi$:

$$\begin{aligned} e^{i\Psi} &= \frac{1}{2}\left(1 + e^{i\Psi} + e^{i\pi} - e^{i(\Psi+\pi)}\right) \\ e^{i\Psi} &= \frac{1}{2}\left(1 + e^{i\Psi} + (-1) - e^{i\Psi}e^{i\pi}\right) \\ e^{i\Psi} &= \frac{1}{2}\left(1 + e^{i\Psi} - 1 + e^{i\Psi}\right) \\ e^{i\Psi} &= \frac{1}{2}\left(2e^{i\Psi}\right) \\ e^{i\Psi} &= e^{i\Psi} \\ \therefore \text{LHS} &= \text{RHS} \end{aligned} \tag{12}$$

Hence, graphically:



$$= \sqrt{2}e^{i\Psi\Phi/\pi} \tag{13}$$

provided $\text{Image}(\Phi) \subseteq \{0, \pi\}$. □

Lemma 5. *Half-π subterms can be reduced to phase-pair subterms:*

$$e^{i\Psi/2} \to C\left(1 + e^{i\Psi'} + e^{i\Phi} - e^{i(\Psi'+\Phi)}\right) \tag{14}$$

where $\Psi' = -\Psi + \frac{\pi}{2}$ *and* $C = \frac{1}{2}$.

Proof. Half-π subterms arise from instances of parameterised local complementation (Appendix B), from which it may be observed that $\text{Image}(\Psi) \subseteq \{\frac{\pi}{2}, \frac{3\pi}{2}\}$.

These terms may be slightly rewritten with a change of variable:

$$e^{-i\frac{\Psi}{2}} e^{i\frac{\pi}{2}} = e^{\frac{i}{2}(-\Psi+\frac{\pi}{2})} e^{\frac{i\pi}{4}} = e^{\frac{i}{2}\Psi'} e^{\frac{i\pi}{4}} \tag{15}$$

where $\Psi' = -\Psi + \frac{\pi}{2}$ such that $\text{Image}(\Psi') \subseteq \{0, \pi\}$. Furthermore:

$$e^{i\Psi'/2} \equiv e^{i\Psi'\Phi/\pi} \tag{16}$$

where $\Phi = \frac{\pi}{2}$. Hence, the half-π subterm type is a special case of the π-pair type, which was shown in Lemma 4 to be reducible to the phase-pair type. □

Given this, all such parameterised scalar expressions are expressible in a consistent format, described by Equation 4.

D Coalescing the data

Consider an example parametric scalar, in matrix form, consisting of 4 parameters $(\mathfrak{p}_1, \mathfrak{p}_2, \mathfrak{p}_3, \mathfrak{p}_4)$ and 2 terms:

$*$	$4\alpha/\pi$	$\mathfrak{p}_1^\psi$	$\mathfrak{p}_2^\psi$	$\mathfrak{p}_3^\psi$	$\mathfrak{p}_4^\psi$	$4\beta/\pi$	$\mathfrak{p}_1^\phi$	$\mathfrak{p}_2^\phi$	$\mathfrak{p}_3^\phi$	$\mathfrak{p}_4^\phi$
1	2	1	1	0	0	7	1	1	1	0
1	4	0	1	1	1	3	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0
1	6	1	0	0	0	3	1	1	1	1
1	5	1	1	1	1	1	1	0	1	0
1	2	0	0	0	1	4	1	1	0	0

While for practical purposes this data is treated as two-dimensional, as far as the memory is concerned it is in fact, necessarily stored linearly. Conventionally, this would be stored, linearised, in row-major order (storing the cells of the first row, followed by those of the second row and so on). This is appropriate when the data is to be processed row-by-row on the CPU (whose processing pattern is inherently sequential), as each subsequent element of data to be processed is stored immediately after the previous, ensuring convenient (quicker) access.

However, if instead, as in this case, each row of this data is to be processed in parallel on the GPU, then this memory arrangement proves to be suboptimal. In this scenario, it is preferential to store the data in column-major order (meaning all the cells of the first column are stored one after the other, followed then by the cells of the second column and so on). A justification for this follows:

Lemma 6. *To better optimise for GPU processing, the two-dimensional subterm data should be stored in memory in column-major order (rather than the more typical row-major order that 2D arrays tend to be stored as).*

Proof. When the rows are to be processed in parallel, they will process their respective *dummy flags* (their data of the first column) at the same time, and then their respective *constant* factors (the second column) at the same time, and so on. Consequently, storing all the *dummy flags* (the first column) together in

memory means that they can all be retrieved very efficiently, with many being moved in one transaction (as opposed to individually locating and sending each row's dummy flag one by one). For this reason, the two-dimensional data of the node-type subterms (and likewise for the 2D arrays of other subterm types) is better stored in column-major order, for a far more efficient ('*coalesced*') memory access pattern. □

E Parallelised summation algorithm

Given an array of n numbers, the best method for summing (or alternatively multiplying) all its elements - via sequential (i.e. single-core CPU) computation - is to simply iterate linearly through the whole array while maintaining a total tally. This gives a runtime complexity of $O(n)$.

Alternatively, by instead processing the data on the GPU, one may compute such a calculation while taking advantage of parallelism. Consider, for example, the following 10-element array:

$$[\,0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\,]$$

One can allocate a single GPU thread for every adjacent pair of elements. For the general case of an n-element array, this means allocating $n/2$ threads, which in this example case means 5 threads, divided as follows:

$$[\ \mathbf{0},\ 1,\ |\ \mathbf{2},\ 3,\ |\ \mathbf{4},\ 5,\ |\ \mathbf{6},\ 7,\ |\ \mathbf{8},\ 9\ ]$$

Each thread, running in parallel, may then sum the two elements under its consideration and overwrite its left element with the result, like so:

$$[\ \mathbf{1},\ 1,\ |\ \mathbf{5},\ 3,\ |\ \mathbf{9},\ 5,\ |\ \mathbf{13},\ 7,\ |\ \mathbf{17},\ 9\ ]$$

The right element of each pair can hereafter be ignored and so is written in grey here. The process can then repeat - this time with $n/4$ threads being allocated to consider, respectively, adjacent groups of 4 elements, like so:

$$[\ \mathbf{1},\ 1,\ 5,\ 3,\ |\ \mathbf{9},\ 5,\ 13,\ 7,\ |\ \mathbf{17},\ 9\ ]$$

Now, for each thread, the leftmost element will be incremented by the value contained 2 elements to its right (unless that index exceeds the length of the array):

$$[\ \mathbf{6},\ 1,\ 5,\ 3,\ |\ \mathbf{22},\ 5,\ 13,\ 7,\ |\ \mathbf{17},\ 9\ ]$$

This process then repeats iterative, halving the number of threads needed each time and doubling, for each thread, the number of elements considered and the gap between its relevant elements. This procedure follows as such:

$$[\ \mathbf{6},\ 1,\ 5,\ 3,\ 22,\ 5,\ 13,\ 7,\ |\ \mathbf{17},\ 9\ ]$$

$$[\ \mathbf{28},\ 1,\ 5,\ 3,\ 22,\ 5,\ 13,\ 7,\ |\ \mathbf{17},\ 9\ ]$$

$$[\ \mathbf{28},\ 1,\ 5,\ 3,\ 22,\ 5,\ 13,\ 7,\ 17,\ 9\ ]$$

$$[\ \mathbf{45},\ 1,\ 5,\ 3,\ 22,\ 5,\ 13,\ 7,\ 17,\ 9\ ]$$

Once the segment size exceeds the length of the array, the procedure is complete and the final result (in this case 45) - being the sum of all the elements in the original array - is stored in the first element.

Theoretically, this method computes in as few as $log_2 n$ iterations, rather than n. Realistically, however, as the number of threads on a GPU is finite, for sufficiently large arrays the theoretically parallelised steps won't all occur simultaneously but rather in locally parallelised batches. Nevertheless, even as $n \to \infty$ this method requires significantly fewer iterations than the naïve approach. And lastly, note that the method also works likewise for multiplication as well as summation.

The pseudocode for this method is provided in algorithm 1, where the *PSum* procedure computes in parallel on n threads - with each given a successive thread index provided by *GetThreadIndex()*.

This algorithm (or variations thereof) is used as described in the paper, for both multiplying the subterms within each term as well as summing all the terms in the overall scalar. The only major difference to note is that instead of the simple number summing calculation on line 7, the complex numbers are summed (and later multiplied) according to the relations described in Lemmas 7 and 8.

Lemma 7. *Two complex numbers, $\psi, \phi \in \mathbb{C}$, each expressed via four simple real values, $A_x, B_x, C_x, D_x \in \mathbb{R}\ \forall x$, in the form:*

$$\begin{aligned} \psi &= A_\psi + B_\psi\sqrt{2} + i\left(C_\psi + D_\psi\sqrt{2}\right) \\ \phi &= A_\phi + B_\phi\sqrt{2} + i\left(C_\phi + D_\phi\sqrt{2}\right) \end{aligned} \tag{17}$$

may be summed together, $\vartheta = \psi + \phi$, using only simple real arithmetic:

$$\vartheta = A_\vartheta + B_\vartheta\sqrt{2} + i\left(C_\vartheta + D_\vartheta\sqrt{2}\right) \tag{18}$$

where here:

$$\begin{aligned} A_\vartheta &= A_\psi + A_\phi \\ B_\vartheta &= B_\psi + B_\phi \\ C_\vartheta &= C_\psi + C_\phi \\ D_\vartheta &= D_\psi + D_\phi \end{aligned} \tag{19}$$

Lemma 8. *Two complex numbers, $\psi, \phi \in \mathbb{C}$, each expressed via four simple real values, $A_x, B_x, C_x, D_x \in \mathbb{R}\ \forall x$, in the form:*

$$\begin{aligned} \psi &= A_\psi + B_\psi\sqrt{2} + i\left(C_\psi + D_\psi\sqrt{2}\right) \\ \phi &= A_\phi + B_\phi\sqrt{2} + i\left(C_\phi + D_\phi\sqrt{2}\right) \end{aligned} \tag{20}$$

may be multiplied together, $\vartheta = \psi \times \phi$, using only simple real arithmetic:

$$\vartheta = A_\vartheta + B_\vartheta\sqrt{2} + i\left(C_\vartheta + D_\vartheta\sqrt{2}\right) \tag{21}$$

where:

$$\begin{aligned} A_\vartheta &= A_\psi A_\phi + 2B_\psi B_\phi - C_\psi C_\phi - 2D_\psi D_\phi \\ B_\vartheta &= A_\psi B_\phi + B_\psi A_\phi - C_\psi D_\phi - D_\psi C_\phi \\ C_\vartheta &= A_\psi C_\phi + 2B_\psi D_\phi + C_\psi A_\phi + 2D_\psi B_\phi \\ D_\vartheta &= A_\psi D_\phi + B_\psi C_\phi + C_\psi B_\phi + D_\psi A_\phi \end{aligned} \tag{22}$$

Algorithm 1 The GPU-parallelised summation algorithm

Initialise and populate $ARR[N_{ELEMS}]$ ▷ The array whose elements are to be summed

procedure PSUM($split, gap$)<< n >> ▷ This is a GPU kernel, executing on n threads
    $i \leftarrow$ GETTHREADINDX() ▷ This function returns the unique thread index
    $elem \leftarrow i \times split$
    if $elem + gap < N_{ELEMS} - 1$ **then**
        $ARR[elem] \leftarrow ARR[elem] + ARR[elem + gap]$
    end if
end procedure

$split \leftarrow 2$
$gap \leftarrow 1$
while $gap < N_{ELEMS}$ **do**
    $n_{chunks} \leftarrow \lceil N_{ELEMS}/split \rceil$
    PSUM($split, gap$)<< n_{chunks} >>
    $split \leftarrow split \times 2$
    $gap \leftarrow gap \times 2$
end while

F Repeated weak simulation

In addition to the use-case of computing marginal probabilities via the summing method (left-hand side of Figure 3), this method may also be applied to repeated strong simulation, and indeed also repeated *weak* simulation. Its relevance to the former is perhaps obvious, though to the latter may need further explanation.

In this situation, each ($1 \leq k \leq n$) marginal probability computation of an n-qubit circuit, U, can be computed parametrically (see right-hand side of Figure 3). With this approach, at each iteration (i.e. in incrementing k) of computing the next marginal probability, one may deduce the next bit in the final bitstring, for any number N of independent bitstrings. Hence, one will ultimately produce N such bitstrings denoting N independent samples of weak simulation, while only ever reducing n doubled ZX-diagram, given n qubits.

This is summarised in algorithm 2, where:

$$P(\mathfrak{a}_1 \cdots \mathfrak{a}_k) \xleftarrow{r} \langle 0 \cdots 0 | U^\dagger (|\mathfrak{a}_1 \cdots \mathfrak{a}_k\rangle \langle \mathfrak{a}_1 \cdots \mathfrak{a}_k| \otimes I_{n-k}) U |0 \cdots 0\rangle \tag{23}$$

denotes the reduction of the doubled marginal probability ZX-diagram (right-hand side of Figure 3) to a parameterised scalar expression, $P(\mathfrak{a}_1 \cdots \mathfrak{a}_k)$, denoting its outcome probability. $P(B)$ then denotes a GPU-based evaluation of this scalar expression for the bitstring $B = a_1 \cdots a_k$, where $a_i \in \mathbb{B}\ \forall i$. Meanwhile, $X||Y$ denotes the concatenation of X and Y, such that $[a,b,c]||d = [a,b,c,d]$, and `Rand()` returns a random floating point number in the range $[0,1)$.

There are further optimisations that may be made to this algorithm, with some particular redundancy among the first few iterations of k, where identical evaluations are inevitable. But such considerations are perhaps superfluous.

Algorithm 2 Algorithm for efficient repeated weak simulation

Input: A quantum circuit U with n_{qubits} qubits, and N_{evals}

$B_i = [\,]\ \forall i \in \{1,\ldots,n_{qubits}\}$ ▷ Initialise empty lists

for $k = 1$ to n_{qubits} **do**

  $P(\mathfrak{a}_1 \cdots \mathfrak{a}_k) \xleftarrow{r} \langle 0\cdots 0|\, U^\dagger (|\mathfrak{a}_1 \cdots \mathfrak{a}_k\rangle \langle \mathfrak{a}_1 \cdots \mathfrak{a}_k| \otimes I_{n-k}) U\, |0\cdots 0\rangle$

  for $i = 1$ to N_{evals} **do**

    $P_{B_i||0} = P(B_i||0)$ ▷ GPU-based evaluation

    $b \leftarrow 0$ **if** `Rand()` $< P_{B_i||0}$ **else** 1

    $B_i = B_i||b$

  end for

end for

Output: $B_i\ \forall i$

G Sigmoid curves

The relationship between the speedup factor, S_N, and the number of evaluations, N, can be visualised as a sigmoid curve of the form:

$$S_N(N) = S_\infty \cdot \frac{N}{N_{inflec} + N} \tag{24}$$

as in Figure 7, where N_{inflec} is the inflection point: that is the value of N at which $S_N = S_\infty/2$. In fact, S_∞ may be calculated as the ratio of the evaluation times of the two methods, while N_{inflec} may be calculated as the ratio of the initialisation time against the evaluation time of the new method.

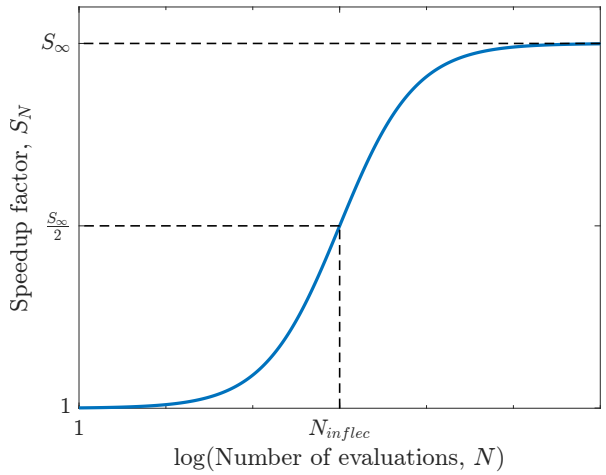


Figure 7: The relationship between the speedup factor, S_N, and the number of evaluations, N, for any particular ZX-diagram may be visualised as a sigmoid curve. Note the logarithmic x-axis.

Classical and Quantum Query Complexity of Boolean Functions under Indefinite Causal Order

Alastair A. Abbott
Univ. Grenoble Alpes, Inria, 38000 Grenoble, France
Mehdi Mhalla
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
Pierre Pocreau
Univ. Grenoble Alpes, Inria, 38000 Grenoble, France
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

Computational models typically assume that operations are applied in a fixed sequential order. In recent years several works have looked at relaxing this assumption, considering computations without any fixed causal structure and showing that such "causally indefinite" computations can provide advantages in various tasks. Recently, the quantum query complexity of Boolean functions has been used as a tool to probe their computational power in a standard complexity theoretic framework, but no separation in exact query complexity has thus-far been found. In this paper, we investigate this problem starting with the simpler and fully classical notion of deterministic query complexity of Boolean functions, and using classical-deterministic processes – which may exhibit causal indefiniteness – as a generalised computational framework. We first show that the standard polynomial and certificate lower bounds of deterministic query complexity also hold in such generalised models. Then, we formulate a Boolean function for which causal indefiniteness permits a reduction in query complexity and show that this advantage can be amplified into a polynomial separation. Finally, with the insights gained in the classical-deterministic setting, we give a Boolean function whose quantum query complexity is reduced by causally indefinite computations.

1 Introduction

Computational models, both classical and quantum, typically assume that operations are applied in a fixed, sequential "causal" order. There has been growing interest, however, in relaxing such assumptions and exploring causally indefinite ways of processing information. On the one hand, quantum mechanics allows the order in which quantum operations are applied to be controlled by other quantum systems and hence rendered indefinite, as in the quantum switch [29]. Such processes, which formally correspond to quantum supermaps [28, 29] and can be represented in the process-matrix formalism [39], have been shown to provide advantages in several tasks in quantum information [25, 10, 21, 33]. On the other hand, classical computational models exploiting closed timelike curves (CTCs) have been studied [23, 13, 3, 4], and it is known that causal indefiniteness in the process matrix framework is not a property unique to quantum processes. Indeed, logically consistent classical (even deterministic) processes can be formulated that are incompatible with any causal explanation and can, for instance, exhibit maximally noncausal correlations while remaining paradox-free [15, 19].

Sparked by this, there has been growing interest in studying more rigorously the computational power of these causally indefinite processes within a complexity theoretic framework. One approach has been to connect quantum supermaps and their classical-deterministic counterpart, process functions [18], to

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 270–300, doi:10.4204/EPTCS.426.11

computation with linear CTCs in order to study their computational complexity [12, 20]. Quantum supermaps and process functions, however, are objects that most naturally describe how to compose various "black-box" operations (be it in a causally ordered, or indefinite, way). It is perhaps most natural then to study their computational power within a query complexity framework. While various works have looked to quantify the number of queries needed to determine relative properties of multiple quantum black-box operations provided as input [10, 42] or to compose them into another quantum operation as a function of the input operations [35], these works fall outside the traditional framework of query complexity.

In a prior work [5], we extended the standard notion of quantum query complexity of Boolean functions [24, 8] to general, potentially causally indefinite, quantum supermaps, showing that causal indefiniteness can indeed provide some advantages in this setting, with a reduction in error-probability obtained for computing some Boolean functions with two queries. However, it remained an open question whether asymptotic separations, or advantages in the exact (rather than bounded-error) setting, can be obtained.

In this paper, we propose to first take a step back and study the computational power of causal indefiniteness in classical processes using the query complexity model. This simpler setting, in addition to being interesting in its own rights, can provide useful insights for the quantum case. Working within the framework of process functions, we introduce a standard notion of deterministic query complexity of Boolean functions which generalises the well-known notion of decision tree complexity [24] to causally indefinite deterministic computations.

We prove two lower bounds on this quantity for process functions based on the polynomial representation [24] and the certificate complexity [24] of Boolean functions, and we show that causal indefiniteness can at best provide a quadratic advantage in query complexity. We then look for explicit speedups obtainable using causal indefiniteness. First, starting from the so-called "Lugano" (or "AF/BW") process function [15, 19] we construct an explicit Boolean function for which causal indefiniteness allows the deterministic query complexity to be reduced from four queries to three. Then, we show that this constant separation can be amplified into a polynomial separation via a recursive construction, proving an asymptotic computational advantage from causal indefiniteness in the classical setting.

Finally, we build on these results in the classical-deterministic setting to prove that causal indefiniteness can also provide an equivalent constant advantage in quantum query complexity, improving upon [5] where only a reduced error for a fixed number of queries in the bounded-error model was obtained, and answering several questions left open in [5].

2 The deterministic query model of computation

Recently, a generalised notion of quantum query complexity was used to compare the computational power of sequential and causally indefinite quantum computations [5]. In this query complexity framework, one aims to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ using an algorithm that can only access the bits of the input $x \in \{0,1\}^n$ via queries to an oracle. This oracle can be taken to be either classical or quantum, and the computational complexity is quantified by counting the number of queries to the oracle. This measure of complexity is a standard tool to compare different types of computations, including deterministic and probabilistic classical computations, or quantum computations [24]. In particular, most of the known quantum advantages over classical algorithms can be understood in this framework [8].

Whereas [5] studied the quantum query complexity of causally indefinite computations, in this paper we first focus on the classical-deterministic query model of computation, where causal indefiniteness can

also arise [19]. Here, the query oracle encoding x is given as a function $O_x : \{1,\dots,n\} \to \{0,1\}$ such that $O_x(i) = x_i$. A *sequential* algorithm, which can be adaptive, can then be represented without loss of generality as a decision tree whose internal nodes are queries to the oracle O_x and whose leaves represent the final outputs of the computation [24]. Such a decision tree is said to compute a Boolean function f if, for all x, the evaluation of the tree outputs the value $f(x)$. The *decision tree complexity* of f, which we will refer to as *deterministic query complexity*, is defined as follows.

Definition 1 ([24])**.** The *deterministic query complexity* of a Boolean function f, denoted $D(f)$, is the minimum T for which there exists a decision tree of depth T computing f.

This sequential notion of deterministic query complexity serves as a baseline to assess the computational power of causal indefiniteness in classical-deterministic scenarios.

3 Classical-deterministic processes

To investigate the power of causal indefiniteness in the classical-deterministic query model, we will use the process function formalism [19] to model more general causally indefinite classical computations. This model encompasses standard sequential algorithms, and will allow us to compare them to causally indefinite computations in a unified framework.

The process function formalism describes how some deterministic operations (i.e., functions) $\vec{\mu} = (\mu_1,\dots,\mu_T)$, with $\mu_k : \mathscr{I}_k \to \mathscr{O}_k$, can be composed together in a logically consistent way [19]. This composition is described by the so-called process function $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$, where $\mathscr{P}$ and $\mathscr{F}$ are, respectively, the domain and codomain of the function resulting from the composition. A process function with T "slots" is thus defined in [19] as the most general kind of transformation that maps the T functions $\vec{\mu}$ into a function from $\mathscr{P}$ to $\mathscr{F}$.[1] Process functions can be elegantly characterised through a fixed-point condition: for any choice of $a \in \mathscr{P}$ and operations $\vec{\mu}$, the process function has a unique fixed-point $\vec{i}$, as shown by the following proposition [18].

Proposition 2 ([19, 18])**.** *A function $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ is a* process function *if and only if, for all $a \in \mathscr{P}$ and all $\vec{\mu} = (\mu_1,\dots,\mu_T)$ with $\mu_k = \mathscr{I}_k \to \mathscr{O}_k$,*

$$\exists!\, \vec{i} : w\big(a, \vec{\mu}(\vec{i})\big) = (\vec{i}, b), \tag{1}$$

with $\vec{i} = (i_1,\dots,i_T) \in \bigtimes_{k=1}^{T} \mathscr{I}_k$ and $b \in \mathscr{F}$.

Whereas a standard query algorithm can be thought of as composing queries in a sequential order, process functions do not assume any such causal order, yet remain classical and deterministic.[2] We will thus refer to process functions as *classical-deterministic processes*.

To study the comparative computational power of causally definite and indefinite processes in this framework, we begin by formalising the notion of a causally definite classical-deterministic process. To this end, we first introduce the *induced functions* $w_k : \mathscr{P} \times \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \mathscr{I}_k$ and $w_F : \mathscr{P} \times \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \mathscr{F}$ defined as $w_k(a,\vec{o}) = w(a,\vec{o})_{|k}$ and $w_F(a,\vec{o}) = w(a,\vec{o})_{|\mathscr{F}}$ which specify the inputs given to the different

[1] We assume throughout that $\mathscr{P}, \mathscr{I}_k, \mathscr{O}_k, \mathscr{F}$ are all nonempty. Note that if any of these are singletons then their role is trivial and they can be omitted from the specification of the process function. Note also that if $\mathscr{P}$ or $\mathscr{F}$ is a singleton, then the resulting function from $\mathscr{P}$ to $\mathscr{F}$ obtained from the process function is constant.

[2] One could also consider classical processes that are not required to be deterministic, for instance to study the bounded-error randomised query complexity of Boolean functions under indefinite causal order. Note, however, that the set of such classical processes is not simply the convex hull of classical-deterministic ones; in particular, some classical processes exhibit a form of so-called fine-tuning [19].

operations μ_k or output to $\mathscr{F}$ [17]. Note that, taken together, these uniquely define w. We will also make use of the notion of a *reduced process* $w^{|\mu_k} : \mathscr{P} \times \bigtimes_{\substack{k'=1 \\ k'\neq k}}^{T} \mathscr{O}_{k'} \to \bigtimes_{\substack{k'=1 \\ k'\neq k}}^{T} \mathscr{I}_{k'} \times \mathscr{F}$ obtained from w by fixing the kth operation μ_k [17]. The reduced process $w^{|a} : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ is similarly obtained by fixing the value a in the past $\mathscr{P}$. Note that $w^{|\mu_k}$ is a $(T-1)$-slot classical-deterministic process, while $w^{|a}$ is a T-slot process with trivial input (i.e., $|\mathscr{P}| = 1$), which we thus generally omit from its specification. Finally, to compactly describe the composition of w and operations $\vec{\mu}$ we will make use of a classical version of the so-called "link product" [28], written '$*$'. In particular, for any $a \in \mathscr{P}$ and operations $\vec{\mu}$, and writing $\vec{i}_a = (i_{a,1}, \ldots, i_{a,T})$ the inputs of the corresponding fixed-point of w we have $(w * \vec{\mu})(a) = w_F\big(\mu_1(i_{a,1}), \ldots, \mu_T(i_{a,T})\big)$. Full definitions of these notions are given in Appendix A.1.

A classical-deterministic process is then said to be causally definite if it is compatible with the operations μ_k being applied in a sequential order. This sequential order may be fixed by w, but importantly it can also be dynamical: the choice of which operation to apply at step k might depend on the choice of the previous operations applied (and their inputs), as well as the value of $a \in \mathscr{P}$ given to the process. We give a recursive definition inspired by the definition of multipartite causally separable quantum supermaps [43]. In Appendix C.2 we show that these definitions are consistent if one represents classical-deterministic processes as (diagonal, deterministic) process matrices.

Definition 3. A classical-deterministic process $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ is said to be *causally definite* if and only if $T = 1$, or

- if $|\mathscr{P}| = 1$, then there *exists* $1 \leq k \leq T$ such that the induced function w_k is constant, and such that for any operation $\mu_k : \mathscr{I}_k \to \mathscr{O}_k$, the reduced process $w^{|\mu_k}$ is itself causally definite;
- if $|\mathscr{P}| > 1$ (i.e., $\mathscr{P}$ is non-trivial), then for all $a \in \mathscr{P}$ the reduced process $w^{|a}$ is itself causally definite.

We end this section with an example of a *causally indefinite* classical-deterministic process, the so-called Lugano (or AF/BW) process [15, 19]. This is a three-slot classical-deterministic process $w_{\text{Lugano}} : \{0,1\}^3 \to \{0,1\}^3$ defined by the induced functions w_k (for $1 \leq k \leq 3$)

$$w_k(o_1, o_2, o_3) = (1 \oplus o_{k\oplus_3 1}) o_{k\oplus_3 2}, \tag{2}$$

where $k \oplus_3 l := [(k+l-1) \bmod 3] + 1$, for any $l \in \mathbb{Z}$. Note that w_{Lugano} is defined with trivial $\mathscr{P}$ and $\mathscr{F}$, which are thus omitted from its definition above. One can show that w_{Lugano} is logically consistent, as it verifies the unique fixed-point characterisation of Proposition 2 [16]. Furthermore, it is causally indefinite, as it is known to violate so-called causal inequalities [18]. The Lugano process will be an important tool in our goal to prove query complexity advantages over causally definite computations in Sections 5 and 7.

4 Generalised deterministic query complexity

In order to study the query complexity power of classical-deterministic processes, we first need to extend the notion of deterministic query complexity introduced in Definition 1 beyond causally-definite computations – which can be seen as evaluating queries to the oracle O_x in a decision tree – to the process function framework. This generalisation is extremely natural, and simply interprets classical-deterministic processes as connecting the inputs and outputs of the different queries in a logically consistent way. This generalisation is analogous to that proposed in [5] for the quantum query complexity using general quantum supermaps.

Definition 4. The *generalised deterministic query complexity* of a Boolean function f, denoted $D^{\text{Gen}}(f)$, is the minimum T for which there exists a classical-deterministic process $w : \{0,1\}^T \to \{1,\dots,n\}^T \times \{0,1\}$ computing f when making T queries to O_x, i.e., such that for all $x \in \{0,1\}^n$ and writing $\vec{O}_x = (\underbrace{O_x,\dots,O_x}_{\text{T}})$, we have $w * \vec{O}_x = f(x)$.

Note that, when computing a function f with a process function w, the output $f(x)$ is thus obtained in the "global future" $\mathscr{F} = \{0,1\}$. We show that when computing a Boolean function f in the query model, decision trees and causally definite classical-deterministic processes are equivalent, justifying the interpretation of $D^{\text{Gen}}(f)$ as a generalisation of $D(f)$.

Proposition 5. *In the query model, there exists a decision tree of depth T computing a Boolean function f if and only if there exists a T-slot causally definite classical-deterministic process computing f.*

Proof sketch. The implication in the forward direction is straightforward, as any decision tree of depth T can be easily seen as defining a causally definite T-slot classical-deterministic process.

For the other direction, while decision trees are always evaluated in a fixed sequential order, a causally definite classical-deterministic processes w may be dynamical, meaning that the operation applied at each step can depend on the outputs of the previous operations. However, this dynamicality is irrelevant when the T operations are identical, and one can always find an equivalent non-dynamical classical-deterministic process whose action is the same as that of w on T queries to O_x. This non-dynamical process can then be seen as a depth T decision tree. See Appendix A.2 for the full proof. □

It follows that for any Boolean function, f, $D^{\text{Gen}}(f) \leq D(f)$. By comparing the two complexities D and D^{Gen} we can compare, on an equal footing, the standard deterministic query model of computation, which is intrinsically sequential, with the generalised one, allowing causally indefinite computations. In particular, because the two complexities are defined relative to the same classical oracle, any separation between the two quantities provides proof of a computational advantage obtained from causal indefiniteness.

4.1 Degree and certificate lower bounds

Before looking for such separations, we first show that some well-known lower bounds on D can be generalised to D^{Gen}. Such lower bounds on D^{Gen} are insightful, as they bound any possible advantages obtainable from causal indefiniteness and provide insight into which Boolean functions are candidates for obtaining an advantage.

The first lower bound we consider is based on the polynomial representation of Boolean functions. A multilinear polynomial $g : \mathbb{R}^n \to \mathbb{R}$, is said to represent f if, for all $x \in \{0,1\}^n$, $f(x) = g(x)$. It is known that any Boolean function f has a unique multilinear polynomial representation, and that $\deg(f) \leq D(f) \leq \deg(f)^3$, where $\deg(f)$ denotes the degree of this representation [24]. We strengthen here this and show that $\deg(f)$ is also a lower bound on D^{Gen} (see Appendix A.3 for the proof).

Proposition 6. *For any Boolean function f,* $\deg(f) \leq D^{\text{Gen}}(f)$.

It follows that for any Boolean function f, $\deg(f) \leq D^{\text{Gen}}(f) \leq D(f) \leq \deg(f)^3$, meaning that for functions such that $\deg(f) = D(f)$, there cannot be any computational advantage from causal indefiniteness.

A second lower bound is obtained by looking at the certificate complexity of f [24]. The certificate complexity quantifies the number of bits of x one has to know, in the worst case, to compute $f(x)$.

Formally, a certificate for an input $x \in \{0,1\}^n$ of f is a set of indices $I \subseteq \{1,\ldots,n\}$ such that, for all $y \in \{0,1\}^n$, $y_{|I} = x_{|I}$ implies $f(x) = f(y)$, where $x_{|I}$ denotes the restriction of x to the bits x_i at indices $i \in I$. Writing $C(f,x)$ the smallest certificate for x, the certificate complexity of of f is defined as $C(f) := \max_x C(f,x)$. It is known that for any function f, $C(f) \leq D(f) \leq C(f)^2$ [24]. We show that $C(f)$ is also a lower bound of D^{Gen} (see Appendix A.3).

Proposition 7. *For any Boolean function f, $C(f) \leq D^{\text{Gen}}(f)$.*

It follows that for any Boolean function f, $C(f) \leq D^{\text{Gen}}(f) \leq D(f) \leq C(f)^2$, meaning that to obtain a separation between D^{Gen} and D, one also needs first a separation between C and D and that, furthermore, any separation will be at most quadratic.

These two lower bounds provide criteria for finding Boolean functions that could exhibit an advantage from causal indefiniteness. In particular, it is known that almost all Boolean functions have full degree, meaning that the fraction of Boolean functions with $\deg(f) < D(f)$ is $o(1)$ [24]. Nevertheless, we show in the next section the existence of a Boolean function satisfying both $\deg(f) < D(f)$ and $C(f) < D(f)$, and for which we obtain a separation between $D(f)$ and $D^{\text{Gen}}(f)$.

5 A constant separation between D^{Gen} and D

In order to demonstrate a separation between D^{Gen} and D and hence a query complexity advantage from causal indefiniteness, we construct here a specific 6-bit Boolean function f_{6c}. The function f_{6c} can be represented as the degree-3 polynomial

$$f_{6c}(x_1,\ldots,x_6) = x_4(1 \oplus x_2)x_3 \oplus x_5(1 \oplus x_3)x_1 \oplus x_6(1 \oplus x_1)x_2 \oplus x_1x_2x_3, \tag{3}$$

where $\oplus$ denotes addition modulo 2.[3] Note that the structure of f_{6c} is similar to the induced functions of the Lugano process (2), with several monomials of the form $x_{k+3}(1 \oplus x_{k\oplus_3 1})x_{k\oplus_3 2}$.

It follows from Eq. (3) that $\deg(f_{6c}) = 3$ and one can readily show that $C(f_{6c}) = 3$ as, for any x, the value of $f_{6c}(x)$ can be obtained from three bits of x. Furthermore, it is easy to compute f_{6c} in a sequential fashion using four queries, and indeed we show that four queries are also necessary, so that $D(f_{6c}) = 4$. These properties (for which we give detailed proofs in Appendix B) mean that f_{6c} satisfies the two conditions identified in Section 4.1 that make it a suitable candidate for a potential query complexity advantage from causal indefiniteness.

The following result shows that there is indeed a causally indefinite classical-deterministic process that computes f_{6c} in three queries, yielding such an advantage.

Proposition 8. $D^{\text{Gen}}(f_{6c}) = 3$.

Proof sketch. The classical-deterministic process that computes f_{6c} in three queries is based on the Lugano process $w_{\text{Lugano}} : \{0,1\}^3 \to \{0,1\}^3$. To consider the action of w_{Lugano} on the query oracle $O_x : \{1,\ldots,6\} \to \{0,1\}$, we modify the process and consider the function $\overline{w}_{\text{Lugano}} : \{0,1\}^3 \to \{1,\ldots,6\}^3 \times \{0,1\}$, to which we have added a non-trivial future space $\mathscr{F} = \{0,1\}$, and which is defined by the induced functions, for all $1 \leq k \leq 3$ and $(o_1,o_2,o_3) \in \{0,1\}^3$,

$$\overline{w}_k(o_1,o_2,o_3) = k + 3 \cdot w_k(o_1,o_2,o_3), \tag{4}$$

[3] One can easily check that Eq. (3) is equivalent (on Boolean inputs) to the multilinear polynomial $g(x_1,\ldots,x_6) = x_4(1-x_2)x_3 + x_5(1-x_3)x_1 + x_6(1-x_1)x_2 + x_1x_2x_3$.

where $w_k(o_1,o_2,o_3) = (1 \oplus o_{k\oplus_3 1})o_{k\oplus_3 2}$ are the induced functions of w_{Lugano} as in Eq. (2), and

$$\overline{w}_F(o_1,o_2,o_3) = o_1(1\oplus o_2)o_3 \oplus o_2(1\oplus o_3)o_1 \oplus o_3(1\oplus o_1)o_2 \oplus o_1o_2o_3. \tag{5}$$

The logical consistency of this function can be proven from that of the original Lugano process. To verify that it computes f_{6c}, one can check explicitly that $\overline{w}_F\big(O_x(i_{x,1}),O_x(i_{x,2}),O_x(i_{x,3})\big) = f_{6c}(x)$, where $\vec{i}_x = (i_{x,1},i_{x,2},i_{x,3})$ is the unique fixed-point of $\overline{w}_{\text{Lugano}}$ acting on three copies of O_x. The detailed proof is given in Appendix B. □

These results exhibit an explicit 6-bit function f_{6c} with $D(f_{6c}) = 4$ and $D^{\text{Gen}}(f_{6c}) = 3$, showing that:

Theorem 9. *There exists a Boolean function f for which $D^{\text{Gen}}(f) < D(f)$.*

This result proves that causal indefiniteness can provide a computational advantage in the standard framework of query complexity. In the next section, we show that this separation can serve as a stepping stone towards an asymptotic separation between D^{Gen} and D for a particular family of Boolean functions.

6 A polynomial advantage from causal indefiniteness

In this section we show that a polynomial query complexity advantage is possible by constructing a family of Boolean functions based on a recursive iteration of the 6-bit function f_{6c} defined in Eq. (3). Similar recursive constructions have previously used to amplify constant separations between exact quantum query complexity and deterministic query complexity into asymptotic ones (see, e.g., [22, 7, 38]). Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the construction considers the recursively defined functions

$$f^{(l+1)}(x_1,\ldots,x_{n^{l+1}}) = f\big(f^{(l)}(x_1,\ldots,x_{n^l}),\ldots,f^{(l)}(x_{(n-1)n^l+1}\ldots x_{n^{l+1}})\big), \tag{6}$$

with $f^{(1)} := f$. The deterministic query complexity of these recursively defined functions is determined by the initial function f: if $D(f) = T$ then $D(f^{(l)}) = T^l$ [37].

By proving an analogous theorem for D^{Gen}, one could amplify any constant separation between D^{Gen} and D given by a function f (such as f_{6c}) into a polynomial separation. Here we prove a slightly weaker result, which is nevertheless sufficient to obtain such a separation, and leave open the question of whether or not one can prove equality under composition for the generalised deterministic query complexity.

Theorem 10. *For any Boolean function f and for any depth $l > 1$,*

$$D^{\text{Gen}}(f) = T \implies D^{\text{Gen}}(f^{(l)}) \leq T^l. \tag{7}$$

Proof sketch. Starting with a classical-deterministic process $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ computing f (which necessarily has $\mathscr{O}_k = \mathscr{F} = \{0,1\}$ and $\mathscr{I}_k = \{1,\ldots,n\}$) and another classical-deterministic process $w^{(l)} : \bigtimes_{k=1}^{T^l} \mathscr{O}_k \to \bigtimes_{k=1}^{T^l} \mathscr{I}_k^{(l)} \times \mathscr{F}$ (with $\mathscr{I}_k^{(l)} = \{1,\ldots,n^l\}$) computing $f^{(l)}$, the goal is to construct a process function $w^{(l+1)}$ computing $f^{(l+1)}$.

We first construct a classical-deterministic process $\tilde{w}^{(l)} : \mathscr{P} \times \bigtimes_{k=1}^{T^l} \mathscr{O}_k \to \bigtimes_{k=1}^{T^l} \mathscr{I}_k^{(l+1)} \times \mathscr{F}$, with $\mathscr{P} = \{1,\ldots,n\}$, that computes $f^{(l)}$ on the n consecutive sequences of length n^l of a bitstring $x \in \{0,1\}^{n^{l+1}}$. It is defined by its induced functions $\tilde{w}_k^{(l)}(a,o_1,\ldots,o_{T^l}) = (a-1)n^l + w_k^{(l)}(o_1,\ldots,o_{T^l})$, and $\tilde{w}_F^{(l)}(a,o_1,\ldots,o_{T^l}) = w_F(o_1,\ldots,o_{T^l})$, and its logical consistency can be seen to following from that of $w^{(l)}$. In particular, this function is such that, for any $x \in \{0,1\}^{n^{(l+1)}}$ and $a \in \mathscr{P}$, we have $(\tilde{w}^{(l)} * \vec{O}_x)(a) = f^{(l)}(x_{(a-1)n^l+1},\ldots,x_{an^l})$.

We then construct a classical-deterministic process $w^{(l+1)}$ from $\tilde{w}^{(l)}$ that computes $f^{(l+1)}$ in T^{l+1} queries. This function is defined for any T^{l+1} operations $\vec{\mu} = (\vec{\mu}_1, \ldots, \vec{\mu}_T)$, with $\vec{\mu}_k = (\mu_{k,1}, \ldots, \mu_{k,T^l})$ as

$$w^{(l+1)} * \vec{\mu} = w * (\tilde{w}^{(l)} * \vec{\mu}_1, \ldots, \tilde{w}^{(l)} * \vec{\mu}_T) \in \mathscr{F}. \tag{8}$$

It follows from the properties of w and $\tilde{w}^{(l)}$ that $w^{(l+1)}$ is logically consistent, and that it computes $f^{(l+1)}$ in $T+1$ queries. We provide a full proof in Appendix A.4. □

Theorem 10 shows that we can amplify the separation obtained in Section 5 for the function f_{6c}, such that $D^{\text{Gen}}(f_{6c}^{(l)}) \leq 3^l$ while $D(f_{6c}) = 4^l$. With that, the family of functions $\{f_{6c}^{(l)}\}_l$ gives us the following result.

Theorem 11. *There exists a family of Boolean functions $\{f_l\}_l$ with $D(f_l) \to \infty$ such that*

$$D^{\text{Gen}}(f_l) = O(D(f_l)^{0.792\ldots}). \tag{9}$$

Theorem 11 shows a polynomial separation between causally definite and indefinite computations obtained in a standard complexity-theoretic model. It provides a clear proof that causal indefiniteness can provide asymptotic advantages in query complexity, analogous to known separations between classical and quantum computational models [8]. Note moreover that in the setting we regard both sequential and causally indefinite computations are given access to the *same* type of oracle O_x – something not true in the comparison of classical and quantum query complexity.

Whether the separation we obtained is tight or not is an open question. As $D^{\text{Gen}}(f_{6c}) = 3$, proving a stronger version of Theorem 10 with an equality instead of an upper-bound would imply a tight separation for the family of function $\{f_{6c}^{(l)}\}_l$. Nevertheless, as shown in Section 4.1, $C(f) \leq D^{\text{Gen}}(f) \leq D(f) \leq C(f)^2$, and hence the largest possible separation is quadratic.

7 From a classical to quantum query complexity advantage

The asymptotic advantage in a fully classical setting shown in the previous section highlights the fact it is not only interesting to study causally indefinite computations in quantum settings. Nonetheless, one of our primary motivations for studying this classical scenario was to gather insight into the quantum case. In [5], where the quantum query complexity of Boolean functions was first studied, it was shown that indefinite causal order can reduce the probability of error in computing some Boolean functions with a fixed number of queries. However, no advantage in the number of queries to compute a function exactly was found, nor was any asymptotic separation – either in the bounded-error or exact case – observed. Inspired by the stronger results obtained in Section 5 for the classical-deterministic setting, we revisit here the quantum setting and show that a constant separation in query complexity can also be obtained.

More precisely, to show a separation in the quantum setting, where both sequential and causally indefinite processes can query superpositions of inputs, we consider another 6-bit Boolean function f_{6q}, obtained by modifying the function f_{6c} used in Section 5. We show that while causally definite quantum supermaps cannot compute this new function in three quantum queries, a modified quantum version of the Lugano process can compute it in three quantum queries. We finish by discussing whether or not such a constant advantage can be turned into an asymptotic with a similar method as that used in Section 6 in the classical case.

7.1 Quantum supermaps and the quantum query model of computation

We begin by briefly introducing the framework of quantum supermaps and the quantum query model of computation, as well as some different mathematical tools and notation.

The process function formalism presented in Section 3 can be viewed as the classical, deterministic limit of a more general framework, that of quantum supermaps [19], which describes how quantum operations can be consistently composed and transformed [27]. A quantum supermap $\mathscr{S}$ with T "slots" is then the most general transformation that maps any T quantum channels $\mathscr{M}_k : \mathscr{L}(\mathscr{H}^{I_k}) \to \mathscr{L}(\mathscr{H}^{O_k})$ (i.e., completely positive (CP) and trace-preserving (TP) maps) into another quantum channel $\mathscr{S}(\mathscr{M}_1,\ldots,\mathscr{M}_T) : \mathscr{L}(\mathscr{H}^P) \to \mathscr{L}(\mathscr{H}^F)$. To be operationally consistent, $\mathscr{S}$ is required to be T-linear, completely CP-preserving and TP-preserving [27]. Using the Choi-Jamiołkowski isomorphism [30, 34], a quantum channel can be represented as a positive semidefinite matrix $\mathsf{M}_k = \sum_{i,i'} |i\rangle\langle i'| \otimes \mathscr{M}_k(|i\rangle\langle i'|) \in \mathscr{L}(\mathscr{H}^{I_k} \otimes \mathscr{H}^{O_k})$ satisfying $\mathrm{Tr}_{O_k}(\mathsf{M}_k) = \mathbb{1}^{I_k}$, with Tr_{O_k} being the partial trace over the system $\mathscr{H}^{O_k}$. Writing $\mathscr{H}^{(IO)_k} = \mathscr{H}^{I_k O_k} = \mathscr{H}^I \otimes \mathscr{H}^O$, $[T] := \{1,\ldots,T\}$, and $\mathscr{H}^{(IO)_{[T]}} = \bigotimes_{k\in[T]} \mathscr{H}^{(IO)_k}$, a quantum supermap can be also represented in the Choi picture as a "process matrix" [39], a positive semidefinite matrix $W \in \mathscr{L}(\mathscr{H}^{P(IO)_{[T]}F})$ belonging to a specific subspace $\mathscr{L}^{\mathrm{Gen}}$ (see Appendix B of [5] for more details) and satisfying a normalisation constraint [9, 43]. Quantum supermaps that compose the $\mathscr{M}_k$ in a fixed causal order represent standard quantum circuits or "quantum combs" [26], and correspond to process matrices belonging to a subspace $\mathscr{L}^{\mathrm{Seq}} \subset \mathscr{L}^{\mathrm{Gen}}$ (see Appendix C of [5]). We will generically denote general and sequential quantum supermaps as $\mathscr{S}^{\mathrm{Gen}}$ and $\mathscr{S}^{\mathrm{Seq}}$, respectively.

The quantum query model of computation was extended from the standard, causally ordered setting [24, 8] to general, potentially causally indefinite, quantum supermaps in [5]. This generalisation is analogous to that presented in Section 2 for the classical setting, with the crucial conceptual difference being that the classical oracles O_x must be generalised to unitary quantum oracles $\widetilde{O}_x : \mathscr{H}^Q \to \mathscr{H}^{Q'}$, where $\mathscr{H}^Q, \mathscr{H}^{Q'}$ are isomorphic $(n+1)$-dimensional Hilbert spaces[4] such that, for any index $i \in \{0,\ldots,n\}$,

$$\widetilde{O}_x |i\rangle = \begin{cases} (-1)^{x_i} |i\rangle & \text{if } i \neq 0, \\ |i\rangle & \text{otherwise.} \end{cases} \tag{10}$$

In particular, this allows the input bits to be queried in superposition. Note that the case $|i\rangle = |0\rangle$ is needed to ensure $\widetilde{O}_x$ is equivalent to the potentially more familiar oracle $\hat{O}_x |j,b\rangle = |j, b \oplus x_j\rangle$, with $1 \leq j \leq n$, $b \in \{0,1\}$ [6, 8]. We denote the quantum channel associated to the unitary $\widetilde{O}_x$ as $\widetilde{\mathscr{O}}_x$, and its Choi matrix as $\widetilde{\mathsf{O}}_x$.

In this paper, we adopt the generalised definition of quantum query complexity formulated in [5] in terms of quantum supermaps. This formulation is the quantum analogue of the generalised classical-deterministic query complexity (Definition 4), and allows one to compare the quantum query complexity of causally definite and indefinite quantum supermaps on an equal footing.

Definition 12. The *generalised quantum query complexity* of a Boolean function f, $Q_E^{\mathrm{Gen}}(f)$, is the minimum T for which there exists a T-slot quantum supermap $\mathscr{S}^{\mathrm{Gen}}$ such that for all $x \in \{0,1\}$, measuring the qubit $\mathscr{S}^{\mathrm{Gen}}(\underbrace{\widetilde{\mathscr{O}}_x,\ldots,\widetilde{\mathscr{O}}_x}_{\text{T copies}}) = \rho_x \in \mathscr{L}(\mathscr{H}^F)$ in the computational basis gives $f(x)$ with probability 1.[5]

When restricted to quantum supermaps that are sequential, i.e., supermaps $\mathscr{S}^{\mathrm{Seq}}$ whose process matrix representation belongs to the linear subspace $\mathscr{L}^{\mathrm{Seq}} \subset \mathscr{L}^{\mathrm{Gen}}$, one then recovers a notion of sequential

[4] We distinguish here and below between isomorphic primed and unprimed spaces, indicating inputs and outputs, so that the corresponding Choi matrices can be unambiguously composed, in particular with process matrices (cf. Fig. 1).

[5] We focus in this paper on the exact complexity Q_E^{Gen}, but the bounded-error complexity Q_2^{Gen} is also of interest [5].

query complexity $Q_E^{\text{Seq}}(f)$ which is equivalent to the standard notion of exact query complexity $Q_E(f)$ defined with quantum circuits. Note that it is known, for the particular task of computing Boolean functions in the query model, that the optimal causally definite computation can always be assumed to be sequential [5], even if more general causally definite computations are generally possible (e.g., with dynamical causal order) [44].

Some relations between the different query complexities introduced so far can be readily given. By definition we have, for any f, $Q_E^{\text{Gen}}(f) \leq Q_E(f)$ and furthermore, as the polynomial lower bound on Q_E is also a lower bound of Q_E^{Gen} [5], $\frac{\deg(f)}{2} \leq Q_E^{\text{Gen}}(f) \leq Q_E(f) \leq \deg(f)^3$. Because the classical-deterministic processes are subsets of quantum supermaps [19] (see Appendix C.1), one also has $Q_E^{\text{Gen}}(f) \leq D^{\text{Gen}}(f)$. It then follows, by the polynomial lower bound on Q_E^{Gen} and since $D^{\text{Gen}}(f) \leq \deg(f)^3$ (see Section 4.1), that these two generalised query complexities are polynomially related. Finally, note that, in general, it is unclear exactly how $Q_E(f)$ and $D^{\text{Gen}}(f)$ are related.

From these bounds, we see that only Boolean functions for which $Q_E > \deg(f)/2$ (i.e., the polynomial bound is not tight) are candidates to prove a separation between Q_E^{Gen} and Q_E. In the next section, we show that such a separation is indeed possible.

7.2 A constant separation between Q_E and Q_E^{Gen}.

The 6-bit function f_{6c} studied in Section 5 can easily be seen to have $Q_E^{\text{Gen}}(f_{6c}) = 3$, since $\bar{w}_{\text{Lugano}}$ can itself be seen as a quantum supermap. However, while $D(f_{6c}) = 4$ implies that $Q_E(f_{6c}) \leq 4$, it is not immediate that this bound is tight in the quantum setting. Indeed, using a semidefinite programming (SDP) formulation of the problem [14] (cf. Appendix D.1), one finds (up to numerical precision in the probability of obtaining the correct value of $f_{6c}(x)$) that $Q_E(f_{6c}) = 3$, and hence no advantage is obtained for this function in the quantum setting. To prove a separation in terms of exact quantum query complexity, we instead modify the function f_{6c} to define the new 6-bit Boolean function

$$f_{6q}(x_1,\ldots,x_6) = f_{6c}(x_1 \oplus x_4, x_2 \oplus x_5, x_3 \oplus x_6, x_4, x_5, x_6). \tag{11}$$

This function differs from f_{6c} in that the references to x_1, x_2 and x_3 are replaced by $x_1 \oplus x_4$, $x_2 \oplus x_5$ and $x_3 \oplus x_6$ respectively. The extra difficulty of computing the parity of certain input bits at the same time as computing f_{6c} itself will incur an extra cost in sequential algorithms, but, as we will see, can be performed at no extra cost by general supermaps.

Because f_{6c} can be computed in four classical queries, it follows that f_{6q} can be computed in four quantum queries, as the parity between two bits can be computed in a single quantum query [31]. Indeed, defining for $1 \leq i, j \leq n$ a unitary $H_{i,j} : \mathscr{H}^Q \to \mathscr{H}^Q$ such that $H_{i,j}|0\rangle = \frac{|i\rangle+|j\rangle}{\sqrt{2}}$ and $H_{i,j}|1\rangle = \frac{|i\rangle-|j\rangle}{\sqrt{2}}$ (and which is completed elsewhere to be unitary), we have $H_{i,j}\widetilde{O}_x H_{i,j}|0\rangle = (-1)^{x_i}|x_i \oplus x_j\rangle$. Using this subroutine to compute the parity between $x_1 \oplus x_4$, $x_2 \oplus x_5$ and $x_3 \oplus x_6$, one needs to query at most one of x_4, x_5, x_6 in order to compute f_{6q} in the same manner that one computes f_{6c} using four classical queries to O_x.

To prove that this sequential computation is optimal, we again use the SDP formulation of the problem due to [14] that we detail in Appendix D.1. This SDP allows one to obtain the minimum error $\varepsilon_T^{\text{Seq}}(f)$ with which one can compute a Boolean function f using T quantum queries. We find numerically that $\varepsilon_3^{\text{Seq}}(f_{6q}) \approx 0.0207 > 0$, and hence obtain the following proposition (see Appendix D.1 for details).

Proposition 13. $Q_E(f_{6q}) = 4$.

In the next section we construct a causally indefinite quantum supermap, once again based on the Lugano process, which can compute the function f_{6q} in three quantum queries, showing a quantum query complexity advantage from causal indefiniteness.

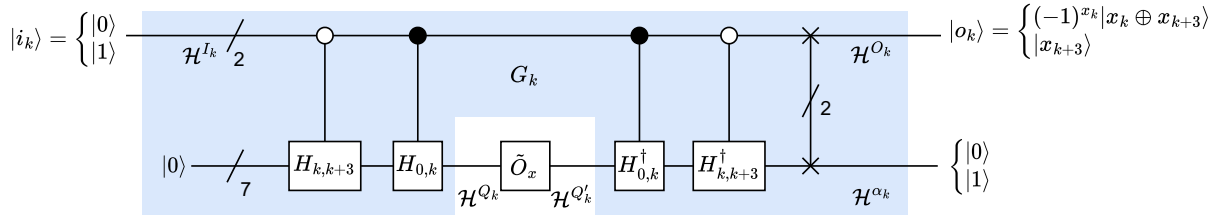


Figure 1: Circuit diagram of the quantum subroutines G_k. Here, the Hilbert space $\mathscr{H}^{\alpha_k}$ has basis states $|i\rangle$ for $0 \leq i \leq 6$, and $H_{i,j}$ are unitaries satisfying $H_{i,j}|0\rangle = \frac{|i\rangle+|j\rangle}{\sqrt{2}}$ and $H_{i,j}|1\rangle = \frac{|i\rangle-|j\rangle}{\sqrt{2}}$ for $0 \leq i,j \leq 6$, $i \neq j$, and which can be completed arbitrarily on the other inputs. The final operation is a swap operation $\text{SWAP} : \mathscr{H}^{O_k} \otimes \mathscr{H}^{\alpha_k} \to \mathscr{H}^{O_k} \otimes \mathscr{H}^{\alpha_k}$ such that for any $a,b \in \{0,1\}$, $\text{SWAP}\,|a\rangle^{O_k}|b\rangle^{\alpha_k} = |b\rangle^{O_k}|a\rangle^{\alpha_k}$ and for any $c \in \{2,\ldots,6\}$, $\text{SWAP}\,|a\rangle^{O_k}|c\rangle^{\alpha_k} = |a\rangle^{O_k}|c\rangle^{\alpha_k}$.

7.2.1 A causally indefinite quantum supermap to compute f_{6q}

As the function f_{6q}, defined by Eq. (11), is constructed from f_{6c}, and because the latter can be computed with the Lugano process in three queries, a modified version of the Lugano process is a natural candidate to compute f_{6q} in three queries. Although the Lugano process w_{Lugano} is a classical-deterministic process, it can be viewed as a quantum supermap whose process matrix is diagonal in a fixed pointer basis [19, 11]. To compute f_{6q} we will compose this supermap with some quantum subroutines to compute the parity between two given bits in one quantum query. Such subroutines can themselves be viewed as quantum supermaps taking a single quantum channel – here, the oracle – as input. We prove the following proposition, whose full proof is given in Appendix D.2.

Proposition 14. $Q_E^{\text{Gen}}(f_{6q}) = 3$.

Proof sketch. The classical-deterministic Lugano process w_{Lugano} can be embedded in the process matrix formalism as a process matrix $W_{\text{Lugano}} \in \mathscr{L}(\mathscr{H}^{(IO)_{[3]}})$ where, for $1 \leq k \leq 3$, the $\mathscr{H}^{O_k}$ and $\mathscr{H}^{I_k}$ are 2-dimensional Hilbert spaces [19, 11]. Because the process is classical, we can further amend W_{Lugano} with a future space $\mathscr{H}^F$ that keeps a copy of the classical values in the registers $\mathscr{H}^{O_1O_2O_3}$, defining

$$\begin{aligned} \widetilde{W}_{\text{Lugano}} = \sum_{o_1,o_2,o_3 \in \{0,1\}} & |o_1,o_2,o_3\rangle\langle o_1,o_2,o_3|^{O_1O_2O_3} \\ & \otimes |\overline{o}_2o_3, \overline{o}_3o_1, \overline{o}_1o_2\rangle\langle \overline{o}_2o_3, \overline{o}_3o_1, \overline{o}_1o_2|^{I_1I_2I_3} \otimes |o_1,o_2,o_3\rangle\langle o_1,o_2,o_3|^F , \end{aligned} \tag{12}$$

where $\overline{o}_k := (1 \oplus o_k)$. We proceed by composing this process with some quantum subroutines G_k, for $k \in \{1,2,3\}$, which adapt the qubit slots of $\widetilde{W}_{\text{Lugano}}$ for the oracle $\widetilde{O}_x$ which operates on a 7-dimensional input, and which can be used to compute the parity between different bits of x in one quantum query. These subroutines, which correspond to 1-slot quantum supermaps whose additional outputs in the spaces $\mathscr{H}^{\alpha_k}$ are also sent to the global future, as well as their actions on the query oracle $\widetilde{O}_x$, are described in Figure 1.

The composition of $\widetilde{W}_{\text{Lugano}}$ with the G_k in each slot defines a new three-slot quantum supermap with global future $\mathscr{H}^{F\alpha_1\alpha_2\alpha_3}$. Its process matrix is given by the link product as $\widetilde{W}_{f_{6q}} := \widetilde{W}_{\text{Lugano}} * G_1 * G_2 * G_3 \in \mathscr{L}(\mathscr{H}^{Q_1Q_2Q_3Q'_1Q'_2Q'_3F\alpha_1\alpha_2\alpha_3})$. We then show that, for any value of x, when acting on three copies of the query oracle $\widetilde{O}_x$, we can obtain enough information to compute f_{6q} by measuring the resulting state $\widetilde{W}_{f_{6q}} * (\widetilde{\mathsf{O}}_x \otimes \widetilde{\mathsf{O}}_x \otimes \widetilde{\mathsf{O}}_x) \in \mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3})$ in the computational basis. □

Together, the propositions above show that the explicit 6-bit Boolean function f_{6q} implies the following key result proving a separation in exact quantum query complexity obtained with a causally indefinite supermap.

Theorem 15. *There exists a Boolean function f for which $Q_E^{Gen}(f) < Q(f)$.*

Given that the constant separation obtained in Section 6 for the classical-deterministic setting could be amplified into an asymptotic polynomial separation, it is extremely natural to wonder whether a similar such separation can be obtained in the quantum case. The recursive approach that was utilised in Section 6 cannot, however, be directly applied to the constant quantum advantage obtained with $\widetilde{W}_{f_{6q}}$, despite this method previously being employed to exhibit quantum-over-classical query complexity advantages [7]. The main difficulty encountered is that $\widetilde{W}_{f_{6q}}$ does not describe a "clean" quantum supermap [2], meaning that the output space of $\widetilde{W}_{f_{6q}} * \widetilde{\mathsf{O}}_x^{\otimes 3} \in \mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3})$ contains information about $f_{6q}(x)$, but also some additional information that cannot be uncomputed without additional queries to $\widetilde{O}_x$ – which would negate the advantage obtained from causal indefiniteness. While this is not a problem when computing $f_{6q}(x)$, it is an issue if one wants to coherently call this supermap recursively as a subroutine (see Appendix D.2 for further discussion).

8 Discussion

In this paper we studied the potential of causally indefinite computations to provide advantages in query complexity, both in the classical-deterministic and quantum settings. In the classical setting, we showed that classical-deterministic process functions can be used to obtain an asymptotic polynomial advantage in the query complexity of a Boolean function over causally ordered computations. Although based on the well-studied Lugano process [15, 19], our asymptotic separation required recursively composing together such processes to efficiently compute the function we constructed, an approach that has not previously been exploited when studying causally indefinite information processing. Our results highlight that interest in causally indefinite computation is not restricted to the quantum setting where most advantages have been studied [20]. Here we focused exclusively on the deterministic setting, but it would be interesting to study also the potential advantages obtainable with more general (non-deterministic, stochastic) classical processes – which are not simply described as the convex-hull of classical-deterministic processes [19] – in both the exact and bounded-error query complexity settings.

With the insights obtained from the study of the simpler classical-deterministic setting, we then proved a constant separation between the exact quantum query complexity Q_E and its generalised version Q_E^{Gen} introduced in [5], answering several questions that were left open therein. This reinforces the observation that causal indefiniteness can be a resource for computation even in more standard "classical" problems – such as computing a Boolean function – and not only in inherently quantum problems with no classical analogue, as most previous studies have investigated [10, 42, 35]. Nevertheless, in contrast to the classical setting, we failed to amplify the observed separation with a recursive construction, and highlighted the need to uncompute the "garbage" ancillary qubits using additional queries [2] as a barrier to exploiting this technique. We leave it as an open question to find an asymptotic separation in quantum query complexity in this setting.

Finally, we note that it would be interesting to study the query complexity for partial Boolean functions, i.e., with a promise on their inputs. Indeed, even in standard quantum computational settings it is only possible to obtain super-polynomial quantum-over-classical advantages in query complexity for partial functions [14], as is the case, e.g., for the Deutsch-Jozsa problem [32, 31] or Simon's problem [41].

We showed that for total functions one can, at best, have a polynomial advantage from indefinite causal order in both classical-deterministic and quantum settings, but for partial functions larger advantages have not been ruled out.

Acknowledgments

The authors acknowledge funding from the French National Research Agency through the Plan France 2030 projects ANR-22-CMAS-0001 and ANR-22-PETQ-0007, and the research grant ANR-22-CE47-0012. For the purpose of open access, the authors have applied a CCBY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

[1] *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.* `https://www.mosek.com/`.

[2] Scott Aaronson (2003): *Quantum lower bound for recursive Fourier sampling*. *Quantum Info. Comput.* 3(2), pp. 165–174, doi:10.26421/QIC5.2-9. arXiv:0209060.

[3] Scott Aaronson (2005): *Guest column: NP-complete problems and physical reality*. *SIGACT News* 36(1), pp. 30–52, doi:10.1145/1052796.1052804. arXiv:0502072.

[4] Scott Aaronson & John Watrous (2009): *Closed timelike curves make quantum and classical computing equivalent*. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* 465(2102), pp. 631–647, doi:10.1098/rspa.2008.0350. arXiv:0808.2669.

[5] Alastair A. Abbott, Mehdi Mhalla & Pierre Pocreau (2024): *Quantum query complexity of Boolean functions under indefinite causal order*. *Phys. Rev. Res.* 6(3), p. L032020, doi:10.1103/PhysRevResearch.6.L032020. arXiv:2307.10285.

[6] Andris Ambainis (2002): *Quantum Lower Bounds by Quantum Arguments*. *J. Comput. Syst. Sci.* 64(4), pp. 750–767, doi:10.1006/jcss.2002.1826. arXiv:quant-ph/0002066.

[7] Andris Ambainis (2013): *Superlinear advantage for exact quantum algorithms*. In: *Proc. Annu. ACM Symp. Theory Comput.*, STOC '13, ACM, Palo Alto, California, USA, pp. 891–900, doi:10.1145/2488608.2488721. arXiv:1211.0721.

[8] Andris Ambainis (2018): *Understanding quantum algorithms via query complexity*. In: *Proc. Int. Congr. Math.*, ICM 2018, World Scientific, pp. 3265–3285, doi:10.1142/9789813272880_0181. arXiv:1712.06349.

[9] Mateus Araújo, Cyril Branciard, Fabio Costa, Adrien Feix, Christina Giarmatzi & Časlav Brukner (2015): *Witnessing causal nonseparability*. *New J. Phys.* 17(10), p. 102001, doi:10.1088/1367-2630/17/10/102001. arXiv:1506.03776.

[10] Mateus Araújo, Fabio Costa & Časlav Brukner (2014): *Computational advantage from quantum-controlled ordering of gates*. *Phys. Rev. Lett.* 113(25), p. 250402, doi:10.1103/PhysRevLett.113.250402. arXiv:1401.8127.

[11] Mateus Araújo, Adrien Feix, Miguel Navascués & Časlav Brukner (2017): *A purification postulate for quantum mechanics with indefinite causal order*. *Quantum* 1, p. 10, doi:10.22331/q-2017-04-26-10. arXiv:1611.08535.

[12] Mateus Araújo, Philippe Allard Guérin & Ämin Baumeler (2017): *Quantum computation with indefinite causal structures*. *Phys. Rev. A* 96, p. 052315, doi:10.1103/PhysRevA.96.052315. arXiv:1706.09854.

[13] Dave Bacon (2004): *Quantum computational complexity in the presence of closed timelike curves*. *Phys. Rev. A* 70(3), p. 032309, doi:10.1103/PhysRevA.70.032309. arXiv:0309189.

[14] Howard Barnum, Michael Saks & Mario Szegedy (2003): *Quantum query complexity and semi-definite programming*. In: *Proc. IEEE Annu. Conf. Comput. Complexity*, CCC '18, IEEE, pp. 179–193, doi:10.1109/CCC.2003.1214419.

[15] Ämin Baumeler, Adrien Feix & Stefan Wolf (2014): *Maximal incompatibility of locally classical behavior and global causal order in multiparty scenarios*. *Phys. Rev. A* 90(4), p. 042106, doi:10.1103/PhysRevA.90.042106. arXiv:1403.7333.

[16] Ämin Baumeler, Amin Shiraz Gilani & Jibran Rashid (2022): *Unlimited non-causal correlations and their relation to non-locality*. *Quantum* 6, p. 673, doi:10.22331/q-2022-03-29-673. arXiv:2104.06234.

[17] Ämin Baumeler & Eleftherios Tselentis (2021): *Equivalence of grandfather and information antinomy under intervention*. In: *Proc. 17th Int. Conf. Quantum Phys. Logic*, QPL 2021, OPA, doi:10.4204/EPTCS.340.1. arXiv:2004.12921.

[18] Ämin Baumeler & Stefan Wolf (2016): *Device-independent test of causal order and relations to fixed-points*. *New J. Phys.* 18(3), p. 035014, doi:10.1088/1367-2630/18/3/035014. arXiv:1511.05444.

[19] Ämin Baumeler & Stefan Wolf (2016): *The space of logically consistent classical processes without causal order*. *New J. Phys.* 18(1), p. 013036, doi:10.1088/1367-2630/18/1/013036. arXiv:1507.01714.

[20] Ämin Baumeler & Stefan Wolf (2018): *Computational tameness of classical non-causal models*. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* 474(2209), p. 20170698, doi:10.1098/rspa.2017.0698. arXiv:1611.05641.

[21] Jessica Bavaresco, Mio Murao & Marco Túlio Quintino (2021): *Strict hierarchy between parallel, sequential, and indefinite-causal-order strategies for channel discrimination*. *Phys. Rev. Lett.* 127(20), p. 200504, doi:10.1103/PhysRevLett.127.200504. arXiv:2011.08300.

[22] Ethan Bernstein & Umesh Vazirani (1997): *Quantum Complexity Theory*. *SIAM J. Comput.* 26(5), pp. 1411–1473, doi:10.1137/S0097539796300921.

[23] Todd A. Brun (2003): *Computers with closed timelike curves can solve hard problems efficiently*. *Found. Phys. Lett.* 16, pp. 245–253, doi:10.1023/A:1025967225931. arXiv:0209061v1.

[24] Harry Buhrman & Ronald de Wolf (2002): *Complexity measures and decision tree complexity: a survey*. *Theor. Comput. Sci.* 288(1), pp. 21–43, doi:10.1016/S0304-3975(01)00144-X.

[25] Giulio Chiribella (2012): *Perfect discrimination of no-signalling channels via quantum superposition of causal structures*. *Phys. Rev. A* 86(4), p. 040301, doi:10.1103/PhysRevA.86.040301. arXiv:1109.5154.

[26] Giulio Chiribella, Giacomo Mauro D'Ariano & Paolo Perinotti (2008): *Quantum Circuit Architecture*. *Phys. Rev. Lett.* 101(6), p. 060401, doi:10.1103/PhysRevLett.101.060401. arXiv:0712.1325.

[27] Giulio Chiribella, Giacomo Mauro D'Ariano & Paolo Perinotti (2008): *Transforming quantum operations: Quantum supermaps*. *Europhys. Lett.* 83(3), p. 30004, doi:10.1209/0295-5075/83/30004. arXiv:0804.0180.

[28] Giulio Chiribella, Giacomo Mauro D'Ariano & Paolo Perinotti (2009): *Theoretical framework for quantum networks*. *Phys. Rev. A* 80(2), p. 022339, doi:10.1103/PhysRevA.80.022339. arXiv:0904.4483.

[29] Giulio Chiribella, Giacomo Mauro D'Ariano, Paolo Perinotti & Benoît Valiron (2013): *Quantum computations without definite causal structure*. *Phys. Rev. A* 88(2), p. 022318, doi:10.1103/PhysRevA.88.022318. arXiv:0912.0195.

[30] Man-Duen Choi (1975): *Completely Positive Linear Maps on Complex Matrices*. *Linear Algebra Appl.* 10(3), pp. 285–290, doi:10.1016/0024-3795(75)90075-0.

[31] Richard Cleve, Artur Ekert, Chiara Macchiavello & Michele Mosca (1998): *Quantum algorithms revisited*. *Proc. R. Soc. A* 454(1969), pp. 339–354, doi:10.1098/rspa.1998.0164. arXiv:9708016.

[32] David Deutsch & Richard Jozsa (1992): *Rapid solution of problems by quantum computation*. *Proc. R. Soc. A* 439(1907), pp. 553–558, doi:10.1098/rspa.1992.0167.

[33] Philippe Allard Guérin, Adrien Feix, Mateus Araújo & Časlav Brukner (2016): *Exponential Communication Complexity Advantage from Quantum Superposition of the Direction of Communication*. *Phys. Rev. Lett.* 117, p. 100502, doi:10.1103/PhysRevLett.117.100502. arXiv:1605.07372.

[34] Andrzej Jamiołkowski (1972): *Linear transformations which preserve trace and positive semidefiniteness of operators*. Rep. Math. Phys. 3(4), pp. 275–278, doi:10.1016/0034-4877(72)90011-0.

[35] Hlér Kristjánsson, Tatsuki Odake, Satoshi Yoshida, Philip Taranto, Jessica Bavaresco, Marco Túlio Quintino & Mio Murao (2024): *Exponential separation in quantum query complexity of the quantum switch with respect to simulations with standard quantum circuits*. arXiv:2409.18420.

[36] Johan Löfberg (2004): *YALMIP: a toolbox for modeling and optimization in MATLAB*. In: *2004 Proc. IEEE Int. Conf. Robot. Autom.*, ICRA 2004, IEEE, pp. 284–289, doi:10.1109/CACSD.2004.1393890.

[37] Ashley Montanaro (2014): *A composition theorem for decision tree complexity*. Chicago J. Theoret. Comput. Sci. 2014(6), doi:10.4086/cjtcs.2014.006. arXiv:1302.4207.

[38] Ashley Montanaro, Richard Jozsa & Graeme Mitchison (2015): *On exact quantum query complexity*. Algorithmica 71, pp. 775–796, doi:10.1007/s00453-013-9826-8. arXiv:1111.0475.

[39] Ognyan Oreshkov, Fabio Costa & Časlav Brukner (2012): *Quantum correlations with no causal order*. Nature Commun. 3(1), p. 1092, doi:10.1038/ncomms2076. arXiv:1105.4464.

[40] Ognyan Oreshkov & Christina Giarmatzi (2016): *Causal and causally separable processes*. New J. Phys. 18(9), p. 093020, doi:10.1088/1367-2630/18/9/093020. arXiv:1506.05449.

[41] Daniel R. Simon (1997): *On the Power of Quantum Computation*. SIAM J. Comput. 26(5), pp. 1474–1483, doi:10.1137/S0097539796298637.

[42] Márcio M. Taddei, Jaime Cariñe, Daniel Martínez, Tania García, Nayda Guerrero, Alastair A. Abbott, Mateus Araújo, Cyril Branciard, Esteban S. Gómez, Stephen P. Walborn, Leandro Aolita & Gustavo Lima (2021): *Computational advantage from the quantum superposition of multiple temporal orders of photonic gates*. PRX Quantum 2(1), p. 010320, doi:10.1103/PRXQuantum.2.010320. arXiv:2002.07817.

[43] Julian Wechs, Alastair A. Abbott & Cyril Branciard (2019): *On the definition and characterisation of multipartite causal (non) separability*. New J. Phys. 21(1), p. 013027, doi:10.1088/1367-2630/aaf352. arXiv:1807.10557.

[44] Julian Wechs, Hippolyte Dourdent, Alastair A. Abbott & Cyril Branciard (2021): *Quantum Circuits with Classical Versus Quantum Control of Causal Order*. PRX Quantum 2, p. 030335, doi:10.1103/PRXQuantum.2.030335. arXiv:2101.08796.

A Classical-deterministic supermaps and generalised deterministic query complexity

A.1 Classical-deterministic processes

In this appendix, we give some more details regarding classical-deterministic processes, which were introduced in Section 3. In particular, we provide detailed formal definitions of the notions of *induced functions*, *reduced process* and of the *link product* between a classical-deterministic process and a set of functions. We start by recalling the fixed-point characterisation of process functions (which we concisely refer to as classical-deterministic processes).

Proposition 2 ([19, 18])**.** *A function* $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ *is a* process function *if and only if, for all* $a \in \mathscr{P}$ *and all* $\vec{\mu} = (\mu_1, \ldots, \mu_T)$ *with* $\mu_k = \mathscr{I}_k \to \mathscr{O}_k$*,*

$$\exists!\, \vec{i} : w\big(a, \vec{\mu}(\vec{i})\big) = (\vec{i}, b), \tag{1}$$

with $\vec{i} = (i_1, \ldots, i_T) \in \bigtimes_{k=1}^{T} \mathscr{I}_k$ *and* $b \in \mathscr{F}$*.*

A classical-deterministic process is uniquely defined by its so-called induced functions. These functions single out the input of a given operation inserted into a "slot" of the process.

Definition 16 (Induced function [17])**.** Given a function $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$, the *induced functions* $w_k : \mathscr{P} \times \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \mathscr{I}_k$, for $1 \leq k \leq T$, are defined such that for all $a \in \mathscr{P}$ and $\vec{o} = (o_1 \ldots, o_T) \in \bigtimes_{k'=1}^{T} \mathscr{O}_{k'}$,

$$w_k(a, \vec{o}) = w(a, \vec{o})_{|k} = (i_1, \ldots, i_T, b)_{|k} = i_k. \tag{13}$$

Similarly, for $\mathscr{F}$, we define $w_F : \mathscr{P} \times \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \mathscr{F}$, such that for all $a \in \mathscr{P}$ and $\vec{o} = (o_1 \ldots, o_T) \in \bigtimes_{k'=1}^{T} \mathscr{O}_{k'}$,

$$w_F(a, \vec{o}) = w(a, \vec{o})_{|\mathscr{F}} = (i_1, \ldots, i_T, b)_{|\mathscr{F}} = b. \tag{14}$$

The induced functions w_k are the restrictions of w on the output spaces $\mathscr{I}_k$, while w_F is its restriction on $\mathscr{F}$.

Note that for w to be logically consistent, it is necessary that for $1 \leq k \leq T$, the input i_k received by the function μ_k is independent of its output o_k. If this is not the case, it opens the possibility for signalling "back in time" through w, from the output of μ_k to its input, which can lead to logical contradictions such as grandfather paradoxes [19]. At the level of the induced functions, this means that for any $\vec{o} = (o_1, \ldots, o_T) \in \bigtimes_{k'=1}^{T} \mathscr{O}_{k'}$, for any $a \in \mathscr{P}$ and any $1 \leq k \leq T$, $w_k(a, \vec{o})$ is independent of o_k, and we can thus unambiguously write $w_k(a, \vec{o}_{\setminus k})$, where $\vec{o}_{\setminus k} = (o_1, \ldots, o_{k-1}, o_{k+1}, \ldots, o_T)$.

The definition of classical-deterministic processes also implies that, by inserting an operation μ_k into the kth slot of a T-slot classical-deterministic process while leaving all the other slots empty, one obtains a $(T-1)$-slot classical-deterministic process $w^{|\mu_k}$, which we call a *reduced process*. One can similarly define a reduced process $w^{|a}$ by fixing a value $a \in \mathscr{P}$. More formally, we have the following definition.

Definition 17 (Reduced process [17])**.** Given a classical-deterministic process $w : \mathscr{P} \times \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \bigtimes_{k'=1}^{T} \mathscr{I}_{k'} \times \mathscr{F}$, for any $1 \leq k \leq T$ and any function $\mu_k : \mathscr{I}_k \to \mathscr{O}_k$, the *reduced process* $w^{|\mu_k} : \mathscr{P} \times \bigtimes_{\substack{k'=1 \\ k' \neq k}}^{T} \mathscr{O}_{k'} \to \bigtimes_{\substack{k'=1 \\ k' \neq k}}^{T} \mathscr{I}_{k'} \times \mathscr{F}$ is the $(T-1)$-slot classical-deterministic process uniquely defined by the following induced functions, for all $k' \neq k$, for all $\vec{o}_{\setminus k} = (o_1, \ldots, o_{k-1}, o_{k+1}, \ldots, o_T) \in \bigtimes_{\substack{l=1 \\ l \neq k}}^{T} \mathscr{O}_l$ and for all $a \in \mathscr{P}$:

$$w_{k'}^{|\mu_k}(a, \vec{o}_{\setminus k}) = w_{k'}\left(a, \left[\vec{o}_{\setminus k}, \mu_k\big(w_k(a, \vec{o}_{\setminus k})\big)\right]\right), \quad \text{and} \tag{15}$$

$$w_F^{|\mu_k}(a, \vec{o}_{\setminus k}) = w_F\left(a, \left[\vec{o}_{\setminus k}, \mu_k\big(w_k(a, \vec{o}_{\setminus k})\big)\right]\right), \tag{16}$$

where we use the shorthand notation $[\vec{o}_{\setminus k}, \mu_k\big(w_k(a, \vec{o}_{\setminus k})\big)] := \big(o_1, \ldots, o_{k-1}, \mu_k\big(w_k(a, \vec{o}_{\setminus k})\big), o_{k+1}, \ldots, o_T\big)$.

Similarly, one can define for any $a \in \mathscr{P}$ the reduced process $w^{|a} : \bigtimes_{k'=1}^{T} \mathscr{O}_{k'} \to \bigtimes_{k'=1}^{T} \mathscr{I}_{k'} \times \mathscr{F}$, defined by the following induced functions, for all $1 \leq k' \leq T$ and all $\vec{o} \in \bigtimes_{l=1}^{T} \mathscr{O}_l$:

$$w_{k'}^{|a}(\vec{o}) = w_{k'}(a, \vec{o}) \text{ and} \tag{17}$$

$$w_F^{|a}(\vec{o}) = w_F(a, \vec{o}). \tag{18}$$

We finish this appendix by defining a version of the so-called "link product" between a classical-deterministic process w and some operations $\vec{\mu}$.[6] This operation allows one to easily represent the resulting function from $\mathscr{P}$ to $\mathscr{F}$ obtained when w acts on the operations $\vec{\mu}$.

[6]The link product was defined in [28] to describe the composition of quantum maps (cf. Appendix C.1). By extension, it can be used to describe the composition of different functions as well as that of stochastic channels. For the purpose of this paper, however, we only explicitly define the composition of classical-deterministic processes with a set of functions.

Definition 18 (Link product between a classical-deterministic process and a list of functions)**.** The *link product*, denoted '$*$', between a classical-deterministic process $w : \mathscr{P} \times \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k \times \mathscr{F}$ and a list of functions $\vec{\mu} = (\mu_1, \ldots, \mu_T)$ is the function $w * \vec{\mu} : \mathscr{P} \to \mathscr{F}$ defined as, for any $a \in \mathscr{P}$ and writing $\vec{i}_a = (i_{a,1}, \ldots, i_{a,T})$ the fixed-point of w under the action of a and the functions $\vec{\mu} = (\mu_1, \ldots, \mu_T)$,

$$(w * \vec{\mu})(a) = w_F\big(\mu_1(i_{a,1}), \ldots, \mu_T(i_{a,T})\big). \tag{19}$$

A.2 Causally definite classical-deterministic processes and decision trees

In this appendix, we give a proof of Proposition 5, showing the equivalence of decision trees and causally definite classical-deterministic processes in the query model of computation.

Proposition 5. *In the query model, there exists a decision tree of depth T computing a Boolean function f if and only if there exists a T-slot causally definite classical-deterministic process computing f.*

Proof. We start by proving the necessary condition. Consider a binary decision tree of depth T computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Each internal node of the tree is labelled by an index $i \in \{1, \ldots, n\}$ specifying a bit x_i to query (i.e., the input i for the oracle), and its leaves are labelled by binary values specifying the output of the computation. Note that we can assume, without loss of generality, that the tree is complete. Every path from the root to a leaf of the tree can then be encoded as a binary string $(o_1, \ldots, o_T) \in \{0,1\}^T$ specifying which branch choice (i.e., left or right) to make at each internal node. The decision tree is then in one-to-one correspondence with the functions $N_l : \{0,1\}^{l-1} \to \{1, \ldots, n\}$ for $1 \leq l \leq T+1$ which specify the label of each node at depth $l-1$ reached by following the path $(o_1, \ldots, o_{l-1})$. Note that $N_1(\lambda)$, where λ is the empty string, is a constant (i.e., a function with no input) labelling the root (i.e., the node at depth 0), while $N_{T+1}(o_1, \ldots, o_T)$ specifies the label of the leaf reached at the end of the corresponding path. Defining recursively the inputs for the oracle specified by the decision tree as $j_1 := N_1(\lambda)$ and $j_l := N_l(x_{j_1}, \ldots, x_{j_{l-1}})$ for $l = 2, \ldots, T$, the output of the decision tree is thus $N_{T+1}(x_{j_1}, \ldots, x_{j_T})$.

The functions N_l can then be used to define a classical-deterministic process $w_{\text{Tree}} : \{0,1\}^T \to \{1, \ldots, n\}^T \times \{0,1\}$ via the induced functions $w_k(o_1, \ldots, o_T) := N_k(o_1, \ldots, o_{k-1})$, for all $1 \leq k \leq T$, and $w_F(o_1, \ldots, o_T) := N_{T+1}(o_1, \ldots, o_T)$. The logical consistency of this process follows from the structure of the decision tree, as for any set of operations $\vec{\mu} = (\mu_1, \ldots, \mu_T)$, the vector $\vec{i} = (i_1, \ldots, i_T)$ with $i_1 := N_1(\lambda)$ and, for $1 < k \leq T$, $i_k := N_k\big(\mu_1(i_1), \ldots, \mu_{k-1}(i_{k-1})\big)$ will be the unique fixed-point of w_{Tree} under $\vec{\mu}$. We thus see that, when $\mu_k = O_x$ for $k = 1, \ldots, T$ so that $\mu_k(i_k) = x_{i_k}$, we have precisely $w_F(\vec{\mu}(\vec{i})) = N_{T+1}(x_{i_1}, \ldots, x_{i_T})$ and hence w computes f.

For the sufficient condition, note that while a decision tree is always evaluated in a fixed sequential order, a causally definite classical-deterministic process can have dynamical order, making the mapping more difficult. Consider such process $w : \{0,1\}^T \to \{1, \ldots, n\}^T \times \{0,1\}$ with T slots that computes f. It follows from Definition 3 that there exists a $k_1 \in [T]$ such that the induced function w_{k_1} outputs a constant i_{k_1} and, for any operation μ_{k_1}, the reduced function $w^{|\mu_{k_1}}$ is causally definite. We can thus define the function $N_1(\lambda) := w_{k_1}(\vec{o}') = i_{k_1}$ specifying the root of the decision tree, where $\vec{o}' \in \{0,1\}^T$ can be taken to be an arbitrary string since w_{k_1} is constant. Note that the reduced process $w^{|\mu_{k_1}}$ depends only on the value $\mu_{k_1}(i_{k_1}) \in \{0,1\}$, so it is sufficient to consider the reduced processes $w^{|\mu_{k_1}^{[o_1]}}$ corresponding to the constant functions $\mu_{k_1}^{[o_1]}(\cdot) = o_1$ for $o_1 \in \{0,1\}$.

Following the recursive nature of Definition 3, let us then introduce the recursively defined notation $w^{|\mu_{k_1 \ldots k_l}^{[o_1 \ldots o_l]}} := (w^{|\mu_{k_1 \ldots k_{l-1}}^{[o_1 \ldots o_{l-1}]}})^{|\mu_{k_l}^{[o_l]}}$ for $l > 1$, where the k_l are those such that each $(w^{|\mu_{k_1 \ldots k_{l-1}}^{[o_1 \ldots o_{l-1}]}})_{k_l}$ is constant. We then define, for $1 < l \leq T$ and arbitrary $\vec{o}' \in \{0,1\}^{T-(l-1)}$, the functions

$$N_l(o_1,\dots o_{l-1}) := (w^{|\mu^{[o_1\dots o_{l-1}]}_{k_1\dots k_{l-1}}})_{k_l}(\vec{o}'), \tag{20}$$

and

$$N_{T+1}(o_1,\dots,o_T) = (w^{|\mu^{[o_1\dots o_{T-1}]}_{k_1\dots k_{T-1}}})_F(o_T), \tag{21}$$

where the reduced functions $(w^{|\mu^{[o_1\dots o_{l-1}]}_{k_1\dots k_{l-1}}})_{k_l}$ do not depend on $\vec{o}'$ since they are constant.

To see that the decision tree defined by the functions N_l for $1 \le l \le T+1$ indeed computes f, consider the unique fixed-point $\vec{i} = (i_1,\dots,i_T)$ of w acting on T copies of O_x and let $k_1,\dots,k_T$ be the order in which these operations are applied (which, except k_1, will in general depend on x). The output of the process is then

$$w_F(x_{i_1},\dots,x_{i_T}) = (w^{|\mu^{[x_{i_{k_1}}\dots x_{i_{k_{T-1}}}]}_{k_1\dots k_{T-1}}})_F(x_{i_{k_T}}) = N_{T+1}(x_{i_{k_1}},\dots,x_{i_{k_T}}), \tag{22}$$

which is precisely the output of the decision tree thus defined. □

A.3 Degree and certificate lower bounds

In this appendix, we prove the extended degree and certificate lower bounds for the generalised deterministic query complexity.

Proposition 6. *For any Boolean function f,* $\deg(f) \le D^{\mathrm{Gen}}(f)$.

Proof. Let $f:\{0,1\}^n \to \{0,1\}$ be a Boolean function with $D^{\mathrm{Gen}}(f) = T$. Then, by definition, there exists a T-slot classical-deterministic process $w:\{0,1\}^T \to \{1,\dots,n\}^T \times \{0,1\}$ such that, for all $x \in \{0,1\}^n$, there exists a unique $\vec{i}_x$ satisfying

$$w_F\big(O_x(i_{x,1}),\dots,O_x(i_{x,T})\big) = f(x). \tag{23}$$

We write the set of fixed-points for the strings x that evaluate to 1 as $I^{(1)}$; note that different strings can have the same fixed-point, e.g., if the value of f is independent of some bits in x. Then, for each fixed-point $\vec{i}_x \in I^{(1)}$ we define a monomial of degree T as

$$R_{\vec{i}_x}(y) = \prod_{\substack{k=1\\ O_x(i_{x,k})=1}}^{T} y_{i_{x,k}} \prod_{\substack{k=1\\ O_x(i_{x,k})=0}}^{T} (1 \oplus y_{i_{x,k}}). \tag{24}$$

These monomials have the property that, for any bitstring $y \in \{0,1\}^T$, $R_{\vec{i}_x}(y) = 1$ implies that $\vec{i}_x = \vec{i}_y$ and $f(x) = f(y) = 1$. This means that for all y there exist a unique monomial $R_{\vec{i}_x}$ such that $R_{\vec{i}_x}(y) = 1$ if $f(y) = 1$ and none if $f(y) = 0$. This implies that the polynomial

$$g(y) = \sum_{\vec{i}_x \in I^{[1]}} R_{\vec{i}_x}(y), \tag{25}$$

of degree T represents f, and therefore that $\deg(f) \le D^{\mathrm{Gen}}(f)$. □

The second result generalises the certificate complexity bound to the generalised deterministic query complexity.

Proposition 7. *For any Boolean function f, $C(f) \le D^{\mathrm{Gen}}(f)$.*

Proof. Let $f:\{0,1\}^n \to \{0,1\}$ a Boolean function with $D^{\text{Gen}}(f) = T$. Then, by definition, there exists a classical-deterministic process $w:\{0,1\}^T \to \{1,\dots,n\}^T \times \{0,1\}$ such that, for all x, there exists a unique $\vec{i}_x$ satisfying

$$w_F\big(O_x(i_{x,1}),\dots,O_x(i_{x,T})\big) = f(x). \tag{26}$$

This implies that the set $I_x = \{i_{x,1},\dots,i_{x,T}\}$ is a certificate for x. Indeed, suppose we have a y such that $y_{|I_x} = x_{|I_x}$, then $\vec{i}_x$ will also be a fixed-point under the queries to O_y, i.e.,

$$w_F\big(O_x(i_{x,1}),\dots,O_x(i_{x,T})\big) = w_F\big(O_y(i_{x,1}),\dots,O_y(i_{x,T})\big) = f(x). \tag{27}$$

By the unicity of fixed-points of classical-deterministic processes, we have $\vec{i}_y = \vec{i}_x$ and $f(x) = f(y)$. Therefore, for all x, $C(f,x) \leq |I_x| = T$ and hence $C(f) \leq T$. □

A.4 Composition of the generalised deterministic query complexity

In this appendix, we prove the recursive composition theorem for the generalised deterministic query complexity of Boolean functions.

Theorem 10. *For any Boolean function f and for any depth $l > 1$,*

$$D^{\text{Gen}}(f) = T \implies D^{\text{Gen}}(f^{(l)}) \leq T^l. \tag{7}$$

Proof. To prove this theorem, it is sufficient to show that from a classical-deterministic process computing f in T queries, one can build a classical-deterministic process that computes $f^{(l)}$ in T^l queries. We will construct this process by induction.

Let $w:\bigtimes_k^T \mathscr{O}_k \to \bigtimes_k^T \mathscr{I}_k \times \mathscr{F}$ be a classical-deterministic process computing f, (which thus necessarily has $\mathscr{O}_k = F = \{0,1\}$ and $\mathscr{I}_k = \{1,\dots,n\}$) and $w^{(l)}:\bigtimes_k^{T^l} \mathscr{O}_k \to \bigtimes_k^{T^l} \mathscr{I}_k^{(l)} \times \mathscr{F}$ a classical-deterministic process computing $f^{(l)}$, where $\mathscr{I}_k^{(l)} = \{1,\dots,n^l\}$.

We begin by constructing a classical-deterministic process $\tilde{w}^{(l)}$ that can compute $f^{(l)}$ on the n consecutive sequences of length n^l of a bitstring $x \in \{0,1\}^{n^{l+1}}$. The construction is based on $w^{(l)}$ which is extended with a global past $\mathscr{P}$ in which it receives as input an offset $a \in \mathscr{P} = \{1,\dots,n\}$. It is this offset that determines on which sequence of n^l bits the function $f^{(l)}$ is computed. Concretely the process is defined as $\tilde{w}^{(l)}:\mathscr{P} \times \bigtimes_{k=1}^{T^l} \mathscr{O}_k \to \bigtimes_{k=1}^{T^l} \mathscr{I}_k^{(l+1)} \times \mathscr{F}$ through its induced functions

$$\tilde{w}_k^{(l)}(a,o_1,\dots,o_{T^l}) = (a-1)n^l + w_k^{(l)}(o_1,\dots,o_{T^l}), \tag{28}$$

and

$$\tilde{w}_F^{(l)}(a,o_1,\dots,o_{T^l}) = w_F(o_1,\dots,o_{T^l}). \tag{29}$$

We now show that $\tilde{w}^{(l)}$ is logically consistent, and hence a valid classical-deterministic process. For any $a \in \mathscr{P}$ and for any set of functions $\{\tilde{\mu}_k\}_{k=1}^{T^l}$ with $\tilde{\mu}_k : \mathscr{I}_k^{(l+1)} \to \mathscr{O}_k$, we define the functions $\{\mu_{a,k}\}_{k=1}^{T^l}$ with $\mu_{a,k} : \mathscr{I}_k^{(l)} \to \mathscr{O}_k$ such that $\mu_{a,k}(i) = \tilde{\mu}_k\big((a-1)n^l + i\big)$. Now, denote $\vec{i} = (i_1,\dots,i_{T^l})$ the unique fixed-point of $w^{(l)}$ under the functions $\{\mu_{a,k}\}_{k=1}^{T^l}$, then the vector $\vec{j} = (j_1,\dots,j_{T^l})$ with $j_k = (a-1)n^l + i_k$ will be a fixed-point of $\tilde{w}^{(l)}$ under a and the functions $\{\tilde{\mu}_k\}_{k=1}^{T^l}$:

$$\begin{aligned}
\tilde{w}_k^{(l)}(a,j_1,\dots,j_{T^l}) &= (a-1)n^l + w_k^{(l)}\big(\tilde{\mu}_1(j_1),\dots,\tilde{\mu}_{T^l}(j_{T^l})\big), &(30)\\
&= (a-1)n^l + w_k^{(l)}\big(\mu_{a,1}(i_1),\dots,\mu_{a,T^l}(i_{T^l})\big), &(31)\\
&= (a-1)n^l + i_k, &(32)\\
&= j_k. &(33)
\end{aligned}$$

For the unicity of the fixed-point, suppose that $\vec{j}' = (j'_1, \ldots, j'_{T^l})$ is a fixed-point of $\tilde{w}^{(l)}$ under $a \in \mathscr{P}$ and the functions $\{\tilde{\mu}_k\}_{k=1}^{T^l}$, then there exists an $\vec{i}' = (i'_1, \ldots, i'_{T^l})$ such that

$$\tilde{w}_k^{(l)}(a, j'_1, \ldots, j'_{T^l}) = (a-1)n^l + w_k^{(l)}\big(\tilde{\mu}_1(j'_1), \ldots, \tilde{\mu}_{T^l}(j'_{T^l})\big), \tag{34}$$

$$= (a-1)n^l + i'_k, \tag{35}$$

$$= j'_k. \tag{36}$$

Thus, $\vec{i}' = (i'_1, \ldots, i'_{T^l})$ will be a fixed-point of $w^{(l)}$ under the functions $\{\mu_{a,k}\}_{k=1}^{T^l}$, and because $w^{(l)}$ is logically consistent we have, by definition, $\vec{i}' = \vec{i}$, implying $\vec{j}' = \vec{j}$.

Furthermore, notice that for any $x \in \{0,1\}^{n^{(l+1)}}$, with T^l copies of the oracle O_x and for any $a \in \mathscr{P}$

$$\tilde{w}_F^{(l)}\big(a, O_x(j_1), \ldots O_x(j_{T^l})\big) = f^{(l)}(x_{(a-1)n^l+1}, \ldots, x_{an^l}), \tag{37}$$

meaning that $\tilde{w}^{(l)}$ can compute f on any of the n consecutive sequence of n^l bits of $x \in \{0,1\}^{n^{l+1}}$.

The last step is to build a classical-deterministic process $w^{(l+1)}$ from $\tilde{w}^{(l)}$ that computes $f^{(l+1)}$ in T^{l+1} queries. Consider a list of T^{l+1} operations $\vec{\mu} = (\vec{\mu}_1, \ldots, \vec{\mu}_T)$, where each $\vec{\mu}_k = (\vec{\mu}_{k,1}, \ldots, \vec{\mu}_{k,T^l})$ and notice that for all $1 \leq k \leq T$, the function $\tilde{w}^{(l)} * \vec{\mu}_k$ is a function from $\mathscr{P}$ to $\mathscr{F}$, which are isomorphic, respectively, to the $\mathscr{I}_k$'s and $\mathscr{O}_k$'s. Let us then consider the value

$$w * (\tilde{w}^{(l)} * \vec{\mu}_1, \ldots, \tilde{w}^{(l)} * \vec{\mu}_T) \in \mathscr{F}. \tag{38}$$

As such, the function $w^{(l+1)} : \bigtimes_k^{T^{l+1}} \mathscr{O}_k \to \bigtimes_k^{T^{l+1}} \mathscr{O}_k \times \mathscr{F}$ is defined for any T^{l+1} operations $\vec{\mu}$ as

$$w^{(l+1)} * \vec{\mu} = w * \big(\tilde{w}^{(l)} * \vec{\mu}_1, \ldots, \tilde{w}^{(l)} * \vec{\mu}_T\big), \tag{39}$$

and is a classical-deterministic process whose logical consistency follows from that of w and $\tilde{w}^{(l)}$.

It follows from Eq. (37) and because w computes f that, for any $x \in \{0,1\}^{l+1}$ and by writing $\vec{O}_x = (\underbrace{O_x, \ldots, O_x}_{T^l})$, we have

$$w * (\underbrace{\tilde{w}^{(l)} * \vec{O}_x, \ldots, \tilde{w}^{(l)} * \vec{O}_x}_{\mathrm{T}}) = f\big(f^{(l)}(x_1, \ldots, x_{n^l}), \ldots, f^{(l)}(x_{(n-1)n^l+1} \ldots x_{n^l})\big) \tag{40}$$

$$= f^{(l+1)}(x), \tag{41}$$

thus completing the proof. □

B Properties of the Boolean function f_{6c}

In this appendix, we prove different properties of the Boolean function f_{6c}, regarding its certificate complexity and its standard and generalised deterministic query complexity. In particular, we show that causal indefiniteness reduces the number of queries that are necessary to compute f_{6c}. The 6-bit Boolean function is represented by the following degree three polynomial.

$$f_{6c}(x_1, \ldots, x_6) = x_4(1-x_2)x_3 + x_5(1-x_3)x_1 + x_6(1-x_1)x_2 + x_1x_2x_3, \tag{42}$$

$$= x_4(1 \oplus x_2)x_3 \oplus x_5(1 \oplus x_3)x_1 \oplus x_6(1 \oplus x_1)x_2 \oplus x_1x_2x_3, \tag{43}$$

where $\oplus$ denotes addition modulo 2. These two ways of writing the polynomial representing f_{6c} are equivalent, and the multilinear polynomial representation (42) is unique, so $\deg(f_{6c}) = 3$. The truth table for f_{6c} is also shown in Table 1.

Table 1: Simplified truth table of the Boolean function f_{6c}.

x_1	x_2	x_3	$f_{6c}(x)$
0	0	0	0
1	0	0	x_5
0	1	0	x_6
0	0	1	x_4
1	1	0	x_5
1	0	1	x_4
0	1	1	x_6
1	1	1	1

We start by proving the following result on the certificate complexity of f_{6c}.

Proposition 19. $C(f_{6c}) = 3$.

Proof. The proof that $C(f_{6c}) = 3$ comes from observing Eq. (43) and verifying that, for any value of $x \in \{0,1\}^6$, the value of $f_{6c}(x)$ depends only on three bits of the input.

Indeed, when x_1, x_2 and x_3 are all equal to zero or to one, the value of f_{6c} is independent of x_4, x_5 and x_6, meaning that $\{1,2,3\}$ is a certificate of size three. When $x_2 = 0$ and $x_3 = 1$, the value of f_{6c} is independent of x_1, x_5 and x_6, and $\{2,3,4\}$ is a size three certificate. When $x_3 = 0$ and $x_1 = 1$, $\{1,3,5\}$ is a certificate. Lastly, when $x_1 = 0$ and $x_2 = 1, \{1,2,6\}$ is a certificate. In each case, it is clear that one can't give a smaller certificate, as the output depends on each bit in the certificate. For every $x \in \{0,1\}^6$ we thus have $C(f_{6c}, x) = 3$, and hence $C(f_{6c}) = 3$. □

We also prove the following result regarding the deterministic query complexity of f_{6c}.

Proposition 20. $D(f_{6c}) = 4$.

Proof. Let us first note that one can indeed compute f_{6c} with 4 queries. Suppose one first queries x_1, x_2 and x_3. If one obtains $x_1 = x_2 = x_3$, then the polynomial representation (43) reduces to $f_{6c}(x) = x_1x_2x_3$ and we are done. Otherwise, a single further query to either x_4, x_5 or x_6 is necessary, depending on whether $(x_2, x_3) = (0,1)$, $(x_1, x_3) = (1,0)$ or $(x_1, x_2) = (0,1)$, respectively.

To see that one cannot compute f_{6c} sequentially with 3 queries, let us first suppose that the first query is to x_4, x_5 or x_6. But then if $x_1 = x_2 = x_3$ three more queries will be needed to evaluate the monomial $x_1x_2x_3$, so one of these bits must be queried first if one is to use only three queries. Imagine then, without loss of generality, that the first query is to x_1. If $x_1 = 0$, then Eq. (43) reduces to $f_{6c}(x) = x_4(1 \oplus x_2)x_3 \oplus x_6x_2$, and we see that, whichever of x_2, x_3, x_4, x_6 is chosen to be queried second, at least two further queries are needed in the worst case, giving a total of four queries, and hence no sequential adaptive algorithm can compute f_{6c} in three queries for all x. □

These results show that f_{6c} satisfies the two conditions identified in Section 4.1 that make a function a candidate for exhibiting a query complexity advantage from causal indefiniteness, namely that $\deg(f) < D(f)$ and $C(f) < D(f)$. We now prove Proposition 8, showing that such a query complexity separation is obtained for f_{6c}.

Proposition 8. $D^{\mathrm{Gen}}(f_{6c}) = 3$.

Proof. To compute the Boolean function f_{6c} in three queries, we take as a starting point the Lugano process $w_{\mathrm{Lugano}} : \{0,1\}^3 \to \{0,1\}^3$ defined by its induced functions as in Eq. (2). In order to use the Lugano process to query several copies of the query oracle $O_x : \{1,\dots,6\} \to \{0,1\}$, we slightly modify the process so that the slots receive 6-dimensional inputs and we add a non-trivial future space $\mathscr{F} = \{0,1\}$, giving a function $\overline{w}_{\mathrm{Lugano}} : \{0,1\}^3 \to \{1,\dots,6\}^3 \times \{0,1\}$. $\overline{w}_{\mathrm{Lugano}}$ is defined by its induced functions which are, for all $(o_1,o_2,o_3) \in \{0,1\}^3$ and any $1 \leq k \leq 3$,

$$\overline{w}_k(o_1,o_2,o_3) = k + 3 \cdot w_k(o_1,o_2,o_3), \tag{44}$$

where w_k are the induced functions of the standard Lugano process as defined in Eq. (2), and

$$\overline{w}_F(o_1,o_2,o_3) = o_1(1 \oplus o_2)o_3 \oplus o_2(1 \oplus o_3)o_1 \oplus o_3(1 \oplus o_1)o_2 \oplus o_1o_2o_3. \tag{45}$$

Before showing that $\overline{w}_{\mathrm{Lugano}}$ computes f_{6c} in three queries, we first prove that it is logically consistent.

Consider three arbitrary functions $(\overline{\mu}_k)_{k\in\{1,2,3\}}$, with $\overline{\mu}_k : \{1,\dots,6\} \to \{0,1\}$, and define the three functions $(\mu_k)_{k\in\{1,2,3\}}$, with $\mu_k : \{0,1\} \to \{0,1\}$, such that $\mu_k(i) = \overline{\mu}_k(k+3i)$. Because w_{Lugano} is a classical-deterministic process, it has a unique fixed-point $(i_1,i_2,i_3) \in \{0,1\}^3$ under the action of the μ_k's. Then $(j_1,j_2,j_3) \in \{1,6\}^3$ with $j_k = k+3i_k$ will be a fixed-point of $\overline{w}_{\mathrm{Lugano}}$ under the actions of the $\overline{\mu}_k$'s. Indeed,

$$\begin{aligned}
\overline{w}_k\big(\overline{\mu}_1(j_1),\overline{\mu}_2(j_2),\overline{\mu}_3(j_3)\big) &= k + 3 \cdot w_k\big(\overline{\mu}_1(j_1),\overline{\mu}_2(j_2),\overline{\mu}_3(j_3)\big), &(46)\\
&= k + 3 \cdot w_k\big(\mu_1(i_1),\mu_2(i_2),\mu_3(i_3)\big), &(47)\\
&= k + 3 \cdot i_k, &(48)\\
&= j_k. &(49)
\end{aligned}$$

To prove the unicity, suppose that $(j'_1,j'_2,j'_3) \in \{0,1\}^6$ is another fixed-point of $\overline{w}_{\mathrm{Lugano}}$ under the functions $(\overline{\mu}_k)_{k\in\{1,2,3\}}$. Then there exist $(i'_1,i'_2,i'_3) \in \{0,1\}^3$ such that

$$\begin{aligned}
\overline{w}_k\big(\overline{\mu}_1(j'_1),\overline{\mu}_2(j'_2),\overline{\mu}_3(j'_3)\big) &= k + 3 \cdot w_k\big(\overline{\mu}_1(j'_1),\overline{\mu}_2(j'_2),\overline{\mu}_3(j'_3)\big), &(50)\\
&= k + 3 \cdot i'_k, &(51)\\
&= j'_k. &(52)
\end{aligned}$$

This shows that (i'_1,i'_2,i'_3) must also be a fixed-point of w_{Lugano} under the functions $\{\mu_k\}_{k\in\{1,2,3\}}$, but because the Lugano is logically consistent this fixed-point is unique, and we have for $1 \leq k \leq 3$, $i'_k = i_k$ and therefore $j'_k = j_k$.

Finally, to show that $\overline{w}_{\mathrm{Lugano}}$ computes f_{6c}, we verify that for any $x \in \{0,1\}^6$ and writing $\vec{O}_x = (\underbrace{O_x,O_x,O_x}_{3})$, we have

$$\overline{w}_{\mathrm{Lugano}} * \vec{O}_x = f(x). \tag{53}$$

To see this, consider the first two columns of Table 2 which show the unique fixed-points of $\overline{w}_{\mathrm{Lugano}}$ for different values of x, under three copies of the query oracle O_x. These depend only on the values of x_1,x_2 and x_3 and one can verify that they are the unique fixed-points of $\overline{w}_{\mathrm{Lugano}}$ by checking that, for each row and for any $1 \leq k \leq 3$, $\overline{w}_k\big(O_x(j_1),O_x(j_2),O_x(j_3)\big) = \overline{w}_k(x_{j_1},x_{j_2},x_{j_3}) = j_k$.

The last two columns of Table 2 give the different outputs of the queries for the different fixed-points together with the value obtained in $\mathscr{F}$, which gives the output of the computation. We see that together,

the first and last columns correspond exactly to the truth table of f_{6c} in Table 1. This shows that the classical-deterministic process $\overline{w}_{\text{Lugano}}$ computes f_{6c} in three queries to O_x. Because $C(f_{6c}) = 3$, this implies that $D^{\text{Gen}}(f_{6c}) = 3$. □

Table 2: Fixed-points and outputs of the classical-deterministic process $\overline{w}_{\text{Lugano}}$ on three copies of the query oracle O_x, for all $x \in \{0,1\}^6$. In the table, $j_k = \overline{w}_k(o_1,o_2,o_3)$ and $o_k = O_x(j_k) = x_{j_k}$. The last column follows from Eq. (45).

x_1	x_2	x_3	j_1	j_2	j_3	o_1	o_2	o_3	$\overline{w}_F(o_1,o_2,o_3)$
0	0	0	1	2	3	0	0	0	0
1	0	0	1	5	3	1	x_5	0	x_5
0	1	0	1	2	6	0	1	x_6	x_6
0	0	1	4	2	3	x_4	0	1	x_4
1	1	0	1	5	3	1	x_5	0	x_5
1	0	1	4	2	3	x_4	0	1	x_4
0	1	1	1	2	6	0	1	x_6	x_6
1	1	1	1	2	3	1	1	1	1

C Process functions, process matrices and causal separability

C.1 Process functions as process matrices

We begin by discussing the restriction of the process matrix framework to classical-deterministic maps, and providing a short proof of the equivalence to process functions in this setting [19, 18]. We fix a canonical computational "classical" basis, and consider an operation (a channel, CP map, or a process) to be classical if its Choi matrix is diagonal in that basis. For clarity, we will denote in this appendix the Choi matrices of such operations with a bar, e.g. as $\overline{\mathsf{M}}$. Moreover, an operation is considered deterministic if it transforms pure classical states into pure classical states, in which case its Choi matrix contains only zeros and ones, and will generically be denoted in this appendix with a tilde, e.g. as $\widetilde{\mathsf{M}}$. A classical-deterministic channel, for example, corresponds to a function $\mu : \mathscr{I} \to \mathscr{O}$, and can be written in the Choi picture as the matrix $\widetilde{\mathsf{M}}_\mu = \sum_{i\in\mathscr{I}} |i\rangle\langle i|^I \otimes |\mu(i)\rangle\langle\mu(i)|^O \in \mathscr{L}(\mathscr{H}^{IO})$. In a slight abuse of language, we will sometimes identify a channel with its Choi matrix, the two being isomorphic.

Let us introduce the notation $\mathscr{H}^{(IO)_k} = \mathscr{H}^{I_k O_k} = \mathscr{H}^I \otimes \mathscr{H}^O$ and $\mathscr{H}^{(IO)_{[T]}} = \bigotimes_{k\in[T]} \mathscr{H}^{(IO)_k}$ with $[T] := \{1,\dots,T\}$, as well as $d_{O_k} = \dim(\mathscr{H}^{O_k})$, $d_{I_k} = \dim(\mathscr{H}^{I_k})$, $d_{O_{[T]}} = \prod_{k\in[T]} d_{O_k}$ and $d_{I_{[T]}} = \prod_{k\in[T]} d_{I_k}$. We then begin by defining classical-deterministic process matrices, where without loss of generality we take the global past $\mathscr{H}^P$ and global future $\mathscr{H}^F$ to be trivial and therefore omit them [43]. We will make use of the link product [26, 28], which allows the composition of quantum maps, potentially over a subset of their input/output systems, to be computed directly in the Choi picture. The link product is defined for any matrices $\mathsf{M}^{XY} \in \mathscr{L}(\mathscr{H}^{XY})$ and $\mathsf{N}^{YZ} \in \mathscr{L}(\mathscr{H}^{YZ})$, as $\mathsf{M}^{XY} * \mathsf{N}^{YZ} = \text{Tr}_Y\left[(\mathsf{M}^{XY} \otimes \mathbb{1}^Z)^{\mathsf{T}_Y} (\mathbb{1}^X \otimes \mathsf{N}^{YZ})\right] \in \mathscr{L}(\mathscr{H}^{XZ})$, where $\cdot^{\mathsf{T}_Y}$ denotes the partial transpose over the Hilbert space $\mathscr{H}^Y$ with respect to the computational basis.

Definition 21 (T-slot classical-deterministic process matrix)**.** A T-slot classical-deterministic process matrix is a diagonal positive semidefinite matrix $\widetilde{W} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ whose diagonal is composed only

of zeros and ones, and such that for any T classical-deterministic channels $(\widetilde{\mathsf{M}}_1,\ldots,\widetilde{\mathsf{M}}_T)$ with $\widetilde{\mathsf{M}}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, we have $\widetilde{W} * (\widetilde{\mathsf{M}}_1 \otimes \cdots \otimes \widetilde{\mathsf{M}}_T) = 1$.[7]

Note that it is sufficient to require that classical-deterministic process matrices be well normalised on the set of classical-deterministic channels, as for any T *quantum* channels $(\mathsf{M}_1,\ldots,\mathsf{M}_T)$, with $\mathsf{M}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, we have $\widetilde{W} * (\mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_T) = \widetilde{W} * (\overline{\mathsf{M}}_1 \otimes \cdots \otimes \overline{\mathsf{M}}_T)$, with the $\overline{\mathsf{M}}_k = \boldsymbol{\pi}_k(\mathsf{M}_k)$ being *classical* channels, where $\boldsymbol{\pi}_k$ is the projector onto the classical computational basis, i.e. $\boldsymbol{\pi}_k(\mathsf{M}_k) := \sum_i |i\rangle\langle i|\,\mathsf{M}_k\,|i\rangle\langle i|$. Since any classical channel can be written as a convex combination of *classical-deterministic* channels, it follows from linearity that $\widetilde{W} * (\mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_T) = 1$, and $\widetilde{W}$ is thus a valid process matrix.

A classical-deterministic process matrix $\widetilde{W}$ can also be seen as the Choi matrix of a classical-deterministic channel from $\mathscr{L}(\mathscr{H}^{O_{[T]}})$ to $\mathscr{L}(\mathscr{H}^{I_{[T]}})$.

Lemma 22. *For any T-slot classical-deterministic process matrix $\widetilde{W} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ there exists a function $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k$ with $\mathscr{O}_k = \{0,\ldots,d_{O_k}-1\}$ and $\mathscr{I}_k = \{0,\ldots,d_{I_k}-1\}$, such that*

$$\widetilde{W} = \sum_{\vec{k}\in\mathscr{O}_{[T]}} |\vec{k}\rangle\langle\vec{k}|^{O_{[T]}} \otimes |w(\vec{k})\rangle\langle w(\vec{k})|^{I_{[T]}}, \tag{54}$$

where $\mathscr{O}_{[T]} = \bigtimes_{k\in[T]} \mathscr{O}_k$ and $|\vec{k}\rangle\langle\vec{k}| = |k_1\ldots k_T\rangle\langle k_1\ldots k_T|$.

We note also that the characterisation of process functions given in Section 3 simplifies slightly in the case considered here, in which $\mathscr{P}$ and $\mathscr{F}$ are trivial and therefore omitted.

Proposition 23 (Process function without $\mathscr{P}$ or $\mathscr{F}$ [18])**.** *A function $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k$ is called a process function if and only if, for all T functions $\vec{\mu} = (\mu_1,\ldots,\mu_T)$ with $\mu_k = \mathscr{I}_k \to \mathscr{O}_k$,*

$$\exists!\vec{i} : w\big(\vec{\mu}(\vec{i})\big) = \vec{i}, \tag{55}$$

with $\vec{i} = (i_1,\ldots,i_T) \in \bigtimes_{k=1}^{T} \mathscr{I}_k$, and $\vec{\mu}(\vec{i}) = (\mu_1(i_1),\ldots,\mu_T(i_T))$

We can now show the equivalence between process functions and classical-deterministic process matrices, by noticing that the unique fixed-point condition of Proposition 23 is equivalent to the normalisation constraint of Definition 21. This result was first shown in [19] and later in [18], but we give an explicit self-contained proof here for completeness, as it will prove instructive for the following section.

Proposition 24. *A function $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k$, with the spaces $\mathscr{O}_k = \{0,\ldots,d_{O_k}-1\}$ and $\mathscr{I}_k = \{0,\ldots,d_{I_k}-1\}$ is a process function if and only if the matrix $\widetilde{W} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ defined as*

$$\widetilde{W} = \sum_{\vec{k}\in\mathscr{O}_{[T]}} |\vec{k}\rangle\langle\vec{k}|^{O_{[T]}} \otimes |w(\vec{k})\rangle\langle w(\vec{k})|^{I_{[T]}} \tag{56}$$

is a T-slot classical-deterministic process matrix.

Proof. Let $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k$, with $\mathscr{O}_k = \{0,\ldots,d_{O_k}-1\}$ and $\mathscr{I}_k = \{0,\ldots,d_{I_k}-1\}$ be a process function, and define $\widetilde{W} = \sum_{\vec{k}\in\mathscr{O}_{[T]}} |\vec{k}\rangle\langle\vec{k}|^{O_{[T]}} \otimes |w(\vec{k})\rangle\langle w(\vec{k})|^{I_{[T]}} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$. By construction $\widetilde{W}$ is a diagonal positive semidefinite matrix, whose diagonal is composed only of zeros and ones. Now consider

[7] This constraint of being well normalised on the set of classical-deterministic channels can also be framed in term of a normalisation constraint and a projection constraint, as it is done for general process matrix in Section 7.1 [39, 9, 43]

any T classical-deterministic channels $(\widetilde{\mathsf{M}}_{\mu_1},\dots,\widetilde{\mathsf{M}}_{\mu_T})$ labelled with their corresponding functions $\mu_k : \mathscr{O}_k \to \mathscr{I}_k$, such that $\widetilde{\mathsf{M}}_{\mu_k} = \sum_{i\in\mathscr{I}_k} |i\rangle\langle i|^{I_k} \otimes |\mu_k(i)\rangle\langle\mu_k(i)|^{O_k}$. We have, writing $\mathscr{I}_{[T]} = \bigtimes_{k\in[T]} \mathscr{I}_k$,

$$\widetilde{W} * (\widetilde{\mathsf{M}}_1 \otimes \cdots \otimes \widetilde{\mathsf{M}}_T) = \mathrm{Tr}\Big(\sum_{\vec{k}\in\mathscr{O}_{[T]}} |\vec{k}\rangle\langle\vec{k}|^{O_{[T]}} \otimes |w(\vec{k})\rangle\langle w(\vec{k})|^{I_{[T]}} \cdot \sum_{\vec{i}\in\mathscr{I}_{[T]}} |\vec{i}\rangle\langle\vec{i}|^{I_{[T]}} \otimes |\vec{\mu}(\vec{i})\rangle\langle\vec{\mu}(\vec{i})|^{O_{[T]}} \Big) \tag{57}$$

$$= \mathrm{Tr}\Big(\sum_{\substack{\vec{i}\in\mathscr{I}_{[T]} \\ w\left(\vec{\mu}(\vec{i})\right)=\vec{i}}} |\vec{\mu}(\vec{i})\rangle\langle\vec{\mu}(\vec{i})|^{O_{[T]}} \otimes |w\big(\vec{\mu}(\vec{i})\big)\rangle\langle w\big(\vec{\mu}(\vec{i})\big)|^{I_{[T]}} \Big) \tag{58}$$

$$= 1, \tag{59}$$

where the last line follows from the existence of a unique $\vec{i}$ such that $w\big(\vec{\mu}(\vec{i})\big) = \vec{i}$. Conversely, if we assume that $\widetilde{W}$ is a valid classical-deterministic process matrix so that $\widetilde{W} * (\widetilde{\mathsf{M}}_1 \otimes \cdots \otimes \widetilde{\mathsf{M}}_T) = 1$, we find in the same way that there must exist a unique $\vec{i}$ such that $w\big(\vec{\mu}(\vec{i})\big) = \vec{i}$. □

C.2 Causal separability in classical-deterministic settings

In Definition 3, we proposed a definition of *causally definite* classical-deterministic processes to describe processes that connect the various operations they take as input in a well-defined causal order. This definition is analogous in spirit to that of *causal separability* for process matrices (or, equivalently, quantum supermaps), as defined in [43]. The goal of this appendix is to demonstrate that our definition, formulated in the process function formalism, is indeed consistent with the more general notion of causal separability defined for process matrices when restricted to classical-deterministic process matrices.

As in the discussion of classical process matrices in the previous section, we consider without loss of generality process matrices $W \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ that are defined with trivial past and future space $\mathscr{H}^P$ and $\mathscr{H}^F$. We begin by recalling the definition of causal separability for process matrices.

Definition 25 (Causal separability of process matrices [43])**.** For $T = 1$, any T-slot process matrix is causally separable. For $T \geq 2$, a T-slot process matrix W is said to be causally separable if and only if, for any extension $\mathscr{H}^{I'_{[T]}}$ of the slots' input spaces and any ancillary quantum state $\rho \in \mathscr{L}(\mathscr{H}^{I'_{[T]}})$, $W \otimes \rho$ can be decomposed as

$$W \otimes \rho = \sum_{k\in[T]} q_k W^{\rho}_{(k)}, \tag{60}$$

with $q_k \geq 0$, $\sum_k q_k = 1$, and where for each k, $W^{\rho}_{(k)} \in \mathscr{L}(\mathscr{H}^{(II'O)_{[T]}})$ is a T-slot process matrix compatible with the kth operation acting first, and is such that for any CP map $\mathsf{M}_k \in \mathscr{L}(\mathscr{H}^{(II'O)_k})$, the conditional $(T-1)$-slot process matrix $(W^{\rho}_{(k)})_{|\mathsf{M}_k} := W^{\rho}_{(k)} * \mathsf{M}_k$ is itself causally separable (up to normalisation).

A process matrix that cannot be decomposed as in Definition 25 is said to be causally nonseparable. In this definition, the extension with a state $\rho \in \mathscr{L}(\mathscr{H}^{I'_{[T]}})$ is important to rule out the possibility of "activation" of causal indefiniteness, wherein a process matrices that can be decomposed as in Eq. (60) when extended with any separable state can become causally nonseparable when extended with some entangled state [40, 43]. The following proposition shows how this definition is simplified when the process we consider is classical, i.e., when its process matrix is diagonal in the computational basis.

Proposition 26 (Causal separability of classical process matrices)**.** *For $T = 1$, any T-slot classical process matrix is causally separable. For $T \geq 2$, a T-slot classical process matrix $\overline{W}$ is causally separable if and only if it can be decomposed as*

$$\overline{W} = \sum_{k\in[T]} q_k \overline{W}_{(k)}, \tag{61}$$

with $q_k \geq 0$, $\sum_k q_k = 1$, and where for each k, $\overline{W}_{(k)} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ is a classical process matrix compatible with the kth operation acting first, and is such that for any classical CP map $\overline{\mathsf{M}}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, the conditional $(T-1)$-slot classical process matrix $(\overline{W}_{(k)})_{|\overline{\mathsf{M}}_k} := \overline{W}_{(k)} * \overline{\mathsf{M}}_k$ is itself causally separable (up to normalisation).

Proof. Clearly, if $\overline{W}$ is causally separable then it has a decomposition of the form of Eq. (61) with the desired properties since Definition 25 includes the case of a trivial ancillary system and requires the recursive condition to hold for all CP maps M_k, which includes classical CP maps $\overline{\mathsf{M}}_k$. We hence focus on proving the converse statement.

We will prove that the simplified conditions above imply causal separability for classical process matrices inductively. Clearly, for $T = 1$ this is the case, since any 1-slot process matrix, classical or not, is causally separable.

Let us then assume that any $(T-1)$-slot classical process matrix with a decomposition following Eq. (61) with the required properties is causally separable. Consider then a classical process matrix $\overline{W}$ that can be decomposed following Eq. (61) and where for every $k \in [T]$ and any classical CP map $\overline{\mathsf{M}}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, $\overline{W}_{(k)} * \overline{\mathsf{M}}_k$ is a causally separable $(T-1)$-slot classical process matrix. To show that $\overline{W}$ is itself causally nonseparable, we will then show that for any extension $\mathscr{H}^{I'_{[T]}}$ of the parties' Hilbert spaces, any quantum state $\rho \in \mathscr{L}(\mathscr{H}^{I'_{[T]}})$, and any (not necessarily classical) CP map $\mathsf{M}_k \in \mathscr{L}(\mathscr{H}^{(II'O)_k})$, the extended process $\overline{W} \otimes \rho = \sum_{k\in[T]} q_k(\overline{W}_{(k)} \otimes \rho)$ indeed has the property that for any k and $\mathsf{M}_k \in \mathscr{L}(\mathscr{H}^{(II'O)_k})$, $(\overline{W}_{(k)} \otimes \rho) * \mathsf{M}_k$ is causally separable.

Let us first note that because $\overline{W}_{(k)}$ is diagonal it satisfies

$$\overline{W}_{(k)} = \sum_i \left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{(IO)_{[T]\setminus k}}\right) \overline{W}_{(k)} \left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{(IO)_{[T]\setminus k}}\right), \tag{62}$$

where $\Pi_i^{(IO)_k} = |i\rangle\langle i|^{(IO)_k}$ and the $|i\rangle^{(IO)_k}$ are the computational basis vectors of $\mathscr{H}^{(IO)_k}$. It then follows, using the definition of the link product, that

$$(\overline{W}_{(k)} \otimes \rho) * \mathsf{M}_k = \sum_i \left(\left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{(IO)_{[T]\setminus k}}\right) \overline{W}_{(k)} \left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{(IO)_{[T]\setminus k}}\right) \otimes \rho \right) * \mathsf{M}_k \tag{63}$$

$$= (\overline{W}_{(k)} \otimes \rho) * \sum_i \left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{I'_k}\right) \mathsf{M}_k \left(\Pi_i^{(IO)_k} \otimes \mathbb{1}^{I'_k}\right). \tag{64}$$

Writing $\mathsf{M}_k = \sum_{ij} |i\rangle\langle j|^{(IO)_k} \otimes \nu_{ij}^{I'_k}$, we note that since M_k is the Choi matrix of a CP map it is positive semidefinite, and hence the matrices $\nu_{ii} \in \mathscr{L}(\mathscr{H}^{I'_k})$ are also positive semidefinite. We then have, following Eq. (64),

$$(\overline{W}_{(k)} \otimes \rho) * \mathsf{M}_k = (\overline{W}_{(k)} \otimes \rho) * \sum_i |i\rangle\langle i|^{(IO)_k} \otimes \nu_{ii}^{I'_k} \tag{65}$$

$$= \sum_i (\overline{W}_{(k)} * |i\rangle\langle i|^{(IO)_k}) \otimes (\rho * \nu_{ii}^{I'_k}) \tag{66}$$

$$= \sum_i r_i (\overline{W}_{(k)} * |i\rangle\langle i|^{(IO)_k}) \otimes \rho_i \tag{67}$$

$$= \sum_i \frac{r_i}{r} (\overline{W}_{(k)} * \overline{\mathsf{M}}_k^{(i)}) \otimes \rho_i, \tag{68}$$

where $r_i := \mathrm{Tr}\left(\rho * \nu_{ii}^{I'_k}\right)$, $\rho_i := \frac{1}{r_i}\rho * \nu_{ii}^{I'_k} \in \mathscr{L}(\mathscr{H}^{I'_{[T]\setminus k}})$ are ancillary states for the remaining $T-1$ parties, $r := \sum_i r_i$, and the $\overline{\mathsf{M}}_k^{(i)} := r\,|i\rangle\langle i|^{(IO)_k}$ are classical (diagonal) CP maps. By the inductive assumption, for each i, the $(T-1)$-slot classical process matrix $(\overline{W}_{(k)})_{|\overline{\mathsf{M}}_k^{(i)}} := \overline{W}_{(k)} * \overline{\mathsf{M}}_k^{(i)}$ is causally separable and hence, by Definition 25, $(\overline{W}_{(k)})_{|\overline{\mathsf{M}}_k^{(i)}} \otimes \rho_i$ is also a causally separable $(T-1)$-slot process matrix. Since causal separability is preserved under convex combinations, we thus find conclude that $(\overline{W}_{(k)} \otimes \rho) * \mathsf{M}_k$ is itself causally separable, thereby completing the proof. □

For classical process matrices $\widetilde{W}$ that are moreover deterministic, causal separability can be characterised even more simply. Indeed, it turns out that in this case one only needs to consider the reduced process matrices obtained by classical-deterministic actions $\widetilde{\mathsf{M}}_k$, and moreover it is sufficient to consider CPTP maps (i.e., channels) rather than arbitrary classical CP maps. We hence find that, for such processes, causal separability is equivalent to the following simpler condition.

Proposition 27 (Causally separable classical-deterministic process matrix)**.** *For $T = 1$, any T-slot classical-deterministic process matrix is causally separable. For $T \geq 2$, a T-slot classical-deterministic process matrix $\widetilde{W}$ is causally separable if and only if there exists a k for which $\widetilde{W}$ is compatible with the kth operation acting first, and such that for any classical-deterministic channel $\widetilde{\mathsf{M}}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, the $(T-1)$-slot classical-deterministic process matrix $\widetilde{W}_{|\widetilde{\mathsf{M}}_k} := \widetilde{W} * \widetilde{\mathsf{M}}_k$ is itself causally separable.*

Proof. Clearly, if $\widetilde{W}$ is causally separable according to Proposition 26, then it also satisfies to conditions of this proposition, as classical-deterministic channels are a subset of classical CP maps, and the fact that $\widetilde{W}$ is deterministic means that it cannot be decomposed as a mixture of classical processes. We hence focus on proving the converse statement.

We will prove that a classical-deterministic process matrix satisfying the conditions of the proposition is also causally separable according to Proposition 26. Clearly, for $T = 1$ this is the case, since any 1-slot process matrix is causally separable.

Consider now that $\widetilde{W}$ is a classical-deterministic process matrix which is compatible with slot k being applied first and for which, for any classical-deterministic channel $\widetilde{\mathsf{M}}_k \in \mathscr{L}(\mathscr{H}^{(IO)_k})$, $\widetilde{W}_{|\widetilde{\mathsf{M}}_k}$ is causally separable. We want to show that for any classical CP map, $\overline{\mathsf{M}}_k$, the conditional process matrix $\widetilde{W} * \overline{\mathsf{M}}_k$ is also causally separable according to Proposition 26 (and hence also according to Definition 25).

Consider the maps $\{\widetilde{\mathsf{M}}_k^{[i,l]} = |i\rangle\langle i|^{I_k} \otimes |l\rangle\langle l|^{O_k}\}_{i,l}$, for $i \in \{0,\ldots,d_{I_k}-1\}$ and $l \in \{0,\ldots,d_{O_k}-1\}$, which form a basis for the space of classical-deterministic CP maps on $\mathscr{L}(\mathscr{H}^{(IO)_k})$. In particular, any CP map $\overline{\mathsf{M}}_k$ can be decomposed as $\overline{\mathsf{M}}_k = \sum_{i,l} \mu_{i,l} \widetilde{\mathsf{M}}_k^{[i,l]}$ for some coefficients $\mu_{i,l} \geq 0$.

Because $\widetilde{W}$ is deterministic, there exists a unique $i \in \{0,\ldots,d_{I_k}-1\}$ (the input that slot k, which is applied first, receives) such that for any l, $\widetilde{W} * \widetilde{\mathsf{M}}_k^{[i,l]} \neq 0$, and it follows by linearity that

$$\widetilde{W} * \overline{\mathsf{M}}_k = \sum_l \mu_{i,l} \widetilde{W} * \widetilde{\mathsf{M}}_k^{[i,l]}. \tag{69}$$

For any l, $\sum_j \widetilde{\mathsf{M}}_k^{[j,l]}$ is a CPTP map and $\widetilde{W} * \sum_j \widetilde{\mathsf{M}}_k^{[j,l]} = \widetilde{W} * \widetilde{\mathsf{M}}_k^{[i,l]}$. It then follows that

$$\widetilde{W} * \overline{\mathsf{M}}_k = \sum_l \mu_{i,l} \widetilde{W} * \widetilde{\mathsf{M}}_k^{[i,l]} \tag{70}$$

$$= \sum_l \mu_{i,l} \widetilde{W} * \sum_j \widetilde{\mathsf{M}}_k^{[j,l]}, \tag{71}$$

and, by assumption, $\widetilde{W} * \sum_j \widetilde{\mathsf{M}}_k^{[j,l]}$ is a causally separable $(T-1)$-slot classical-deterministic process matrix. The conditional process matrix $\widetilde{W} * \overline{\mathsf{M}}_k$ is therefore, by linearity, also causally separable (up to normalisation), which concludes the proof. □

Following the equivalence of classical-deterministic process matrices and process functions (see Appendix C.1), we obtain the following corollary.

Corollary 28. *A classical-deterministic process $w : \bigtimes_{k=1}^{T} \mathscr{O}_k \to \bigtimes_{k=1}^{T} \mathscr{I}_k$ is causally definite if and only if the corresponding classical-deterministic process matrix $\widetilde{W} \in \mathscr{L}(\mathscr{H}^{(IO)_{[T]}})$ defined as in Eq.* (56) *is causally separable.*

This proves the consistency of our proposed definition of causally definite process functions with the established notion of causal separability for process matrices. We reiterate that this equivalence holds also in the case where $\mathscr{P}$ and $\mathscr{F}$ are nontrivial, simply be reinterpreting these as additional slots with trivial input and output spaces, respectively.

D Properties of the function f_{6q}

D.1 A numerical method to compute the quantum query complexity of f_{6q}

In this appendix, we give some details on the semidefinite program (SDP) mentioned in Section 7.2 to show that the Boolean function f_{6q} cannot be computed sequentially with less than four quantum queries. The SDP is due to [14] and was used in [38] to obtain the minimum error $\varepsilon_T^{\text{Seq}}(f)$ with which one can compute a Boolean function f in T queries to the quantum oracle $\widetilde{O}_x$, i.e., for which there exists a T-slot supermap $\mathscr{S}^{\text{Seq}}$ such that for all x, measuring the qubit state $\mathscr{S}^{\text{Seq}}(\widetilde{\mathscr{O}}_x, \ldots, \widetilde{\mathscr{O}}_x) = \rho_x$ in the computational basis gives the outcome $f(x)$ with probability greater than $1 - \varepsilon_T^{\text{Seq}}(f)$. The SDP is formulated as follows, where $\circ$ is the Hadamard (entry-wise) matrix product.

Theorem 29 ([14])**.** *The minimum bounded error $\varepsilon_T^{Seq}(f)$ for which there exists a sequential supermap $\mathscr{S}^{\text{Seq}}$ that computes $f : \{0,1\}^n \to \{0,1\}$ in T quantum queries is given by the SDP*

$$\varepsilon_T^{Seq}(f) = \min_{\varepsilon, \{M_i^{(j)}\}_{i,j}, \Gamma_0, \Gamma_1} \quad \varepsilon \tag{72}$$

$$\text{s.t.} \quad \sum_{i=0}^{n} M_i^{(0)} = E_0, \tag{73}$$

$$\sum_{i=0}^{n} M_i^{(j)} = \sum_{i=0}^{n} E_i \circ M_i^{(j-1)}, \text{ for } 1 \le j \le T-1, \tag{74}$$

$$\Gamma_0 + \Gamma_1 = \sum_{i=0}^{n} E_i \circ M_i^{(T-1)}, \tag{75}$$

$$F_0 \circ \Gamma_0 = (1-\varepsilon) F_0, \tag{76}$$

$$F_1 \circ \Gamma_1 = (1-\varepsilon) F_1, \tag{77}$$

where the $\{M_i^{(j)}\}_{i,j}$ (for $0 \le i \le n$ and $0 \le j \le T-1$) and Γ_0, Γ_1 are all 2^n-dimensional real symmetric positive semidefinite matrices, E_0 is the constant 1 matrix and for $1 \le i \le n$ the E_i are defined such that $\langle x| E_i |y\rangle = (-1)^{x_i + y_i}$, and where F_0 and F_1 are diagonal matrices such that $\langle x| F_z |x\rangle = 1$ if and only if $f(x) = z$, otherwise $\langle x| F_z |x\rangle = 0$.

This method was used in [38] to compute the sequential quantum query complexity of symmetric Boolean functions up to 6 bits in order to prove some separations between D and Q_E. However, because f_{6q} is not symmetric, the computation was not performed for this specific function. Solving numerically the SDP, we obtain the following result.

Proposition 13. $Q_E(f_{6q}) = 4$.

Proof. By fixing the number of queries to $T = 3$, and solving the SDP of Theorem 29 we obtain $\varepsilon_3^{\text{Seq}}(f_{6q}) = 0.0207$. This result is subject to numerical imprecisions, but with both the primal and dual converging to the same values and with constraints that are violated only up to a magnitude of 10^{-8}, this value of $\varepsilon_3^{\text{Seq}}$ can be judged sufficiently far from 0 to assert with confidence that f_{6q} cannot be computed in three queries.[8] The proof is completed by recalling that f_{6q} can be computed in four quantum queries as described in Section 7.2. □

Our code is freely accessible on Github.[9] with the SDP of Theorem 29 being formulated using the package Yalmip [36] and solved using Mosek [1].

D.2 A causally indefinite quantum supermap computing f_{6q}

We finish by providing a more detailed proof the a causally indefinite quantum supermap that can compute the Boolean function f_{6q} in three quantum queries. The truth table for f_{6q} is shown in Table 1, and will be useful for the following proposition.

Table 3: Simplified truth table of the Boolean function f_{6q}.

$x_1 \oplus x_4$	$x_2 \oplus x_5$	$x_3 \oplus x_6$	$f_{6q}(x_1,\ldots,x_6)$
0	0	0	0
1	0	0	x_5
0	1	0	x_6
0	0	1	x_4
1	1	0	x_5
1	0	1	x_4
0	1	1	x_6
1	1	1	1

Proposition 14. $Q_E^{\text{Gen}}(f_{6q}) = 3$.

Proof. Recall that the classical-deterministic Lugano process is defined as a function w_{Lugano} mapping the binary output sets $\mathscr{O}_1 \times \mathscr{O}_2 \times \mathscr{O}_3$ to the binary input sets $\mathscr{I}_1 \times \mathscr{I}_2 \times \mathscr{I}_3$ such that $w_k(o_1,o_2,o_3) = (1 \oplus o_{k\oplus_3 1})o_{k\oplus_3 2}$ (see Section 3). This classical-deterministic process can be described a process matrix $W_{\text{Lugano}} \in \mathscr{L}(\mathscr{H}^{(IO)_{[3]}})$ where, for $1 \leq k \leq 3$, the $\mathscr{H}^{O_k}$ and $\mathscr{H}^{I_k}$ are two-dimensional Hilbert spaces,

$$W_{\text{Lugano}} = \sum_{o_1,o_2,o_3 \in \{0,1\}} |o_1,o_2,o_3\rangle\langle o_1,o_2,o_3|^{O_1O_2O_3} \otimes |\overline{o}_2 o_3, \overline{o}_3 o_1, \overline{o}_1 o_2\rangle\langle \overline{o}_2 o_3, \overline{o}_3 o_1, \overline{o}_1 o_2|^{I_1I_2I_3}, \qquad (78)$$

[8] To obtain a rigorous lower-bound on $\varepsilon_3^{\text{Seq}}$, one could extract a certificate from the dual SDP of Theorem 29, in a method similar to that developed in [21].

[9] `https://github.com/pierrepocreau/Classical-QuantumQueryComplexity_ICO`

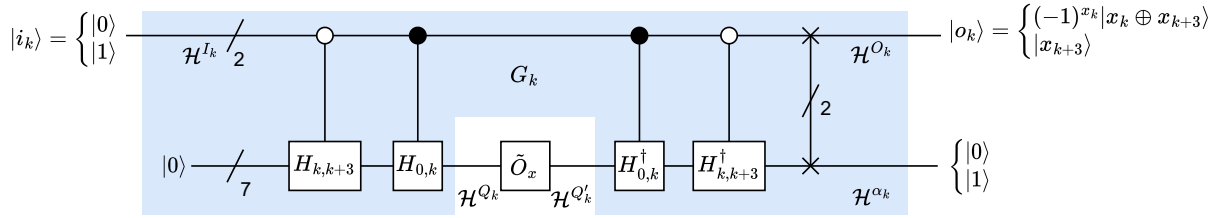


Figure 2: Circuit diagram of the quantum subroutines G_k. The unitary $H_{i,j}$ satisfies $H_{i,j}|0\rangle = \frac{|i\rangle+|j\rangle}{\sqrt{2}}$ and $H_{i,j}|1\rangle = \frac{|i\rangle-|j\rangle}{\sqrt{2}}$ for $0 \leq i,j \leq 6$, $i \neq j$, and can be completed arbitrarily on the other inputs. The final operation is a swap operation $\text{SWAP} : H^{O_k} \otimes H^{\alpha_k} \to H^{O_k} \otimes H^{\alpha_k}$ such that for any a and b in $\{0,1\}$, $\text{SWAP}\,|a\rangle^{O_k}|b\rangle^{\alpha_k} = |b\rangle^{O_k}|a\rangle^{\alpha_k}$ and for any $c \in \{2,\ldots,6\}$, $\text{SWAP}\,|a\rangle^{O_k}|c\rangle^{\alpha_k} = |a\rangle^{O_k}|c\rangle^{\alpha_k}$.

with $\overline{o}_k := (1 \oplus o_k)$. Note that any classical-deterministic process can be embedded in such a way [19, 11]; see also Appendix C.1. Because this process is classical, we can extend it with a (8-dimensional) future space $\mathscr{L}(\mathscr{H}^F)$ which keeps a copy of the classical values in the output registers $\mathscr{L}(\mathscr{H}^{O_1O_2O_3})$, defining the process

$$\begin{aligned}\widetilde{W}_{\text{Lugano}} = \sum_{o_1,o_2,o_3\in\{0,1\}} & |o_1,o_2,o_3\rangle\langle o_1,o_2,o_3|^{O_1O_2O_3} \\ & \otimes |\overline{o}_2 o_3, \overline{o}_3 o_1, \overline{o}_1 o_2\rangle\langle \overline{o}_2 o_3, \overline{o}_3 o_1, \overline{o}_1 o_2|^{I_1I_2I_3} \otimes |o_1,o_2,o_3\rangle\langle o_1,o_2,o_3|^F .\end{aligned} \tag{79}$$

We now define some quantum subroutines that can be used to compute the parity between different bits in one quantum query. These subroutines, indexed by $k \in \{1,2,3\}$, are formally one-slot quantum supermaps with corresponding Choi representations (i.e., process matrices) $G_k \in \mathscr{L}(\mathscr{H}^{I_kQ_kQ'_kO_k\alpha_k})$, and take a quantum channel from $\mathscr{L}(\mathscr{H}^{Q_k})$ to $\mathscr{L}(\mathscr{H}^{Q'_k})$ as input, with both $\mathscr{H}^{Q_k}$ and $\mathscr{H}^{Q'_k}$ being Hilbert spaces of dimension 7, and where the $\mathscr{H}^{\alpha_k}$ are also a Hilbert spaces of dimension 7. We show a circuit description of these supermaps in Figure 2. When acting on the query oracle $\widetilde{O}_x$, the G_k act as follows: if the qubit in $\mathscr{H}^{I_k}$ is $|0\rangle$, then the output qubit in $\mathscr{H}^{O_k}$ will be $|x_k \otimes x_{k+3}\rangle$; however, if the qubit in $\mathscr{H}^{I_k}$ is $|1\rangle$, then the output in $\mathscr{H}^{O_k}$ is $|x_{k+3}\rangle$. This dependency on the value of the qubit in $\mathscr{H}^{I_k}$ is similar to the behaviour of the classical-deterministic process $\overline{w}_{\text{Lugano}}$ defined in the proof of Proposition 8. Indeed, that process was defined as a modification of the original w_{Lugano} such that if a function f_k received 0 (resp. 1) in the process w_{Lugano}, it would receive k (resp. $k+3$) in $\tilde{w}_{\text{Lugano}}$.

The last step is thus to compose the Lugano supermap with these quantum subroutines. This composition can be expressed at the level of their Choi matrices via the link product as

$$\widetilde{W}_{f_{6q}} := \widetilde{W}_{\text{Lugano}} * G_1 * G_2 * G_3 \in \mathscr{L}(\mathscr{H}^{Q_1Q_2Q_3Q'_1Q'_2Q'_3F\alpha_1\alpha_2\alpha_3}). \tag{80}$$

This process matrix characterises a quantum supermap that we write $\mathscr{S}^{f_{6q}}$, which takes three channels as input, from $\mathscr{L}(\mathscr{H}^{Q_k})$ to $\mathscr{L}(\mathscr{H}^{Q'_k})$, for $1 \leq k \leq 3$ and with a global future space $\mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3})$. We summarise in Table 4 its action for any $x \in \{0,1\}^6$ on three copies of the query oracle $\widetilde{O}_x$ (with Choi matrix $\widetilde{O}_x$), which produces the output state

$$\widetilde{W}_{f_{6q}} * \widetilde{\mathsf{O}}_x^{\otimes 3} \in \mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3}), \tag{81}$$

where we write, with a slight abuse of notation, $\widetilde{\mathsf{O}}_x^{\otimes 3} = \widetilde{\mathsf{O}}_x \otimes \widetilde{\mathsf{O}}_x \otimes \widetilde{\mathsf{O}}_x \in \mathscr{L}(\mathscr{H}^{Q_1Q_2Q_3Q'_1Q'_2Q'_3})$.

Table 4: Values of the registers $\mathscr{H}^F$ and $\mathscr{H}^{\alpha_1\alpha_2\alpha_3}$ when the supermap characterised by $\widetilde{W}_{f_{6q}}$ acts on three copies of $\widetilde{O}_x$, for different values of x.

$x_1 \oplus x_4$	$x_2 \oplus x_5$	$x_3 \oplus x_6$	$\mathscr{H}^F$	$\mathscr{H}^{\alpha_1\alpha_2\alpha_3}$	$f_{6q}(x)$
0	0	0	$\|000\rangle$	$\|000\rangle$	0
1	0	0	$\|1x_50\rangle$	$\|010\rangle$	x_5
0	1	0	$\|01x_6\rangle$	$\|001\rangle$	x_6
0	0	1	$\|x_401\rangle$	$\|100\rangle$	x_4
1	1	0	$\|1x_50\rangle$	$\|010\rangle$	x_5
1	0	1	$\|x_401\rangle$	$\|100\rangle$	x_4
0	1	1	$\|01x_6\rangle$	$\|001\rangle$	x_6
1	1	1	$\|111\rangle$	$\|000\rangle$	1

We can see from Table 4 that by measuring the three states in $\mathscr{H}^{\alpha_k}$ in the computational basis $\{|i\rangle\}_{i\in\{0,\dots,6\}}$, one obtains the necessary information to extract the value of $f(x)$ from the register $\mathscr{H}^F$, for any $x \in \{0,1\}^6$. Indeed, if the measurement result is 000, then measuring any qubit of $\mathscr{H}^F$ in the computational basis outputs the value of $f(x)$. If the measurement result contains a 1, then measuring, in the computational basis, the qubit of $\mathscr{H}^F$ that is in the same position as that 1 also gives the values of $f(x)$. This measurement procedure can be seen as a channel $\mathscr{M} : \mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3}) \to \mathscr{L}(\mathscr{H}^{\tilde{F}})$ with $\dim(\mathscr{H}^{\tilde{F}}) = 2$, which post-processes the output such that measuring $\widetilde{W}_{f_{6q}} * \widetilde{\mathsf{O}}_x^{\otimes 3} * \mathsf{M} \in \mathscr{L}(\mathscr{H}^{\tilde{F}})$ in the computational basis outputs $f_{6q}(x)$ with probability one, concluding the proof. □

In this proof, the final composition with the channel $\mathscr{M}$ described above highlights that the causally indefinite computation we described does not compute f_{6q} "cleanly". Indeed, because the state $\widetilde{W}_{f_{6q}} * \widetilde{\mathsf{O}}_x^{\otimes 3} \in \mathscr{L}(\mathscr{H}^{F\alpha_1\alpha_2\alpha_3})$ contains information about $f_{6q}(x)$ but also on x (see Table 4), this channel $\mathscr{M}$ erases this additional information about x. This extra information prevents us from directly amplifying this quantum query separation, as it prevents us from using this supermap as a coherent subroutine to compute a recursive composition of f_{6q}.

Dagger-Drazin Inverses

Robin Cockett*
Department of Computer Science
University of Calgary, Canada
robin@ucalgary.ca

Jean-Simon Pacaud Lemay
School of Mathematical and Physical Sciences
Macquarie University, Australia
js.lemay@mq.edu.au

Priyaa Varshinee Srinivasan †
Department of Software Sciences
Tallinn University of Technology, Estonia
priyaavarshinee@gmail.com

Drazin inverses are a special kind of generalized inverses that can be defined for endomorphisms in any category. A natural question to ask is whether one can somehow extend the notion of Drazin inverse to arbitrary maps – not simply endomorphisms. It turns out that this is possible and, indeed, natural to do so for dagger categories. This paper, thus, introduces the notion of a dagger-Drazin inverse, which is a new kind of generalized inverse appropriate for arbitrary maps in a dagger category. This inverse is closely related to the Drazin inverse, for having dagger-Drazin inverses is equivalent to asking that positive maps have Drazin inverses. Moreover, dagger-Drazin inverses are also closely related to Moore-Penrose inverses as we observe that a map has a Moore-Penrose inverse if and only if it is a Drazin inverse. Furthermore, we explain how Drazin inverses of opposing pairs correspond precisely to dagger-Drazin inverses in cofree dagger categories. We also give examples of dagger-Drazin inverses for matrices over (involutive) fields, bounded linear operators, and partial injections.

1 Introduction

Generalized inverses are of particular interest in matrix theory [4] since they allow one to solve linear equations $A\vec{x} = \vec{y}$ even when A is not necessarily invertible. One such kind of generalized inverse that is of interest is the *Drazin inverse* [4, Chap 7] – named after Michael P. Drazin who originally introduced the concept under the name "pseudo-inverse" [10]. Every square matrix over a field has a Drazin inverse [4, Def 7.2.3], and can be used to solve linear differential equations [4, Chap 9].

While Drazin inverses have been extensively studied in matrix theory, the notion of a Drazin inverse can more generally be defined in an arbitrary semigroup [10, 19] and they have also been well studied in ring theory, as they are deeply connected to strong π-regularity [1] and Fitting's Lemma/Theorem [11]. As such, one can consider Drazin inverses of matrices over a ring [21], linear endomorphisms of modules [28], and even bounded linear endomorphisms of Banach spaces [15] or Hilbert spaces [29]. The Drazin inverse is a useful tool for computation and thus has found many applications. In particular, for quantum information theory, Drazin inverses have been used to characterize quantum gates [30], for quantum Turing automata [3], in quantum error mitigation [5], and quantum thermodynamics [7, 6, 17].

There has also been renewed interest in studying Drazin inverses in categories. They arise naturally since for any object A in a category $\mathbb{X}$, the homset $\mathbb{X}(A,A)$ is a semigroup (in fact a monoid) with respect to composition. As such, we may consider Drazin inverses in $\mathbb{X}(A,A)$. Thus, we may talk about

*Partially funded by NSERC.
†Funded by the Estonian Research Council (MOB3JD1227).

A. Díaz-Caro, O. Oreshkov, and A.B. Sainz (Eds.):
Quantum Physics and Logic 2025 (QPL 2025)
EPTCS 426, 2025, pp. 301–316, doi:10.4204/EPTCS.426.12

Drazin inverses of endomorphisms in an arbitrary category (Def 3.1). Drazin inverses in categories were introduced by Puystjens and Robinson in their 1987 paper [25], where they were particularly interested in generalized inverses in *additive* categories. Unfortunately, a deeper categorical approach to Drazin inverses was never initiated until very recently. In [9], the authors further developed the theory of Drazin inverse in category theory. While, as one might expect, one can generalize ring theoretic Drazin inverse results to additive/abelian categories [9, Sec 6], the categorical perspective also broaden the theory of Drazin inverses in unexpected ways, such as relating Drazin inverses to the idempotent completion [9, Sec 5] and to the notion of eventual image duality [16]. However, arriving with categorical eyes to the subject of Drazin inverses it is natural to want to have a notion of a Drazin inverse for arbitrary maps, not simply endomorphisms. The main purpose of this paper is to explain how this is quite natural in *dagger* categories.

Recall that a dagger category (which we write as †-category) is a category equipped with an involution † on maps. The term "dagger category" was introduced by Selinger in [27] based on the use in physics of the symbol † for conjugate transpose. The theory of †-categories is now a firmly established sub-field of category theory with a rich literature. In particular, †-categories are a fundamental component of categorical quantum mechanics. In this paper, we introduce the notion of a *dagger-Drazin inverse* (which we write as †-Drazin inverses), which is a new kind of generalized inverse for maps in a †-category (Def 2.1). In particular, †-Drazin inverses can be defined for maps of any type and not simply endomorphisms. While a †-Drazin inverse of a map may not always exist, if it does it is unique (Prop 2.2). As the name suggests, †-Drazin inverses are in fact deeply connected to Drazin inverses (Sec 3). First of all, for self-adjoint maps (which are always endomorphisms), the notion of †-Drazin inverses and Drazin inverses coincide (Lemma 3.4). Having †-Drazin inverses it transpires corresponds precisely to positive maps (which recall are always self-adjoint endomorphisms) having Drazin inverses (Thm 3.6). We give examples of †-Drazin inverses for matrices over fields (Ex 3.10), matrices over involutive fields (Ex 3.11 & 4.4), bounded linear operators between Hilbert spaces (Ex 3.14 & 4.6), and partial injections (Ex 3.12 & 3.13).

Moreover, †-Drazin inverses are related to another kind of generalized inverse: the *Moore-Penrose* inverse. It is named after E. H. Moore and R. Penrose: Moore introduced the notion in 1920 [18], and it was independently rediscovered by Penrose in 1955 [20]. The Moore-Penrose inverse is arguably the most well-known generalized inverse, with many interesting applications [2]. In particular, every complex matrix has a Moore-Penrose inverse constructed using Singular Value Decomposition [4, Chap 1]. Moore-Penrose inverses can also be defined in †-categories, as was first done by Puystjens and Robinson in a series of papers in the 1980s [22, 23, 24, 25, 26], and later picked up again by the first and second named author in [8] for QPL2023. It turns out that one can give a novel alternative characterization of the Moore-Penrose inverse using †-Drazin inverses. Indeed, a map has a Moore-Penrose inverse if and only if it is *a* †-Drazin inverse (Thm 4.2), and in this case the Moore-Penrose inverse is the †-Drazin inverse.

We also explain how †-Drazin inverses are connected to the notion of Drazin inverses for opposing pairs [9, Sec 7] (which was the authors' first attempt at generalizing Drazin inverses for arbitrary maps in a category) via *cofree* †-categories. For any category, there is a cofree †-category over it [12], and cofree †-categories are used in reversible computation [14]. It turns out that Drazin inverses for opposing pairs correspond precisely to †-Drazin inverses in cofree †-categories (Thm 5.4).

Lastly, it is important to note the perhaps subtle distinction we make throughout this paper between *being* a †-Drazin inverse and *having* a †-Drazin inverse: the former is a stronger requirement than the latter. Thus, when a map *has* a †-Drazin inverse, it is not necessarily the case that it *is* a †-Drazin inverse. In particular, a map with a †-Drazin inverse is not in general the †-Drazin inverse of its †-Drazin inverse.

On the other hand, every †-Drazin inverse does have a †-Drazin inverse and, in this case, it is also the †-Drazin inverse of its †-Drazin inverse (Prop 2.6.(iv)+(v)).

2 Dagger-Drazin Inverses

In this section we introduce the main novel concept of this paper: *dagger-Drazin inverses*, which is a new kind of generalized inverses in a dagger category that is, as we will see in the next section, deeply connected to the notion of Drazin inverses.

For an arbitrary category $\mathbb{X}$, we denote objects by capital letters A,B,C, etc. and maps by lowercase letters f,g,x,y, etc. Identity maps are denoted as $1_A : A \to A$. Composition is written in *diagrammatic order*, that is, the composition of a map $f : A \to B$ followed by $g : B \to C$ is denoted $fg : A \to C$. Then recall that a **dagger** on a category $\mathbb{X}$ is a contravariant functor $(\_)^\dagger : \mathbb{X} \to \mathbb{X}$ which is the identity on objects and involutive. We refer to $f^\dagger$ as the **adjoint** of f. A **†-category** is a category $\mathbb{X}$ equipped with a chosen dagger †. Concretely, a †-category can be described as a category $\mathbb{X}$ where for each map $f : A \to B$, there is a chosen map of dual type $f^\dagger : B \to A$ such that $1_A^\dagger = 1_A$, $(fg)^\dagger = g^\dagger f^\dagger$, and $(f^\dagger)^\dagger = f$. For a more in-depth introduction to dagger categories, we refer the reader to [13].

Definition 2.1 *In a †-category, a* ***†-Drazin inverse*** *of a map $f : A \to B$ is a map of dual type $f^\partial : B \to A$ such that:*

[D†.1] *There is a $k \in \mathbb{N}$ such that*[1] $(ff^\dagger)^k ff^\partial = (ff^\dagger)^k$ *and* $f^\partial f(f^\dagger f)^k = (f^\dagger f)^k$*;*

[D†.2] $f^\partial ff^\partial = f^\partial$*;*

[D†.3] $(ff^\partial)^\dagger = ff^\partial$*;*

[D†.4] $(f^\partial f)^\dagger = f^\partial f$.

If f has a †-Drazin inverse, we say that f is ***†-Drazin invertible*** *or simply* ***†-Drazin****. We call the smallest natural number k such that* ***[D†.1]*** *holds the* ***†-Drazin index of*** *f, which we denote by* $\mathsf{ind}^\partial(f)$*. A* ***†-Drazin category*** *is a †-category such that every map is †-Drazin.*

Proposition 2.2 *In a †-category, if a map has a †-Drazin inverse, then it is unique.*

PROOF: Suppose that a map f has two possible †-Drazin inverses f^∂ and f^δ. To show that these maps are indeed equal, it will be useful to first compute that for any $n \in \mathbb{N}$ the following equalities hold:

$$f^\partial = \left(f^\dagger f\right)^m f^\partial \left((f^{\partial^\dagger} f^\partial)^\dagger\right)^m \qquad\qquad f^\delta = \left(f^\delta (f^{\delta^\dagger})^\dagger\right)^m f^\delta (ff^\dagger)^m \tag{1}$$

Clearly these hold for $m = 0$. Then for $m \geq 1$, to compute the above equality on the left, we recursively use **[D†.2]**, **[D†.3]**, and **[D†.4]** as follows:

$$f^\partial \overset{\text{[D†.2]}}{=} f^\partial ff^\partial \overset{\text{[D†.4]}}{=} (f^\partial f)^\dagger f^\partial = f^\dagger f^{\partial^\dagger} f^\partial \overset{\text{[D†.2]}}{=} f^\dagger (f^\partial ff^\partial)^\dagger f^\partial = f^\dagger f^{\partial^\dagger} f^\dagger f^{\partial^\dagger} f^\partial$$
$$= f^\dagger (ff^\partial)^\dagger (f^{\partial^\dagger} f^\partial)^\dagger \overset{\text{[D†.3]}}{=} f^\dagger ff^\partial (f^{\partial^\dagger} f^\partial)^\dagger = \ldots = \left(f^\dagger f\right)^m f^\partial \left((f^{\partial^\dagger} f^\partial)^\dagger\right)^m$$

Similarly, we can also prove the right equality of (1). Now by **[D†.1]**, there exists a $k_1 \in \mathbb{N}$ such that $(ff^\dagger)^{k_1} ff^\partial = (ff^\dagger)^{k_1}$, and there also exists a $k_2 \in \mathbb{N}$ such that $f^\delta f(f^\dagger f)^{k_2} = (f^\dagger f)^{k_2}$. Then we compute:

$$f^\delta \overset{(1)}{=} \left(f^\delta (f^{\delta^\dagger})^\dagger\right)^k f^\delta (ff^\dagger)^k \overset{\text{[D†.1]}}{=} \left(f^\delta (f^{\delta^\dagger})^\dagger\right)^k f^\delta (ff^\dagger)^k ff^\partial \overset{(1)}{=} f^\delta ff^\partial$$
$$\overset{(1)}{=} f^\delta f \left(f^\dagger f\right)^{k_2} f^\partial \left((f^{\partial^\dagger} f^\partial)^\dagger\right)^{k_2} \overset{\text{[D†.1]}}{=} \left(f^\dagger f\right)^{k_2} f^\partial \left((f^{\partial^\dagger} f^\partial)^\dagger\right)^{k_2} \overset{(1)}{=} f^\partial$$

[1] By convention, for an endomorphism x, $x^0 = 1_A$.

So $f^\delta = f^\partial$, and therefore we conclude that †-Drazin inverses are unique. □

Therefore, we can now speak of *the* †-Drazin inverse of a map. This also implies that for a †-category, being †-Drazin is a property rather than structure. We give examples of †-Drazin inverses/categories in the following sections. We also remark that there is some redundancy in the above definition since **[D†.1]** can be expressed in other equivalent ways.

Lemma 2.3 *In a †-category, given maps $f : A \to B$ and $f^\partial : B \to A$ which together satisfy **[D†.3]** and **[D†.4]**, then **[D†.1]** is equivalent to any of the following statements:*

[D†.5] *There is a $j \in \mathbb{N}$ such that $ff^\partial(ff^\dagger)^j = (ff^\dagger)^j$;*

[D†.6] *There is a $k \in \mathbb{N}$ such that $(ff^\dagger)^k ff^\partial = (ff^\dagger)^k$;*

[D†.7] *There is an $m \in \mathbb{N}$ such that $f^\partial f(f^\dagger f)^m = (f^\dagger f)^m$;*

[D†.8] *There is an $n \in \mathbb{N}$ such that $(f^\dagger f)^n f^\partial f = (f^\dagger f)^n$.*

Moreover if f is †-Drazin, then **[D†.5]–[D†.8]** *all hold for all $k \geq \mathsf{ind}^\partial(f)$.*

PROOF: We will first explain why **[D†.5]–[D†.8]** are all equivalent. For **[D†.5]** ⇒ **[D†.6]**, suppose we have that $ff^\partial(ff^\dagger)^j = (ff^\dagger)^j$ for some j. Then setting $k = j$, we compute that:

$$(ff^\dagger)^j ff^\partial \overset{\textbf{[D}^\dagger\textbf{.3]}}{=} (ff^\dagger)^j (ff^\partial)^\dagger = \left((ff^\dagger)^\dagger\right)^j (ff^\partial)^\dagger = \left(ff^\partial (ff^\dagger)^j\right)^\dagger \overset{\textbf{[D}^\dagger\textbf{.5]}}{=} \left((ff^\dagger)^\dagger\right)^j = (ff^\dagger)^j$$

For **[D†.6]** ⇒ **[D†.7]**, suppose we have that $(ff^\dagger)^k ff^\partial = (ff^\dagger)^k$ for some k. Setting $m = k+1$, we compute that:

$$f^\partial f(f^\dagger f)^{k+1} \overset{\textbf{[D}^\dagger\textbf{.4]}}{=} (f^\partial f)^\dagger (f^\dagger f)^{k+1} = (f^\partial f)^\dagger \left((f^\dagger f)^\dagger\right)^{k+1} = \left((f^\dagger f)^{k+1} f^\partial f\right)^\dagger = \left(f^\dagger (ff^\dagger)^k ff^\partial f\right)^\dagger$$
$$\overset{\textbf{[D}^\dagger\textbf{.6]}}{=} \left(f^\dagger (ff^\dagger)^k f\right)^\dagger = \left((f^\dagger f)^{k+1}\right)^\dagger = \left((f^\dagger f)^\dagger\right)^{k+1} = (f^\dagger f)^{k+1}$$

For the remaining parts, **[D†.7]** ⇒ **[D†.8]** is proven is similar fashion to how **[D†.5]** ⇒ **[D†.6]** was proved (though using **[D†.4]** instead), while **[D†.8]** ⇒ **[D†.5]** is proven similar fashion to how we showed **[D†.6]** ⇒ **[D†.7]** (but where we use **[D†.3]** instead). Next note that **[D†.1]** is precisely **[D†.6]** and **[D†.7]** together, and therefore is indeed equivalent to any of **[D†.5]–[D†.8]**. Lastly, it is straightforward to check that if f is †-Drazin, then **[D†.5]–[D†.8]** all hold for all $k \geq \mathsf{ind}^\partial(f)$. □

Corollary 2.4 *In a †-category, for a map $f : A \to B$, a map of dual type $f^\partial : B \to A$ is the †-Drazin inverse of f if and only if f and f^∂ satisfies **[D†.2]**, **[D†.3]**, **[D†.4]**, and any one of* **[D†.5]–[D†.8]**.

In any †-category, there are certain maps which are †-Drazin by default:

Lemma 2.5 *In a †-category,*

(i) Identity maps 1_A are †-Drazin where $1_A^\partial = 1_A$ and $\mathsf{ind}^\partial(1_A) = 0$;

(ii) If f is an isomorphism then f is †-Drazin where $f^\partial = f^{-1}$ and $\mathsf{ind}^\partial(f) = 0$;

(iii) If f is a partial isometry (i.e. $ff^\dagger f = f$) then f is †-Drazin where $f^\partial = f^\dagger$ and $\mathsf{ind}^\partial(f) \leq 1$;

(iv) If f is unitary (i.e. $ff^\dagger = 1$ and $f^\dagger f = 1$) then f is †-Drazin where $f^\partial = f^\dagger$ and $\mathsf{ind}^\partial(f) = 0$;

(v) If e is a †-idempotent (i.e. $ee = e$ and $e^\dagger = e$) then e is †-Drazin where $e^\partial = e$. Furthermore, when $e \neq 1_A$, $\mathsf{ind}^\partial(e) = 1$ and $\mathsf{ind}^\partial(e) = 0$ precisely when $e = 1_A$.

PROOF: These are straightforward to check, so we leave them as an exercise for the reader. □

Having a dagger-inverse has a number of consequences:

Proposition 2.6 *In a †-category, if a map f is †-Drazin then:*

(i) $ff^\partial = f^{\partial^\dagger} f^\dagger$ *and* $f^\partial f = f^\dagger f^{\partial^\dagger}$;

(ii) $f^\partial = f^\dagger f^{\partial^\dagger} f^\partial = f^\partial f^{\partial^\dagger} f^\dagger$ *and* $f^{\partial^\dagger} = ff^\partial f^{\partial^\dagger} = f^{\partial^\dagger} f^\partial f$;

(iii) $f^\dagger$ *is also †-Drazin where* $f^{\dagger^\partial} = f^{\partial^\dagger}$ *and* $\mathsf{ind}^\partial(f) = \mathsf{ind}^\partial(f^\dagger)$;

(iv) f^∂ *is †-Drazin where* $f^{\partial\partial} = ff^\partial f$ *and* $\mathsf{ind}^\partial(f^\partial) \leq 1$;

(v) $f^{\partial\partial\partial} = f^\partial$;

(vi) If $f^\partial = f^\dagger$ *then f is a partial isometry;*

(vii) If $\mathsf{ind}^\partial(f) = 0$ *then f is an isomorphism and* $f^{-1} = f^\partial$;

(viii) If $f^\partial = f^\dagger$ *and* $\mathsf{ind}^\partial(f) = 0$ *then f is unitary;*

(ix) $ff^\dagger$ *and* $f^\dagger f$ *are †-Drazin where* $(ff^\dagger)^\partial = f^{\dagger^\partial} f^\partial$ *and* $(f^\dagger f)^\partial = f^\partial f^{\dagger^\partial}$;

(x) ff^∂ *and* $f^\partial f$ *are partial isometries and so †-Drazin where* $(ff^\partial)^\partial = ff^\partial$ *and* $(f^\partial f)^\partial = f^\partial f$.

PROOF: For (i), this is just expanding out **[D†.3]** and **[D†.4]** expanded out. For (ii), this follows from **[D†.2]** and (i). For (iii), suppose that $\mathsf{ind}^\partial(f) = k$. We show that $f^{\dagger^\partial} := f^{\partial^\dagger}$ satisfies the necessary axioms to be a †-Drazin inverse of $f^\dagger$:

[D†.6] $(f^\dagger f)^k f^\dagger f^{\dagger^\partial} \overset{\text{def.}}{=} (f^\dagger f)^k f^\dagger f^{\partial^\dagger} \overset{\text{(i)}}{=} (f^\dagger f)^k f^\partial f \overset{\textbf{[D†.8]}}{=} (f^\dagger f)^k$

[D†.2] $f^{\dagger^\partial} f^\dagger f^{\dagger^\partial} \overset{\text{def.}}{=} f^{\partial^\dagger} f^\dagger f^{\partial^\dagger} = \left(f^\partial f f^\partial\right)^\dagger \overset{\textbf{[D†.2]}}{=} f^{\partial^\dagger} \overset{\text{def.}}{=} f^{\dagger^\partial}$

[D†.3] $\left(f^\dagger f^{\dagger^\partial}\right)^\dagger \overset{\text{def.}}{=} \left(f^\dagger f^{\partial^\dagger}\right)^\dagger \overset{\text{(i)}}{=} (f^\partial f)^\dagger \overset{\textbf{[D†.4]}}{=} f^\partial f \overset{\text{(i)}}{=} f^\dagger f^{\partial^\dagger} \overset{\text{def.}}{=} f^\dagger f^{\dagger^\partial}$

[D†.4] $\left(f^{\dagger^\partial} f^\dagger\right)^\dagger \overset{\text{def.}}{=} \left(f^{\partial^\dagger} f^\dagger\right)^\dagger \overset{\text{(i)}}{=} (ff^\partial)^\dagger \overset{\textbf{[D†.3]}}{=} ff^\partial \overset{\text{(i)}}{=} f^{\partial^\dagger} f^\dagger \overset{\text{def.}}{=} f^{\dagger^\partial} f^\dagger$

So we conclude that $f^\dagger$ is †-Drazin. It is straightforward to check that $\mathsf{ind}^\partial(f) = \mathsf{ind}^\partial(f^\dagger)$. For (iv), we show that $f^{\partial\partial} := ff^\partial f$ satisfies the necessary axioms to be †-Drazin inverse of f^∂.

[D†.6] $f^\partial f^{\partial^\dagger} f^\partial f^{\partial\partial} \overset{\text{def.}}{=} f^\partial f^{\partial^\dagger} f^\partial ff^\partial f \overset{\text{(i)}}{=} f^\partial f^{\partial^\dagger} f^\dagger f^{\partial^\dagger} = f^\partial (f^\partial ff^\partial)^\dagger \overset{\textbf{[D†.2]}}{=} f^\partial f^{\partial^\dagger}$

[D†.2] $f^{\partial\partial} f^\partial f^{\partial\partial} \overset{\text{def.}}{=} ff^\partial ff^\partial ff^\partial f \overset{\textbf{[D†.2]}}{=} ff^\partial ff^\partial f \overset{\textbf{[D†.2]}}{=} ff^\partial f \overset{\text{def.}}{=} f^{\partial\partial}$

[D†.3] $(f^\partial f^{\partial\partial})^\dagger \overset{\text{def.}}{=} (f^\partial ff^\partial f)^\dagger \overset{\textbf{[D†.2]}}{=} (f^\partial f)^\dagger \overset{\textbf{[D†.4]}}{=} f^\partial f \overset{\textbf{[D†.2]}}{=} f^\partial ff^\partial f \overset{\text{def.}}{=} f^\partial f^{\partial\partial}$

[D†.4] $(f^{\partial\partial} f^\partial)^\dagger \overset{\text{def.}}{=} (ff^\partial ff^\partial)^\dagger \overset{\textbf{[D†.2]}}{=} (ff^\partial)^\dagger \overset{\textbf{[D†.3]}}{=} ff^\partial \overset{\textbf{[D†.2]}}{=} ff^\partial ff^\partial \overset{\text{def.}}{=} f^{\partial\partial} f^\partial$

So we conclude that $f^{\partial\partial} = ff^\partial f$ is the †-Drazin inverse of f^∂. It easy to check that $\mathsf{ind}^\partial(f^\partial) \leq 1$. Now for (v), applying (iv) to f^∂ we get that $f^{\partial\partial}$ is also †-Drazin where $f^{\partial\partial\partial} = f^\partial f^{\partial\partial} f^\partial$. Expanding this out further, we compute that $f^{\partial\partial\partial} \overset{\text{def.}}{=} f^\partial f^{\partial\partial} f^\partial \overset{\text{def.}}{=} f^\partial ff^\partial ff^\partial \overset{\textbf{[D†.2]}}{=} f^\partial ff^\partial \overset{\textbf{[D†.2]}}{=} f^\partial$. So f^∂ is the †-Drazin inverse of $f^{\partial\partial}$. For (vi), suppose that $f^\partial = f^\dagger$. Then **[D†.2]** tells us that $f^\dagger ff^\dagger = f^\dagger$, which says that $f^\dagger$ is a partial isometry. Then applying the dagger to this equality gives $ff^\dagger f = f$, and so we have that f is a partial isometry. For (vii), suppose that $\mathsf{ind}^\partial(f) = 0$. This means that $ff^\partial = 1$ and

$f^\partial f = 1$, which is precisely that f is an isomorphism with inverse $f^{-1} = f^\partial$. Then (viii) immediately follows from (vi) and (vii). We will prove (ix) in Thm 3.6. For (x), note that we can easily check that $ff^\partial(ff^\partial)^\dagger ff^\partial \overset{\textbf{[D}^\dagger\textbf{.3]}}{=} ff^\partial ff^\partial ff^\partial \overset{\textbf{[D}^\dagger\textbf{.2]}}{=} ff^\partial ff^\partial \overset{\textbf{[D}^\dagger\textbf{.2]}}{=} ff^\partial$, and similarly that $f^\partial f(f^\partial f)^\dagger f^\partial f = f^\partial f$. So both ff^∂ and $f^\partial f$ are partial isometries, so by Lemma 2.5.(iii), they are both †-Drazin and their adjoints are their †-Drazin inverse. However by **[D†.3]** and **[D†.4]**, we get $(ff^\partial)^\partial = ff^\partial$ and $(f^\partial f)^\partial = f^\partial f$. □

Unfortunately, like other generalized inverses (such as the Drazin inverse and the Moore-Penrose inverse), †-Drazin inverses do not necessarily play well with composition. Indeed, even if f and g are †-Drazin, fg is may not be †-Drazin, and even if it was, $(fg)^\partial$ may not equal $g^\partial f^\partial$.

We conclude this section by discussing the special case of having a †-Drazin index less than or equal to 1. In classical Drazin inverse theory, having Drazin index less than or equal to 1 corresponds precisely to another special kind of generalized inverse called a *group inverse* [4, Def 7.2.4] – which can be defined without explicitly mentioning Drazin inverses. Borrowing this terminology, we introduce the notion of a †-*group inverse*.

Definition 2.7 *In a †-category, a* **†-group inverse** *of a map $f : A \to B$ is a map of dual type $f^\partial : B \to A$ such that:*

[G†.1.a] $ff^\dagger ff^\partial = ff^\dagger$

[G†.1.b] $f^\partial ff^\dagger f = f^\dagger f$

[G†.2] $f^\partial ff^\partial = f^\partial$;

[G†.3] $(ff^\partial)^\dagger = ff^\partial$;

[G†.4] $(f^\partial f)^\dagger = f^\partial f$.

Lemma 2.8 *In a †-category, a map $f : A \to B$ has a †-group inverse if and only if f is †-Drazin with $\mathsf{ind}^\partial(f) \leq 1$. Moreover, in this case, the †-group inverse coincides with the †-Drazin inverse, and the following equalities also hold:*

[G†.1.c] $ff^\partial ff^\dagger = ff^\dagger$

[G†.1.d] $f^\dagger ff^\partial f = f^\dagger f$

PROOF: For the $\Rightarrow$ direction, suppose that f has a †-group inverse f^∂. Then we clearly see that **[G†.1.a]** and **[G†.1.b]** combined is **[D†.1]** for $k = 1$, while **[G†.2]**, **[G†.3]**, and **[G†.4]** are precisely **[D†.2]**, **[D†.3]**, and **[D†.4]** respectively. So f is †-Drazin with †-Drazin being its †-group inverse, and moreover **[G†.1.a]** and **[G†.1.b]** tells us precisely that $\mathsf{ind}^\partial(f) \leq 1$. For the $\Leftarrow$ direction, suppose that f is †-Drazin with †-Drazin inverse f^∂ $\mathsf{ind}^\partial(f) \leq 1$. Then clearly, **[D†.2]**, **[D†.3]**, and **[D†.4]** are the same as **[G†.2]**, **[G†.3]**, and **[G†.4]** respectively, while $\mathsf{ind}^\partial(f) \leq 1$ tells us that **[D†.1]** holds for $k = 1$ which is precisely **[G†.1.a]** and **[G†.1.b]**. Lastly, it follows from Lemma 2.3 that **[G†.1.c]** and **[G†.1.d]** hold, as they are **[D†.5]** and **[D†.8]** for $k = 1$. □

Since †-group inverses are special cases of †-Drazin inverses, they are unique as well. Moreover, Lemma 2.5.(ii), (iii), and (v) tells us that isomorphisms (in particular, identity maps and unitary maps), partial isometries, and †-idempotents all have †-group inverses. Moreover, observe that by combining Lemma 2.5.(ii) and Prop 2.6.(vii), we see that a map f is †-Drazin with $\mathsf{ind}^\partial(f) = 0$ if and only if f is an isomorphism, and in this case its †-group inverse is its inverse, $f^\partial = f^{-1}$. As such, the maps with †-Drazin index 1 are precisely those that have a †-group inverse and are *not* an isomorphism.

3 Comparing Dagger-Drazin and Drazin

In this section, we explain the relationship between Drazin inverses and †-Drazin inverses. In particular, the main result of this section is that having †-Drazin inverses is equivalent to having Drazin inverses of positive maps. For a full detailed introduction to Drazin inverses in categories, see [9].

Definition 3.1 *[9, Def 2.1] In a category, a **Drazin inverse** of an endomorphism $x : A \to A$ is an endomorphism $x^D : A \to A$ such that:*

[D.1] *There is a $k \in \mathbb{N}$ such that $x^{k+1}x^D = x^k$;*

[D.2] $x^D x x^D = x^D$;

[D.3] $xx^D = x^D x$.

*If x has a Drazin inverse, we say that x is **Drazin invertible** or simply **Drazin**. We call the smallest natural number k such that **[D.1]** holds the **Drazin index of** x, which we denote by $\mathsf{ind}^D(x)$. A **Drazin category** is a category such that every map is Drazin.*

Drazin inverses are *unique* [9, Prop 2.3], so again we may speak of *the* Drazin inverse of a map and that being Drazin is a property rather than structure. Many examples and properties of Drazin inverses can be found in [9], including the fact that having a Drazin index less than or equal to 1 corresponds precisely to the notion of a *group inverse*:

Definition 3.2 *[9, Def 3.8] In a category, a **group inverse** of an endomorphism $x : A \to A$ is an endomorphism $x^D : A \to A$ such that:*

[G.1] $xx^D x = x$;

[G.2] $x^D x x^D = x^D$;

[G.3] $x^D x = xx^D$

In [9, Lemma 3.9] it is proven that x is Drazin with $\mathsf{ind}(x) \leq 1$ if and only if x has a group inverse . As such, a group inverse is the Drazin inverse for endomorphisms with Drazin index less than or equal to 1.

We now add dagger structure back into the story. Thankfully, Drazin inverses behave well in a †-category.

Lemma 3.3 *[9, Lemma 7.22] In a †-category, an endomorphism x is Drazin if and only if $x^\dagger$ is Drazin. Moreover, if x is Drazin then $x^{\dagger D} = x^{D\dagger}$.*

Let us now begin exploring the connection between Drazin inverses and †-Drazin inverses. We first observe that for *self-adjoint* maps[2], the two notions coincide.

Lemma 3.4 *In a †-category, a self-adjoint map x (that is, $x = x^\dagger$) is †-Drazin if and only if x is Drazin. Moreover, if x is self-adjoint and (†-)Drazin, then its (†-)Drazin inverse is also self-adjoint and $x^D = x^\partial$.*

PROOF: Let x be self-adjoint, so $x = x^\dagger$. For $\Rightarrow$, suppose that x is †-Drazin. We will show that its †-Drazin inverse x^∂ is also its Drazin inverse. Now since x is self-adjoint, we can rewrite **[D†.6]** as saying there exists a $k \in \mathbb{N}$ such that $x^{2k+1}x^\partial = x^{2k}$, which implies that **[D.1]** holds. Note that **[D†.2]** and **[D.2]** are the same. Next, observe that by Prop 2.6.(iii), we have that $x^{\partial\dagger} = x^{\dagger\partial} = x^\partial$, so x^∂ is self-adjoint. Therefore, we may rewrite both **[D†.3]** and **[D†.4]** as $xx^\partial = x^\partial x$, which is precisely **[D.3]**. Therefore,

[2] Note that self-adjoint maps are always endomorphisms.

we conclude that x is Drazin with Drazin inverse $x^D = x^\partial$. For $\Leftarrow$, suppose that x is Drazin. We will show that its Drazin inverse x^D is also its †-Drazin inverse. Now if $\mathsf{ind}^D(x) = k$, [9, Lemma 2.2.(i)] tells us that for all $n \geq k$, we have that $x^{n+1}x^D = x^n$. In particular this gives us that $x^{2k+1}x^D = x^{2k}$, which since x is self-adjoint is precisely **[D†.6]**. As before, note that **[D.2]** and **[D†.2]** are the same. Next, by Lemma 3.3, since x is self-adjoint, x^D is also self-adjoint. Then by self-adjointness and **[D.3]**, we have that $(xx^D)^\dagger = x^Dx = xx^D = (x^Dx)^\dagger$, which gives both **[D†.3]** and **[D†.4]**. Therefore, we conclude that x is †-Drazin with †-Drazin inverse $x^\partial = x^D$. □

An important class of self-adjoint maps are the *positive* maps. Recall that in a †-category, a **positive map** is an endomorphism $p : A \to A$ such that there exists a map $f : A \to B$ such that $p = ff^\dagger$.

Corollary 3.5 *In a †-category, a positive map is †-Drazin if and only if it is Drazin.*

Now for any map f in a †-category, we have two positive maps associated to it: $ff^\dagger$ and $f^\dagger f$. We will now show that f being †-Drazin is equivalent to either one of these positive maps being Drazin.

Theorem 3.6 *In a †-category, for a map f, the following are equivalent:*

(i) *f is †-Drazin;*

(ii) *$f^\dagger$ is †-Drazin;*

(iii) *$ff^\dagger$ is (†-)Drazin;*

(iv) *$f^\dagger f$ is (†-)Drazin.*

Moreover, if f is †-Drazin, then:

(vi) $f^\partial = f^\dagger(ff^\dagger)^D = (f^\dagger f)^D f^\dagger$ *and* $f^{\dagger\partial} = f(f^\dagger f)^D = (ff^\dagger)^D f$;

(vii) $(ff^\dagger)^D = f^{\dagger\partial} f^\partial$ *and* $(f^\dagger f)^D = f^\partial f^{\dagger\partial}$;

(viii) $\mathsf{ind}^\partial(f) = \mathsf{max}\{\mathsf{ind}^D(ff^\dagger), \mathsf{ind}^D(f^\dagger f)\}$.

PROOF: First note that $(i) \Leftrightarrow (ii)$ follows from Prop 2.6.(iii). On the other hand, $(iii) \Leftrightarrow (iv)$ follows from the fact in any category, for maps $g : A \to B$ and $h : B \to A$, gh is Drazin if and only if hg is Drazin [9, Lemma 7.3]. Then combining this fact with Cor 3.5 we get that $ff^\dagger$ is (†-)Drazin if and only if $f^\dagger f$ is (†-)Drazin. So to prove the desired statement, it remains to show that $(i) \Leftrightarrow (iii)$.

For $(i) \Rightarrow (iii)$, suppose that f is †-Drazin. We will show that $(ff^\dagger)^D := f^{\partial\dagger} f^\partial$ satisfies the three axioms of a Drazin inverse.

[D.1] $(ff^\dagger)^{k+1}(ff^\dagger)^D \overset{\text{def.}}{=} (ff^\dagger)^{k+1} f^{\partial\dagger} f^\partial = (ff^\dagger)^k ff^\dagger f^{\partial\dagger} f^\partial \overset{\text{Prop. 2.6.(ii)}}{=} (ff^\dagger)^k ff^\partial \overset{\textbf{[D}^\dagger\textbf{.6]}}{=} (ff^\dagger)^k$

[D.2] $(ff^\dagger)^D ff^\dagger (ff^\dagger)^D \overset{\text{def.}}{=} f^{\partial\dagger} f^\partial ff^\dagger f^{\partial\dagger} f^\partial \overset{\text{Prop. 2.6.(ii)}}{=} f^{\partial\dagger} f^\partial ff^\partial \overset{\textbf{[D}^\dagger\textbf{.2]}}{=} f^{\partial\dagger} f^\partial \overset{\text{def.}}{=} (ff^\dagger)^D$

[D.3] $(ff^\dagger)^D ff^\dagger \overset{\text{def.}}{=} f^{\partial\dagger} f^\partial ff^\dagger \overset{\text{Prop. 2.6.(ii)}}{=} ff^\partial f^{\partial\dagger} f^\dagger \overset{\text{Prop. 2.6.(ii)}}{=} ff^\dagger f^{\partial\dagger} f^\partial \overset{\text{def.}}{=} ff^\dagger (ff^\dagger)^D$

So we conclude that $ff^\dagger$ Drazin.

For $(iii) \Rightarrow (i)$, suppose that $ff^\dagger$ is Drazin. We will show that $f^\partial := f^\dagger(ff^\dagger)^D$ satisfies the necessary axioms to be a †-Drazin inverse of f.

[D†.6] $(ff^\dagger)^k ff^\partial \overset{\text{def.}}{=} (ff^\dagger)^k ff^\dagger (ff^\dagger)^D = (ff^\dagger)^{k+1}(ff^\dagger)^D \overset{\textbf{[D.1]}}{=} (ff^\dagger)^k$

[D†.2] $f^\partial ff^\partial \overset{\text{def.}}{=} f^\dagger(ff^\dagger)^D ff^\dagger (ff^\dagger)^D \overset{\textbf{[D.2]}}{=} f^\dagger(ff^\dagger)^D \overset{\text{def.}}{=} f^\partial$

For the remaining two axioms, recall that by Lemma 3.3, since $ff^\dagger$ is self-adjoint then so is $(ff^\dagger)^D$.

[D†.3] $(ff^\partial)^\dagger \overset{\text{def.}}{=} (ff^\dagger(ff^\dagger)^D)^\dagger = ff^\dagger(ff^\dagger)^D \overset{\textbf{[D.3]}}{=} ff^\dagger(ff^\dagger)^D \overset{\text{def.}}{=} ff^\partial$

[D†.4] $(f^\partial f)^\dagger \overset{\text{def.}}{=} (f^\dagger(ff^\dagger)^D f)^\dagger = f^\dagger(ff^\dagger)^D f \overset{\text{def.}}{=} f^\partial f$

So we conclude that f is †-Drazin. Therefore, we have that $(i) \Leftrightarrow (ii) \Leftrightarrow (iii) \Leftrightarrow (iv)$ as desired.

Now for (vi), we have already shown that $f^\partial = f^\dagger(ff^\dagger)^D$. For the other part of the equality we will need the only following fact: for maps $g : A \to B$ and $h : B \to A$ such that gh and hg are Drazin, then $(gh)^D g = g(hg)^D$ and $h(gh)^D = (hg)h^D$ [9, Lemma 7.5]. Therefore, we indeed have that $f^\partial = f^\dagger(ff^\dagger)^D = (f^\dagger f)^D f^\dagger$. Then by Prop. 2.6.(iii) and Lemma 3.3, we also get that ${f^\dagger}^\partial = f(f^\dagger f)^D = (ff^\dagger)^D f$. For (vii), we have already shown that $(ff^\dagger)^D = {f^\partial}^\dagger f^\partial$, so by Prop. 2.6.(iii) we get that $(ff^\dagger)^D = {f^\dagger}^\partial f^\partial$, and then applying the same formula to $f^\dagger$, we get that $(f^\dagger f)^D = f^\partial {f^\dagger}^\partial$. Lastly, by the arguments made in the above computations, it is straightforward to check that (viii) holds. □

It immediately follows that, unsurprisingly, having a †-group inverse corresponds to both induced positive maps having a group inverse.

Corollary 3.7 *In a †-category, a map f has a †-group inverse if and only if $ff^\dagger$ and $f^\dagger f$ have group inverses.*

PROOF: For the $\Rightarrow$ direction, suppose that f has a †-group inverse. Which implies that f is †-Drazin with $\mathsf{ind}^\partial(f) \leq 1$. Then by Thm 3.6, we have that $ff^\dagger$ and $f^\dagger f$ are Drazin, and also that $\mathsf{ind}^D(ff^\dagger) \leq \mathsf{ind}^\partial(f) \leq 1$ and $\mathsf{ind}^D(f^\dagger f) \leq \mathsf{ind}^\partial(f) \leq 1$. This implies that $ff^\dagger$ and $f^\dagger f$ have group inverses. Conversely, for the $\Leftarrow$ direction, suppose that $ff^\dagger$ and $f^\dagger f$ both have group inverses. So they are both Drazin and $\mathsf{ind}^D(ff^\dagger) \leq 1$ and $\mathsf{ind}^D(f^\dagger f) \leq 1$. Then by Thm 3.6, we have that f is †-Drazin and $\mathsf{ind}^\partial(f) = \mathsf{max}\{\mathsf{ind}^D(ff^\dagger), \mathsf{ind}^D(f^\dagger f)\} \leq 1$. Therefore, f has a †-group inverse. □

We also obtain an equivalent characterization of †-Drazin categories:

Corollary 3.8 *A †-category is †-Drazin if and only if every positive map is Drazin.*

This also allows us to conclude that a Drazin category, which has a †, is automatically a †-Drazin category.

Corollary 3.9 *Every Drazin †-category (by which we mean a †-category whose underlying category is Drazin) is †-Drazin. In particular, if $\mathbb{X}$ is a Drazin category, then for any dagger † on $\mathbb{X}$, $\mathbb{X}$ is †-Drazin.*

We are now in a position to give examples of †-Drazin inverses. Our first examples are matrices over a field. For any field, every matrix over the field has a †-Drazin inverse with respect to the transpose operator. Moreover, a matrix over an involutive field also has †-Drazin inverse with respect to the conjugate transpose operator.

Example 3.10 *For a field $\mathbb{K}$, let $\mathsf{MAT}(\mathbb{K})$ be the category whose objects are natural numbers $n \in \mathbb{N}$ and where a map $A : n \to m$ is an $n \times m$ $\mathbb{K}$-matrix. Every square $\mathbb{K}$-matrix has a Drazin inverse, and therefore $\mathsf{MAT}(\mathbb{K})$ is a Drazin category – see [9, Sec 3.2] for how to construct the Drazin inverse. Moreover, the Drazin index of a square matrix B corresponds precisely to the **index** of B [4, Def 7.2.1], that is, the least $k \in \mathbb{N}$ such that $\mathsf{rank}(B^{k+1}) = \mathsf{rank}(B^k)$. On the other hand, $\mathsf{MAT}(\mathbb{K})$ is a †-category whose dagger is the transpose operator T, $A^\mathsf{T}(i,j) = A(j,i)$. As such, $\mathsf{MAT}(\mathbb{K})$ is a Drazin †-category, and so in particular also †-Drazin. So every $\mathbb{K}$-matrix A has a T-Drazin inverse given by $A^\partial = A^\mathsf{T}(AA^\mathsf{T})^D$. Furthermore, the T-Drazin index of a matrix A is the maximum between the index of AA^T and the index of $A^\mathsf{T}A$.*

Example 3.11 *If $\mathbb{K}$ is an involutive field, with involution $x \mapsto \overline{x}$, then $\mathsf{MAT}(\mathbb{K})$ has another dagger this time given by the conjugate transpose operator $*$, $A^*(i,j) = \overline{A(j,i)}$. Then $\mathsf{MAT}(\mathbb{K})$ is still a Drazin †-category with respect to $*$, and so in particular also †-Drazin. So every $\mathbb{K}$-matrix A has a $*$-Drazin inverse given by $A^\partial = A^*(AA^*)^D$, and where the $*$-Drazin index of A is the maximum between the index of AA^* and the index of A^*A. We will revisit this example when $\mathbb{K}$ is the involutive field of complex numbers in Ex 4.4 below.*

The converse of Cor 3.9 is not true, that is, there are †-Drazin categories that are not Drazin. An example arises from considering *inverse categories*.

Example 3.12 *An **inverse category** may be viewed as a †-category in which every map is a partial isometry (that is, $ff^\dagger f = f$) and for all parallel maps f and g, $ff^\dagger gg^\dagger = gg^\dagger ff^\dagger$. Recall that by Lemma 2.5.(iii), every partial isometry is †-Drazin with †-Drazin inverse being its adjoint and †-Drazin index less than or equal to 1. Therefore every inverse category is †-Drazin.*

Example 3.13 *Let PINJ be the category of sets and partial injections. Then PINJ is an inverse category where for a partial injection $f : X \to Y$, $f^\circ : Y \to X$ is defined as $f^\circ(y) = x$ if $f(x) = y$ and is undefined otherwise. As PINJ is an inverse category it is certainly †-Drazin, and so every partial injection f is †-Drazin with $f^\partial = f^\circ$. However, PINJ is not Drazin. Consider the successor function $s : \mathbb{N} \to \mathbb{N}$, $s(n) = n+1$. Suppose that s had a Drazin inverse $s^D : \mathbb{N} \to \mathbb{N}$ and $\mathsf{ind}(s) = k$. By **[D.3]**, we would have that $s^D(n) = s^D(s^n(0)) = s^n(s^D(0)) = s^D(0) + n$. So s^D is completely determined by $s^D(0)$. If $s^D(0)$ is defined, then by **[D.1]** we compute that $k = s^k(0) = s^D(s^{k+1}(0)) = s^D(k+1) = s^D(0) + k + 1$. This implies that $0 = s^D(0) + 1$ – which is a contradiction since $s^D(0) \in \mathbb{N}$. On the other hand if $s^D(0)$ is undefined, then $s^k(0) = s^D(s^{k+1}(0)) = s^{k+1}(s^D(0))$ would also be undefined, but this is a contradiction since s^k is total. So the successor function does not have a Drazin inverse, and therefore PINJ is not Drazin.*

Even in †-categories that are neither Drazin nor †-Drazin, it is sometimes possible to characterize the maps that are †-Drazin.

Example 3.14 *Let HILB be the category of (complex) Hilbert spaces and bounded linear operators between them. Then HILB is a †-category where the dagger is given by the adjoint $*$, that is, for a bounded linear operator $f : H_1 \to H_2$, $f^* : H_2 \to H_1$ is the unique bounded linear operator such that $\langle f(x)|y\rangle = \langle x|f^*(y)\rangle$ for all $x \in H_1$ and $y \in H_2$. HILB is not Drazin, nor is it †-Drazin. Nevertheless, we can still characterize the $*$-Drazin maps in HILB, since there is a characterization of when a bounded linear operator is Drazin. For a bounded linear operator $f : H_1 \to H_2$, let $\mathsf{N}(f)$ denote its null space and $\mathsf{R}(f)$ its range space. A bounded linear operator $g : H \to H$ is said to be has **finite ascent** (resp. **descent**) if there exists a $k \in \mathbb{N}$ such that $\mathsf{N}(g^k) = \mathsf{N}(g^{k+1})$ (resp. $\mathsf{R}(g^k) = \mathsf{R}(g^{k+1})$), and the least such k is called the **ascent** (resp. **descent**) of g. Then a bounded linear operator $g : H \to H$ is Drazin if and only if it has both finite ascent and finite descent [15, 29], and furthermore its Drazin index corresponds to its ascent (or equivalently its descent, which are equal in this). Therefore, it follows that a bounded linear operator $f : H_1 \to H_2$ is $*$-Drazin if and only if ff^* (or equivalently f^*f) has both finite ascent and finite descent. In this case, the $*$-Drazin of f is equal to the maximum of the ascent (or equivalently descent) of ff^* or f^*f.*

4 Comparing Dagger-Drazin and Moore-Penrose

In this section, we explain the relationship between †-Drazin inverses and Moore-Penrose inverses. In particular, the main result of this section is that having a Moore-Penrose inverse is equivalent to being *a*

†-Drazin inverse (which we stress is different than simply being †-Drazin inverse and having a †-Drazin inverse). For a detailed introduction to Moore-Penrose inverses in dagger categories, we invite the reader to see [8].

Definition 4.1 *[8, Def 2.3] In a †-category, a* ***Moore-Penrose inverse*** *of a map $f : A \to B$ is a map of dual type $f^\circ : B \to A$ such that:*

[MP.1] $ff^\circ f = f$

[MP.2] $f^\circ ff^\circ = f^\circ$

[MP.3] $(ff^\circ)^\dagger = ff^\circ$

[MP.4] $(f^\circ f)^\dagger = f^\circ f$

If f has a Moore-Penrose inverse, we say that f is ***Moore-Penrose invertible*** *or simply* ***Moore-Penrose****. A* ***Moore-Penrose †-category*** *is a †-category such that every map is Moore-Penrose.*

Moore-Penrose inverses are *unique* [8, Lemma 2.4], so again we may speak of *the* Moore-Penrose inverse of a map and that being Moore-Penrose is a property rather than structure. Many examples and properties of Moore-Penrose inverses and Moore-Penrose dagger categories can be found in [8].

We now show that having a Moore-Penrose inverse is equivalent to being a †-Drazin inverse. It is important to note that being a †-Drazin inverse is strictly stronger than being †-Drazin. Indeed, while every †-Drazin inverse is †-Drazin, not every †-Drazin map is itself a †-Drazin inverse. A †-Drazin map is a †-Drazin inverse precisely when it is the †-Drazin inverse of its †-Drazin inverse.

Theorem 4.2 *In a †-category, the following are equivalent for a map f:*

(i) *f is Moore-Penrose;*

(ii) *f is a †-Drazin inverse;*

(iii) *f is †-Drazin and $f^{\partial\partial} = f$.*

Therefore, if f is Moore-Penrose then f is †-Drazin where $f^\partial = f^\circ$ and $\mathsf{ind}^\partial(f) \leq 1$ (in other words, f° is the †-group inverse of f), and, moreover, f is the †-Drazin inverse of f°, that is, $f^{\circ\partial} = f$.

PROOF: We will prove $(i) \Rightarrow (iii) \Rightarrow (ii) \Rightarrow (i)$. So for $(i) \Rightarrow (iii)$, suppose that f is Moore-Penrose. We will show that its Moore-Penrose inverse f° is also its †-Drazin inverse. However observe that **[D†.2]** is **[MP.1]**, **[D†.3]** is **[MP.4]**, and **[D†.4]** is **[MP.3]**. It remains to show **[D†.5]**. So set $j = 1$, then we easily have that $ff^\partial ff^\dagger \overset{\text{def.}}{=} ff^\circ ff^\dagger \overset{\textbf{[MP.1]}}{=} ff^\dagger$. So f° is the †-Drazin inverse of f. Lastly, by Prop 2.6.(iv), $f^{\partial\partial} = f$ is precisely **[MP.1]**. Next note $(iii) \Rightarrow (ii)$ is automatic by assumption that $f^{\partial\partial} = f$. So it remains to show that $(ii) \Rightarrow (i)$. So suppose that f is a †-Drazin inverse for map g, that is, $f = g^\partial$. By Prop 2.6.(iv), we know that f is also †-Drazin where $f^\partial = gfg$. We will show that f^∂ is also the Moore-Penrose inverse of f. To do so, recall that by Prop 2.6.(v) that $f^{\partial\partial} = g^{\partial\partial\partial} = g^\partial = f$. So f is the †-Drazin inverse of f^∂. Since f is the †-Drazin inverse of f^∂, by **[D†.2]** we have that $ff^\partial f = f$, which is **[MP.1]**. On the other hand, since f^∂ is the †-Drazin inverse of f, **[D†.2]** gives us $f^\partial ff^\partial = f^\partial$, which is **[MP.2]**. Lastly, **[MP.3]** and **[MP.4]** are precisely **[D†.3]** and **[D†.4]** as well. So f is Moore-Penrose. □

This also allows us to give an equivalent characterization of Moore-Penrose † categories as special kind of †-Drazin category.

Corollary 4.3 *A †-category is Moore-Penrose if and only if it is †-Drazin where every map is also a †-Drazin inverse.*

We already know from Ex 3.11 that complex matrices have †-Drazin inverses (with respect to the conjugate transpose dagger). However we now have an alternative explanation as to why, and an alternative description of its †-Drazin inverse from the perspective of Moore Penrose inverse.

Example 4.4 *Let $\mathbb{C}$ be the field of complex numbers. $\mathsf{MAT}(\mathbb{C})$ with the conjugate transpose dagger $*$ is a Moore-Penrose †-category where the Moore-Penrose inverse of a complex matrix can be constructed from its singular value decomposition [8, Ex 2.9]. Therefore, every complex matrix is $*$-Drazin and its $*$-Drazin inverse is its Moore-Penrose inverse. Of course, we can also say that every complex matrix is the $*$-Drazin inverse of its Moore-Penrose inverse.*

It is important to note that while for any (involutive) field, its category of matrices is always †-Drazin, it may not be Moore-Penrose.

Example 4.5 *Consider $\mathsf{MAT}(\mathbb{C})$ this time with simply the transpose T dagger, which we know is †-Drazin from Ex 3.10, that is, every complex matrix has a T-Drazin inverse. However, $\mathsf{MAT}(\mathbb{C})$ is not Moore-Penrose with the transpose dagger, in particular, the matrix $\begin{bmatrix} i & 1 \end{bmatrix}$ does not have a Moore-Penrose inverse with respect to the transpose [8, Ex 2.10]. However $\begin{bmatrix} i & 1 \end{bmatrix}$ does have a T-Drazin inverse. In fact, since $\begin{bmatrix} i & 1 \end{bmatrix} \begin{bmatrix} i & 1 \end{bmatrix}^{\mathsf{T}} = 0$, it follows that $\begin{bmatrix} i & 1 \end{bmatrix}^{\partial} = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathsf{T}}$.*

Recall that in Ex 3.14, we characterized †-Drazin maps in the category of Hilbert spaces. We can now take a look at the subcategory of finite dimensional Hilbert spaces.

Example 4.6 HILB *is not Moore-Penrose but we can again characterize when bounded linear maps are Moore-Penrose – they are precisely those whose range is closed [8, Ex 2.12]. Moreover, for appropriate bounded linear maps, it is sometimes possible to formulate the Drazin inverse (and hence the †-Drazin inverse) in terms of the Moore-Penrose inverse [29, Lemma 2.1]. Now let* FHILB *be the subcategory of finite dimensional Hilbert spaces.* FHILB *is a Moore-Penrose †-category [8, Ex 2.12], and therefore* FHILB *is also †-Drazin. So every linear operator between finite dimensional Hilbert spaces is $*$-Drazin, where its $*$-Drazin inverse is its Moore-Penrose inverse.*

5 Revisiting Drazin Opposing Pairs

In [9], the authors introduced the notion of Drazin inverses for opposing pairs, which was the authors first attempt to generalize the notion of Drazin inverses to maps of arbitrary type. In section, we relate Drazin inverse of opposing pairs to †-Drazin inverses. In particular, we show that Drazin inverse of opposing pairs correspond precisely to †-Drazin inverses in *cofree* dagger categories.

It is useful to understand the original motivation for considering Drazin inverses of opposing pairs. A fundamental idea behind Drazin inverses is that certain endomorphisms can be undone after iterating them a certain number of times. Thus if one wishes to give a generalization of a Drazin inverse for a map $f : A \to B$ in an arbitrary category, then one also needs a map of dual type $g : B \to A$ so that the Drazin iteration axiom can be expressed. Thus, in an arbitrary category $\mathbb{X}$, an **opposing pair** $(f,g) : A \to B$ is a pair (f,g) consisting of maps of dual type, so $f : A \to B$ and $g : B \to A$. Such an opposing pair, $(f,g) : A \to B$, induces two endomorphism $fg : A \to A$ and $gf : B \to B$ for which we can seek Drazin inverses. These in turn – if they exist – induce an opposing pair in the opposite direction. This provides a notion of inverse for an arbitrary pairs of opposing maps.

Definition 5.1 *[9, Def 7.1] In a category $\mathbb{X}$, a **Drazin inverse** of an opposing pair $(f,g) : A \to B$ is an opposing pair of dual type $(f,g)^D = (f^{\frac{D}{g}}, g^{\frac{D}{f}}) : B \to A$ such that:*

[DV.1] *There is a* $k \in \mathbb{N}$ *such that* $(fg)^k f f^{\frac{D}{g}} = (fg)^k$ *and* $(gf)^k g g^{\frac{D}{f}} = (gf)^k$*;*

[DV.2] $f^{\frac{D}{g}} f f^{\frac{D}{g}} = f^{\frac{D}{g}}$ *and* $g^{\frac{D}{f}} g g^{\frac{D}{f}} = g^{\frac{D}{f}}$*;*

[DV.3] $f f^{\frac{D}{g}} = g^{\frac{D}{f}} g$ *and* $f^{\frac{D}{g}} f = g g^{\frac{D}{f}}$*.*

We say that an opposing pair (f,g) *is* ***Drazin*** *if it has a Drazin inverse. The map* $f^{\frac{D}{g}} : B \to A$ *is called the* ***Drazin inverse of*** f ***over*** g*, while the map* $g^{\frac{D}{f}} : A \to B$ *is called the* ***Drazin inverse of*** g ***over*** f*. The* ***Drazin index*** *of* (f,g) *is the smallest* $k \in \mathbb{N}$ *such that* **[DV.1]** *holds, which we denote by* $\mathsf{ind}^D(f,g) = k$*.*

Drazin inverses of opposing pairs share many of the same properties as Drazin inverses of endomorphisms, see [9, Sec 7]. In particular, the Drazin inverse of an opposing pair is unique [9, Prop 7.7]. One can also recover Drazin inverses of endomorphisms by considering the Drazin inverse of the opposing pair consisting of an endomorphism and the identity [9, Lemma 7.10]. Moreover, since every opposing pair induces two endomorphisms, it turns out that Drazin opposing pairs can also be characterized in terms of Drazin inverses.

Theorem 5.2 *[9, Thm 7.6] In a category* $\mathbb{X}$*, the following are equivalent for an opposing pair* (f,g)*:*

(i) (f,g) *is Drazin;*

(ii) fg *is Drazin;*

(iii) gf *is Drazin.*

In particular, if (f,g) *is Drazin then [9, Cor 7.8]:*

(iv) $(fg)^D = g^{\frac{D}{f}} f^{\frac{D}{g}}$ *and* $(gf)^D = f^{\frac{D}{g}} g^{\frac{D}{f}}$*;*

(v) $f^{\frac{D}{g}} = g(fg)^D = (gf)^D g$ *and* $g^{\frac{D}{f}} = f(gf)^D = (fg)^D f$*;*

(vi) $\mathsf{ind}^D(f,g) = \mathsf{max}\{\mathsf{ind}^D(fg), \mathsf{ind}^D(gf)\}$*.*

Now in a †-category, every map and its adjoint form an opposing pair, which we call an **adjoint opposing pair**. Then, it is straightforward to see that a map being †-Drazin is equivalent to its adjoint opposing pair being Drazin.

Proposition 5.3 *In a †-category, a map* f *is †-Drazin if and only if the opposing pair* $(f, f^\dagger)$ *is Drazin. Moreover, if* f *is †-Drazin then* $f^{\frac{D}{f^\dagger}} = f^\partial$ *and* $f^\dagger{}^{\frac{D}{f}} = f^{\partial^\dagger}$*, and also that* $\mathsf{ind}^\partial(f) = \mathsf{ind}^D(f,g)$*.*

PROOF: By combining Thm 3.6 and Thm 5.2, we get that f is †-Drazin if and only if $ff^\dagger$ is Drazin if and only if $(f, f^\dagger)$ is Drazin. By Thm 3.6.(vi) and Thm 5.2.(v), we get that $f^{\frac{D}{f^\dagger}} = f^\dagger (ff^\dagger)^D = f^\partial$ and $f^\dagger{}^{\frac{D}{f}} = f(f^\dagger f)^D = f^{\dagger\partial} = f^{\partial^\dagger}$. Lastly, Thm 3.6.(viii) and Thm 5.2.(vi) give us that $\mathsf{ind}^\partial(f) = \mathsf{max}\{\mathsf{ind}^D(ff^\dagger), \mathsf{ind}^D(f^\dagger f)\} = \mathsf{ind}^D(f,g)$. □

We now turn our attention to proving that Drazin inverses of opposing pairs correspond to the †-Drazin inverses in *cofree* dagger categories. This makes sense since maps in cofree dagger categories are precisely opposing pairs [12, Def 3.1.16]. Indeed, given a category $\mathbb{X}$, let $\mathbb{X}^{\leftrightarrows}$ be the category that has the same objects as $\mathbb{X}$ and where a map is an opposing pair $(f,g) : A \to B$ in $\mathbb{X}$. Composition is defined as $(f,g)(h,k) = (fh, kg)$ and the identity is $(1_A, 1_A)$. Then $\mathbb{X}^{\leftrightarrows}$ is a †-category where $(f,g)^\dagger = (g,f)$. Moreover $\mathbb{X}^{\leftrightarrows}$ is the cofree †-category over $\mathbb{X}$ [12, Thm 3.1.17].

Theorem 5.4 *In a category* $\mathbb{X}$*, an opposing pair* (f,g) *in* $\mathbb{X}$ *is Drazin if and only if* (f,g) *is †-Drazin in* $\mathbb{X}^{\leftrightarrows}$*. Moreover in this case,* $(f,g)^D = (f,g)^\partial$ *and* $\mathsf{ind}^D(f,g) = \mathsf{ind}^\partial(f,g)$*.*

PROOF: For the $\Rightarrow$ direction, suppose that (f,g) is Drazin in $\mathbb{X}$. We show that $(f,g)^\partial = (f^{\frac{D}{g}}, g^{\frac{D}{f}})$ is the $\dagger$-Drazin inverse of (f,g) in $\mathbb{X}^{\leftrightarrows}$. We begin by checking **[D†.2]**–**[D†.4]**.

[D†.2] $(f,g)^\partial(f,g)(f,g)^\partial \overset{\text{def.}}{=} (f^{\frac{D}{g}}, g^{\frac{D}{f}})(f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (f^{\frac{D}{g}} f f^{\frac{D}{g}}, g^{\frac{D}{f}} g g^{\frac{D}{f}}) \overset{\textbf{[DV.2]}}{=} (f^{\frac{D}{g}}, g^{\frac{D}{f}}) \overset{\text{def.}}{=} (f,g)^\partial$

[D†.3] $(f,g)(f,g)^\partial \overset{\text{def.}}{=} (f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) \overset{\textbf{[DV.3]}}{=} (g^{\frac{D}{f}} g, f f^{\frac{D}{g}}) = (f f^{\frac{D}{g}}, g^{\frac{D}{f}} g)^\dagger = \big((f,g)(f,g)^\partial\big)^\dagger$

[D†.4] $(f,g)^\partial(f,g) \overset{\text{def.}}{=} (f^{\frac{D}{g}}, g^{\frac{D}{f}})(f,g) = (f^{\frac{D}{g}} f, g g^{\frac{D}{f}}) \overset{\textbf{[DV.3]}}{=} (g g^{\frac{D}{f}}, f^{\frac{D}{g}} f) = (f^{\frac{D}{g}} f, g g^{\frac{D}{f}})^\dagger = \big((f,g)^\partial(f,g)\big)^\dagger$

For **[D†.6]**, we first note that $fgff^{\frac{D}{g}} \overset{\textbf{[DV.3]}}{=} fgg^{\frac{D}{f}} g \overset{\textbf{[DV.3]}}{=} ff^{\frac{D}{g}} fg$. So $fgff^{\frac{D}{g}} = ff^{\frac{D}{g}} fg$. Therefore **[DV.1]** tells us that there is a $k \in \mathbb{N}$ such that $(fg)^k ff^{\frac{D}{g}} = (fg)^k = ff^{\frac{D}{g}} (fg)^k$. So then we compute that:

$$\begin{aligned}\big((f,g)(f,g)^\dagger\big)^k (f,g)(f,g)^\partial &\overset{\text{def.}}{=} ((f,g)(g,f))^k (f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (fg,fg)^k (f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) \\ &= ((fg)^k, (fg)^k)(f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) = ((fg)^k f f^{\frac{D}{g}}, g^{\frac{D}{f}} g (fg)^k) \\ &\overset{\textbf{[DV.3]}}{=} ((fg)^k f f^{\frac{D}{g}}, f f^{\frac{D}{g}} (fg)^k) \overset{\textbf{[DV.1]}}{=} ((fg)^k, (fg)^k) = \big((f,g)(f,g)^\dagger\big)^k\end{aligned}$$

So **[D†.6]** holds. So we conclude that (f,g) is $\dagger$-Drazin in $\mathbb{X}^{\leftrightarrows}$.

For the $\Leftarrow$ direction, suppose that (f,g) is $\dagger$-Drazin in $\mathbb{X}^{\leftrightarrows}$. Let $(f,g)^\partial$ be its $\dagger$-Drazin, which we denote as $(f,g)^\partial = (f^{\frac{D}{g}}, g^{\frac{D}{f}})$. We will show that (f,g) is Drazin in $\mathbb{X}$ where $(f^{\frac{D}{g}}, g^{\frac{D}{f}})$ is its Drazin inverse. To do so, let us expand out the $\dagger$-Drazin inverse axioms. Starting with **[D†.2]**, we get:

$$(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (f,g)^\partial \overset{\textbf{[D}^\dagger\textbf{.2]}}{=} (f,g)^\partial(f,g)(f,g)^\partial = (f^{\frac{D}{g}}, g^{\frac{D}{f}})(f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (f^{\frac{D}{g}} f f^{\frac{D}{g}}, g^{\frac{D}{f}} g g^{\frac{D}{f}})$$

This give us that $f^{\frac{D}{g}} = f^{\frac{D}{g}} f f^{\frac{D}{g}}$ and $g^{\frac{D}{f}} = g^{\frac{D}{f}} g g^{\frac{D}{f}}$, which is **[DV.2]**. Next we expand out **[D†.3]** and **[D†.4]**:

$$(f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) = (f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) = (f,g)(f,g)^\partial \overset{\textbf{[D}^\dagger\textbf{.3]}}{=} \Big((f,g)(f,g)^\partial\Big)^\dagger = (f f^{\frac{D}{g}}, g^{\frac{D}{f}} g)^\dagger = (g^{\frac{D}{f}} g, f f^{\frac{D}{g}})$$

$$(f^{\frac{D}{g}} f, g g^{\frac{D}{f}}) = (f^{\frac{D}{g}}, g^{\frac{D}{f}})(f,g) = (f,g)^\partial(f,g) \overset{\textbf{[D}^\dagger\textbf{.4]}}{=} \Big((f,g)^\partial(f,g)\Big)^\dagger = (f^{\frac{D}{g}} f, g g^{\frac{D}{f}})^\dagger = (g g^{\frac{D}{f}}, f^{\frac{D}{g}} f)$$

So we get that $f f^{\frac{D}{g}} = g^{\frac{D}{f}} g$ and $f^{\frac{D}{g}} f = g g^{\frac{D}{f}}$, so **[DV.3]** holds. Next we expand out **[D†.6]** and **[D†.7]**:

$$\begin{aligned}&((fg)^k, (fg)^k) = \big((f,g)(f,g)^\dagger\big)^k \overset{\textbf{[D}^\dagger\textbf{.6]}}{=} \big((f,g)(f,g)^\dagger\big)^k (f,g)(f,g)^\partial = ((f,g)(g,f))^k (f,g)(f^{\frac{D}{g}}, g^{\frac{D}{f}}) \\ &= (fg,fg)^k (f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) = ((fg)^k, (fg)^k)(f f^{\frac{D}{g}}, g^{\frac{D}{f}} g) = ((fg)^k f f^{\frac{D}{g}}, g^{\frac{D}{f}} g (fg)^k) = ((fg)^k f f^{\frac{D}{g}}, f f^{\frac{D}{g}} (fg)^k)\end{aligned}$$

$$\begin{aligned}&((gf)^k, (gf)^k) = \big((f,g)^\dagger(f,g)\big)^k = (f,g)^\partial(f,g)\big((f,g)^\dagger(f,g)\big)^k \overset{\textbf{[D}^\dagger\textbf{.7]}}{=} (f^{\frac{D}{g}}, g^{\frac{D}{f}})(f,g)((g,f)(f,g))^k \\ &= (f^{\frac{D}{g}} f, g g^{\frac{D}{f}})(gf,gf)^k = (f^{\frac{D}{g}} f, g g^{\frac{D}{f}})((gf)^k, (gf)^k) = (f^{\frac{D}{g}} f (gf)^k, (gf)^k g g^{\frac{D}{f}}) = (g g^{\frac{D}{f}} (gf)^k, (gf)^k g g^{\frac{D}{f}})\end{aligned}$$

So we get that $(fg)^k f f^{\frac{D}{g}} = (fg)^k$ and $(gf)^k g g^{\frac{D}{f}} = (gf)^k$, so **[DV.1]** holds. So we conclude that (f,g) is Drazin in $\mathbb{X}$. $\square$

References

[1] G. Azumaya (1954): *Strongly π-regular rings*. Journal of the Faculty of Science Hokkaido University. Ser. 1 Mathematics 13(1), pp. 034–039. Available at http://hdl.handle.net/2115/55983.

[2] O. M. Baksalary & G. Trenkler (2021): *The Moore–Penrose inverse: a hundred years on a frontline of physics research*. The European Physical Journal H 46, pp. 1–10, doi:10.1140/epjh/s13129-021-00011-y.

[3] M. Bartha (2014): *Quantum Turing automata*. In Benedikt Löwe & Glynn Winskel, editors: Proceedings 8th International Workshop on Developments in Computational Models, Cambridge, United Kingdom, 17 June 2012, Electronic Proceedings in Theoretical Computer Science 143, Open Publishing Association, pp. 17–31, doi:10.4204/EPTCS.143.2.

[4] S. Campbell & C. Meyer (2009): *Generalized inverses of linear transformations*. SIAM, doi:10.1137/1.9780898719048.

[5] N. Cao, J. Lin, D. Kribs, Y.-T. Poon, B. Zeng & R. Laflamme (2021): *NISQ: Error correction, mitigation, and noise simulation*. arXiv preprint arXiv:2111.02345.

[6] J. Chen, Y. Wang, J. Chen & S. Su (2023): *Optimal figure of merit of low-dissipation quantum refrigerators*. Physical Review E 107(4), p. 044118, doi:10.1103/PhysRevE.107.044118.

[7] J.-F. Chen, C. Sun & H. Dong (2021): *Extrapolating the thermodynamic length with finite-time measurements*. Physical Review E 104(3), p. 034117, doi:10.1103/PhysRevE.104.034117.

[8] J.R.B. Cockett & J-S. P. Lemay (2023): *Moore-Penrose Dagger Categories*. In: Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023, Electronic Proceedings in Theoretical Computer Science 384, Open Publishing Association, pp. 171–186, doi:10.4204/EPTCS.384.10.

[9] R. Cockett, J.-S. P. Lemay & P. V. Srinivasan (2024): *Drazin Inverses in Categories*. Theory and Applications of Categories 43(14), pp. 455–519. Available at http://www.tac.mta.ca/tac/volumes/43/14/43-14abs.html.

[10] M. P. Drazin (1958): *Pseudo-inverses in associative rings and semigroups*. The American mathematical monthly 65(7), pp. 506–514, doi:10.2307/2308576.

[11] H. Fitting (1933): *Die Theorie der Automorphismenringe Abelscher Gruppen und ihr Analogon bei nicht kommutativen Gruppen*. Mathematische Annalen 107, pp. 514–542, doi:10.1007/BF01448909.

[12] C. Heunen (2009): *Categorical quantum models and logics*. Amsterdam University Press, doi:10.5117/9789085550242.

[13] C. Heunen & J. Vicary (2019): *Categories for Quantum Theory: an Introduction*. Oxford University Press, doi:10.1093/oso/9780198739623.001.0001.

[14] B. Jacobs, C. Heunen & I. Hasuo (2009): *Categorical semantics for arrows*. Journal of functional programming 19(3-4), pp. 403–438, doi:10.1017/S0956796809007308.

[15] C. F. King (1977): *A note on Drazin inverses*. Pacific Journal of Mathematics 70, pp. 383–390, doi:10.2140/pjm.1977.70.383.

[16] T. Leinster (2023): *The eventual image*. Theory and Applications of Categories. Available at http://www.tac.mta.ca/tac/volumes/42/9/42-09abs.html.

[17] H. Miller, M. Scandi, J. Anders & M. Perarnau-Llobet (2019): *Work fluctuations in slow processes: quantum signatures and optimal control*. Physical review letters 123(23), p. 230603, doi:10.1103/PhysRevLett.123.230603.

[18] E. H. Moore (1920): *On the reciprocal of the general algebraic matrix*. Bull. Am. Math. Soc. 26, pp. 394–395, doi:10.1090/S0002-9904-1920-03322-7.

[19] W. D. Munn (1961): *Pseudo-inverses in semigroups*. In: Mathematical Proceedings of the Cambridge Philosophical Society, 57, Cambridge University Press, pp. 247–250, doi:10.1017/S0305004100035143.

[20] R. Penrose (1955): *A generalized inverse for matrices*. In: *Mathematical proceedings of the Cambridge philosophical society*, 51, Cambridge University Press, pp. 406–413, doi:10.1017/S0305004100030401.

[21] R. Puystjens & M.C. Gouveia (2004): *Drazin invertibility for matrices over an arbitrary ring*. *Linear algebra and its applications* 385, pp. 105–116, doi:10.1016/S0024-3795(03)00618-9.

[22] R. Puystjens & D. w. Robinson (1981): *The Moore-Penrose inverse of a morphism with factorization*. *Linear Algebra and its Applications* 40, pp. 129–141, doi:10.1016/0024-3795(81)90145-2.

[23] R. Puystjens & D. W. Robinson (1984): *The Moore-Penrose inverse of a morphism in an additive category*. *Communications in algebra* 12(3), pp. 287–299, doi:10.1080/00927878408823004.

[24] R. Puystjens & D. W. Robinson (1985): *EP morphisms*. *Linear algebra and its applications* 64, pp. 157–174, doi:10.1016/0024-3795(85)90273-3.

[25] R. Puystjens & D. W. Robinson (1987): *Generalized inverses of morphisms with kernels*. *Linear Algebra and Its Applications* 96, pp. 65–86, doi:10.1016/0024-3795(87)90336-3.

[26] R. Puystjens & D. W. Robinson (1990): *Symmetric morphisms and the existence of Moore-Penrose inverses*. *Linear Algebra and Its Applications* 131, pp. 51–69, doi:10.1016/0024-3795(90)90374-L.

[27] P. Selinger (2007): *Dagger compact closed categories and completely positive maps*. *Electronic Notes in Theoretical computer science* 170, pp. 139–163, doi:10.1016/j.entcs.2006.12.018.

[28] Z. Wang (2017): *A class of Drazin inverses in rings*. *Filomat* 31(6), pp. 1781–1789, doi:10.2298/FIL1706781W.

[29] Y. Wei & S. Qiao (2003): *The representation and approximation of the Drazin inverse of a linear operator in Hilbert space*. *Applied Mathematics and Computation* 138(1), pp. 77–89, doi:10.1016/S0096-3003(02)00100-5.

[30] Z. Wu, S. Zhang & C. Zhu (2014): *Remarks on generalized quantum gates*. *Hacettepe Journal of Mathematics and Statistics* 43(3), pp. 451–460. Available at https://dergipark.org.tr/tr/download/article-file/649195.