EPTCS 406

Proceedings of the

21st International Conference on Quantum Physics and Logic

Buenos Aires, Argentina, July 15-19, 2024

Edited by: Alejandro Díaz-Caro and Vladimir Zamdzhiev

Published: 12th August 2024
DOI: 10.4204/EPTCS.406
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Preface

Alejandro Díaz-Caro
Universidad Nacional de Quilmes. DCyT. Bernal, Buenos Aires, Argentina
Universidad de Buenos Aires. FCEyN. DC. Buenos Aires, Argentina
CONICET-Universidad de Buenos Aires. ICC. Buenos Aires, Argentina
alejandro@diaz-caro.info

Vladimir Zamdzhiev
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France
vladimir.zamdzhiev@inria.fr

This volume contains the proceedings of the 21st International Conference on Quantum Physics and Logic (QPL 2024). The conference was held from 15 to 19 July 2024, at Instituto de Ciencias de la Computación in Buenos Aires, Argentina, co-organized by Universidad Nacional de Quilmes and Universidad de Buenos Aires.

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 2024 conference solicited four kinds of submissions: proceedings submissions, non-proceedings submissions, poster submissions, and programming tool 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 during a talk at the conference, and these papers were included in the proceedings of QPL 2024. No other types of submissions were considered for inclusion in the proceedings. Non-proceedings submissions consisted of a three-page summary, together with a link to a separate published paper or preprint. Authors of accepted non-proceedings submissions were invited to present their work in the form of a talk during the conference. Poster submissions consisted of a three-page abstract of (partial) results or work in progress, and authors of accepted poster submissions were invited to present their work during the poster session. Programming tool submissions consisted of three-page descriptions of programming tools or frameworks. Authors of accepted programming tool submissions were given the opportunity to present their software during a dedicated "Tool Session".

These proceedings contain 9 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. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. ii–iii, doi:10.4204/EPTCS.406.0

A total of 135 submissions (excluding withdrawals and retractions) were considered for review by the PC. QPL 2024 had 59 accepted submissions in the non-proceedings track and 9 accepted submissions in the proceedings track. Most of the talks were presented during parallel sessions, but a selection of talks was presented during plenary sessions in the mornings. The program also included a poster session with 30 accepted posters, and one session dedicated to showcasing the accepted programming tool submission. There was also an industry session where industrial sponsors of QPL 2024 were given the opportunity to present their companies. The industry session consisted of three talks—one by Quantinuum, one by the Technology Innovation Institute, and one by Quandela, all of them platinum sponsors.

The QPL 2024 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 2024 to Nicolas Heurtel (Quandela, Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles) for his paper “A complete graphical language for linear optical circuits with finite-photon-number sources and detectors”.

The official website of the conference is `https://qpl2024.dc.uba.ar`, and it contains a lot of relevant information about QPL 2024.

The Program Committee consisted of 50 members: Barbara Amaral, Pablo Arrighi, Miriam Backens, Rui Soares Barbosa, Alessandro Bisio, Titouan Carette, Ulysse Chabaud, Giulio Chiribella, Bob Coecke, Alejandro Díaz-Caro (co-chair), Ross Duncan, Pierre-Emmanuel Emeriau, Stefano Gogioso, Amar Hadzihasanovic, Chris Heunen, Matty Hoban, Federico Holik, Dominic Horsman, Emmanuel Jeandel, Martti Karvonen, Kohei Kishida, Aleks Kissinger, Ravi Kunjwal, Martha Lewis, Shane Mansfield, Simon Martiel, Mio Murao, Ognyan Oreshkov, Anna Pearson, Simon Perdrix, Robert Rand, Neil Ross, Mehrnoosh Sadrzadeh, Ana Belén Sainz, Carlo Maria Scandolo, John Selby, Peter Selinger, Sonja Smets, Pawel Sobocinski, Isar Stubbe, Benoît Valiron, John van de Wetering, Augustin Vanrietvelde, V. Vilasini, Renaud Vilmart, Juliana Kaizer Vizzotto, Quanlong Wang, Alexander Wilce, Vladimir Zamdzhiev (co-chair), and Margherita Zorzi.

The Organising Committee consisted of four members: Guido Bellomo, Alejandro Díaz-Caro (chair), Santiago Figueira, and Federico Holik.

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 2024. We also thank all external subreviewers for their help and the authors for their submissions to QPL 2024. Thanks are also due to Ana Belén Sainz for making a few decisions on papers for which the PC co-chairs were not available. We are grateful to the EPTCS team for their help in preparing the proceedings of the conference. We also thank the members of the Organising Committee for their help in setting up the conference and the collaborators Martín Bosyk and Octavio Malherbe. We thank the student helpers who volunteered to assist us: Carlos Miguel Soto, Malena Ivnisky, Marcos Lammers, Nicolás Ciancaglini, Nicolás Alberto Monzón, Rafael Romero, and Santiago Cifuentes. Finally, we thank the QPL Steering Committee for their support and all the people who have contributed to the success of QPL 2024.

QPL 2024 received financial support from Quantinuum, the Technology Innovation Institute, and Quandela, as well as grants RD315 from CONICET and UBACyT-RC-12-2024 from Universidad de Buenos Aires.

July 2024,
Alejandro Díaz-Caro and Vladimir Zamdzhiev

A Sound and Complete Equational Theory for 3-Qubit Toffoli-Hadamard Circuits

Matthew Amy
Simon Fraser University
Burnaby, Canada
matt_amy@sfu.ca

Neil J. Ross
Dalhousie University
Halifax, Canada
neil.jr.ross@dal.ca

Scott Wesley
Dalhousie University
Halifax, Canada
scott.wesley@dal.ca

We give a sound and complete equational theory for 3-qubit quantum circuits over the Toffoli-Hadamard gate set $\{X, CX, CCX, H\}$. That is, we introduce a collection of true equations among Toffoli-Hadamard circuits on three qubits that is sufficient to derive any other true equation between such circuits. To obtain this equational theory, we first consider circuits over the Toffoli-K gate set $\{X, CX, CCX, K\}$, where $K = H \otimes H$. The Toffoli-Hadamard and Toffoli-K gate sets appear similar, but they are crucially different on exactly three qubits. Indeed, in this case, the former generates an infinite group of operators, while the latter generates the finite group of automorphisms of the well-known E_8 lattice. We take advantage of this fact, and of the theory of automorphism groups of lattices, to obtain a sound and complete collection of equations for Toffoli-K circuits. We then extend this equational theory to one for Toffoli-Hadamard circuits by leveraging prior work of Li *et al.* on Toffoli-Hadamard operators.

1 Introduction

The *Toffoli-Hadamard* gate set is obtained by extending the classical reversible gate set $\{X, CX, CCX\}$ with the Hadamard gate H. The addition of the Hadamard gate promotes the gate set $\{X, CX, CCX\}$ from one that is universal for classical reversible computation to one that is universal for quantum computation [1, 23]. Because the Hadamard gate can introduce phases of (-1) and produce superpositions, one can think of the addition of the Hadamard gate as a simple way to augment classical reversible computation with these typically quantum features. In turn, this motivates the study of Toffoli-Hadamard circuits [1, 4, 3, 10, 18, 23, 25, 26].

In recent years, an important effort has been made to understand quantum circuits equationally. If G is a set of quantum gates, an *equational theory* for G is given by a set of *equations* (or *relations*) among the circuits over G. The equational theory is *sound* if it only equates circuits that correspond to the same operator, and *complete* if it always equates circuits that correspond to the same operator. Equational theories can be used to optimize and verify quantum circuits in practice, but, more fundamentally, they can illuminate the mathematical structure underlying the gate set G. Sound and complete equational theories have been found for several important gate sets [2, 7, 8, 11, 19, 22].

In this paper, we give a sound and complete equational theory for 3-qubit Toffoli-Hadamard circuits. A presentation for the group of Toffoli-Hadamard operators was given in [18], but the presentation uses 1-, 2-, and 4-level operators as generators. While these operators can be represented by Toffoli-Hadamard circuits, this leads to an unnatural presentation, from the perspective of quantum circuits. What is more, the presentation of [18] contains over 2000 relations, even when restricted to 3-qubit operators. Many of these relations can be presented concisely as relation schemas in the language of operators, but these relations need to be expanded to be stated in the language of circuits. In contrast, our presentation contains only 65 relations, most of which are natural from the perspective of quantum circuits.

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 1–43, doi:10.4204/EPTCS.406.1

To obtain our presentation, we first consider circuits over the Toffoli-K gate set $\{X, CX, CCX, K\}$, where $K = H \otimes H$. The Toffoli-Hadamard and Toffoli-K gate sets appear similar, but they are crucially different on exactly three qubits. Indeed, in this case, the former generates an infinite group of operators, while the latter generates the finite group of automorphisms of the well-known E_8 lattice. The correspondence between 3-qubit Toffoli-K circuits and the automorphisms of the E_8 lattice was previously known (see [14, 21]). We take advantage of this correspondence, and of the theory of automorphism groups of lattices, to obtain a sound and complete collection of equations for Toffoli-K circuits. The automorphism group of the E_8 lattice admits a finite Coxeter presentation, which enjoys many geometric and combinatorial properties, and we use *Tietze transformations* to turn the Coxeter presentation of the group of Toffoli-K operators into a concise circuit presentation. We then extend this equational theory to one for Toffoli-Hadamard circuits by building upon [18]. Our paper therefore regards the group of 3-qubit Toffoli-Hadamard circuits as an extension of the automorphism group of the E_8 lattice in order to elucidate its underlying mathematical structure.

The paper is organized as follows. In Section 2, we define three groups of interest. In Section 3, we recall prior results on finite group presentations and we review Tietze transformations. In Section 4, we use the theory of Coxeter groups to obtain a presentation for the group of 3-qubit Toffoli-K circuits using a minimal number of generators. We moreover show that every operator in this group can be represented by a circuit of Toffoli-count at most 120. In Sections 5 and 6, the results of Li *et al.* [18] are used to extend this presentation to a presentation for 3-qubit circuits over the gate set $\{X, CX, CCX, K, CCZ\}$, and then to one for 3-qubit Toffoli-Hadamard circuits. Our approach relies on a large number of derivations and intricate rewriting proofs, which we relegate to several appendices and a supplement [5].

2 Three Groups and Their Generators

Let $\mathbb{Z}$ denote the ring of integers. The *half-integers* $\mathbb{Z}+1/2$ are defined as $\mathbb{Z}+1/2 = \{a+1/2 \mid a \in \mathbb{Z}\}$ and the ring of *dyadic fractions* $\mathbb{D}$ is defined as $\mathbb{D} = \mathbb{Z}[1/2] = \{a/2^k \mid a \in \mathbb{Z} \text{ and } k \in \mathbb{N}\}$. Equivalently, $\mathbb{D}$ is the smallest subring of $\mathbb{Q}$ that contains both $\mathbb{Z}$ and $1/2$. The E_8 *lattice* Γ_8 is the following collection of 8-dimensional vectors,

$$\Gamma_8 = \left\{x \in \mathbb{Z}^8 \cup (\mathbb{Z}+1/2)^8 \;\middle|\; \sum x_i \equiv 0 \pmod{2}\right\}.$$

In other words, Γ_8 consists of the vectors in $\mathbb{R}^8$ whose components sum to an even integer and are either all integers or all half-integers. The E_8 lattice is well-studied because it enjoys many remarkable properties [12]; in particular, it provides the densest sphere packing in dimension 8 [24].

We now introduce the three groups that will be the focus of this paper. Let R be a ring. For each $n \in \mathbb{N}$, let $\mathrm{GL}(n,R)$ denote the general linear group over R in dimension n and let $\mathrm{O}(n,R)$ denote the orthogonal group over R in dimension n. Define $W(E_8)$ to be the subgroup of $\mathrm{O}(8,\mathbb{D})$ consisting of the elements of $\mathrm{O}(8,\mathbb{D})$ that fix the E_8 lattice. Define $\mathsf{TofH}(n)$ to be the subgroup of $\mathrm{O}(2^n, \mathbb{Z}[1/\sqrt{2}])$ consisting of matrices $M/\sqrt{2}^k$, where M is an integer matrix and $k \in \mathbb{N}$. We will be interested in the groups $W(E_8)$, $\mathrm{O}(8,\mathbb{D})$, and $\mathsf{TofH}(3)$. Note that we have $W(E_8) \leq \mathrm{O}(8,\mathbb{D}) \leq \mathsf{TofH}(3)$.

The above three groups are generated by well-known quantum gates. Let I denote the 2×2 identity matrix and $\otimes$ denote the Kronecker tensor product. Given a dimension 2 matrix M, define $M_0 = M \otimes I \otimes I$, $M_1 = I \otimes M \otimes I$, and $M_2 = I \otimes I \otimes M$. That is, M_j applies operator M to the j-th qubit. Furthermore, define $CM_{j,k}$ to be the operator that sends each standard basis state $|x_0x_1x_2\rangle$ to $(M_k)^{x_j}|x_0x_1x_2\rangle$. That is, $CM_{j,k}$ applies operator M to the k-th qubit whenever the j-th qubit is in the basis state $|1\rangle$. Likewise, define $CCM_{j,k}$ to be the operator that sends each standard basis state $|x_0x_1x_2\rangle$ to $(M_l)^{x_jx_k}|x_0x_1x_2\rangle$ for

$l \in \{1,2,3\} \setminus \{j,k\}$. That is, $CCM_{j,k}$ applies operator M to the l-th qubit whenever the j-th and k-th qubits are both in the basis state $|1\rangle$. The operators $CM_{j,k}$ and $CCM_{j,k}$ denote the usual *controlled-M gate* and *doubly-controlled-M gate*, respectively. Now recall the *Pauli X*, *Pauli Z*, *Hadamard*, and *K* matrices,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \text{and} \qquad K_{j,k} = H_j \circ H_k,$$

where $(\circ)$ denotes matrix multiplication. Then $CCX_{j,k}$ denotes the *Toffoli* gate. Note that the matrices of the form X_j, $K_{j,k}$, $CX_{j,k}$, $CCX_{j,k}$, and $CCZ_{j,k}$ belong to $\mathsf{O}(8,\mathbb{D})$. It is known that $\{X_j, CX_{j,k}, CCX_{j,k}, K_{j,k}\}$ is a generating set for $W(E_8)$ [12]. Similarly it is known that $\{X_j, CX_{j,k}, CCX_{j,k}, K_{j,k}, CCZ_{j,k}\}$ and $\{X_j, CX_{j,k}, CCX_{j,k}, H_j\}$ are generating sets for $\mathsf{O}(8,\mathbb{D})$ and $\mathsf{TofH}(3)$, respectively [3][1].

3 Presentations and Tietze Transformations

We now recall key results from combinatorial group theory. In particular, we discuss monoid presentations, as well as *Tietze transformations*, which will play an important role in the rest of the paper.

3.1 Presentations

Let Σ be an alphabet (i.e., a set of symbols). Then Σ^* is the *free monoid on* Σ. The elements of Σ^* are the words over Σ, the monoid operation is string concatenation, which we denote by $(\cdot)$, and the identity element in Σ^* is the empty word, which we denote by ε.

If G is a monoid and Q is a quotient of G, then we write $\pi_Q : G \twoheadrightarrow Q$ to denote the canonical projection of G onto Q. Given a subset R of $\Sigma^* \times \Sigma^*$, we write $Q = \langle \Sigma \mid R \rangle$ to denote the largest quotient of $G = \Sigma^*$ such that $\pi_Q(q) = \pi_Q(r)$ for all $(q,r) \in R$. If $M \cong \langle \Sigma \mid R \rangle$, then we say that $\langle \Sigma \mid R \rangle$ is a *presentation of* M and write $q \approx_R$ for each $(q,r) \in R$. The elements of Σ are called *generators* and the elements of R are called *relations*. If, for each $x \in \Sigma$, there exists a $w \in \Sigma^*$ such that $\pi_Q(x \cdot w) = \pi_Q(\varepsilon)$, then M is a group and $\langle \Sigma \mid R \rangle$ is a *monoid presentation for the group M*. In either case, if Σ and R are finite, then $\langle \Sigma \mid R \rangle$ is a *finite presentation*. We distinguish between the presentations $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid R' \rangle$ whenever $R \neq R'$, even if R and R' generate the same quotient.

Certain aspects of presentations can be conveniently expressed in the language of string rewriting. Let Σ be an alphabet and $R \subseteq \Sigma^* \times \Sigma^*$. Fix some $u \in \Sigma^*$ and $v \in \Sigma^*$. If there exists some $(q,r) \in R$ and $s,t \in \Sigma^*$ such that $u = s \cdot q \cdot t$ and $v = s \cdot r \cdot t$, then we write,

$$u \overset{R}{\rightarrow} v.$$

If either $u \overset{R}{\rightarrow} v$ or $u \overset{R}{\leftarrow} v$, then we write $u \overset{R}{\leftrightarrow} v$. We say that u *rewrites to* v, denoted $u \sim_R v$, if either $u = v$ or there exists a finite sequence,

$$u \overset{R}{\leftrightarrow} w_1 \overset{R}{\leftrightarrow} w_2 \overset{R}{\leftrightarrow} \cdots \overset{R}{\leftrightarrow} w_n \overset{R}{\leftrightarrow} v.$$

That is, $(\sim_R)$ is the symmetric, transitive and reflexive closure of $\overset{R}{\rightarrow}$. Importantly, $\pi_Q(u) = \pi_Q(v)$ in $Q = \langle \Sigma, R \rangle$ if and only if $u \sim_R v$ [9, Ch. 7]. That is, two words u and v represent the same element in G if and only if the relations in R suffice to rewrite u into v. In this sense, the relations in R define a *complete equational theory* for the monoid Q with respect to the generators Σ. For further information on presentations and on rewriting, the reader is encouraged to consult [17] and [9], respectively.

[1] The generator $CCZ_{j,k}$ is necessary to apply these results to the ancilla-free three-qubit case.

3.2 Tietze Transformations

The transformations, which we state formally below, allow one to add a generator, remove a generator, add a relation, and remove a relation. Let Σ be an alphabet, $R \subseteq \Sigma^* \times \Sigma^*$, and $G = \langle \Sigma \mid R \rangle$ be a monoid.

- **Gen(+)**. Let x be a symbol. If $x \notin \Sigma$ and $w \in \Sigma^*$, then $G \cong \langle \Sigma \cup \{x\} \mid R \cup \{x \approx w\} \rangle$.
- **Gen(−)**. Let $x \in \Sigma$, $x \approx_R w$, $\Pi = \Sigma \setminus \{x\}$, and $Q = R \setminus \{x \approx w\}$. If $Q \subseteq \Pi^* \times \Pi^*$, then $G \cong \langle \Pi \mid Q \rangle$.
- **Rel(+)**. If $q \sim_R r$, then $G \cong \langle \Sigma \mid R \cup \{q \approx r\} \rangle$.
- **Rel(−)**. Let $q \approx_R r$ and $Q = R \setminus \{q \approx r\}$. If $q \sim_Q r$, then $G \cong \langle \Sigma \mid Q \rangle$.

The **Gen(+)** rule states that one can add a generator if one also adds a relation defining it in terms of the other generators. The **Gen(−)** rule states that a generator can be removed if it is defined in terms of the other generators, and does not appear in any of the other relations. The **Rel(+)** rule states that if a relation can be derived from the existing ones, then it can be added to the set of relations. Finally, the **Rel(−)** rule conversely states that if a relation can be derived from other relations in the presentation, it is redundant and can be removed.

Tietze transformations are sound and complete for the isomorphism of finite monoid presentations. That is, two presentations $\langle \Sigma \mid R \rangle$ and $\langle \Pi \mid Q \rangle$ specify the same monoid if and only if $\langle \Sigma \mid R \rangle$ can be obtained from $\langle \Pi \mid Q \rangle$ through a finite sequence of Tietze transformations [15, Section 1].

The goal of this paper is to find presentations for groups of quantum operators in which each generator corresponds to a specific quantum gate. More explicitly, given a group G, a generating set Σ, and a *semantic interpretation* $[\![\cdot]\!]_\Sigma : \Sigma \to G$, our goal is to find a set of relations $R \subseteq \Sigma^* \times \Sigma^*$ such that $[\![\cdot]\!]$ induces an isomorphism between $\langle \Sigma \mid R \rangle$ and G. In what follows, we start from a known presentation $\langle \Pi \mid Q \rangle$ over different generators Π with a semantic interpretation $[\![\cdot]\!]_\Pi : \Pi \to G$, and obtain $\langle \Sigma \mid R \rangle$ via a sequence of Tietze transformations. As these Tietze transformations act on the abstract group $\langle \Pi \mid Q \rangle$, one must ensure that the transformations respect the intended interpretation $[\![\cdot]\!]_\Sigma$ of the new generators in Σ, as discussed further in Appendix A.

4 From Coxeter to Circuit Presentations of $W(E_8)$

A *Coxeter group* is a group G which admits a group presentation of the form $\langle r_1, \ldots, r_n \mid (r_j r_k)^{N_{j,k}} \approx \varepsilon \rangle$, where N is an $n \times n$ matrix over $\mathbb{N} \cup \{\infty\}$ such that $N_{j,j} = 1$ and $N_{j,k} > 1$ for all $j \neq k$ [16]. The matrix N is known as the *Coxeter matrix* of G. Note that since $\pi_G(r_j \cdot r_j) = \varepsilon$ for each r_j, then every Coxeter presentation is automatically a monoid presentation for a group. Coxeter groups are an abstraction of reflection groups and, in particular, for every finite Coxeter group G, there is a faithful group representation $G \to \mathrm{O}(n)$ that maps each r_j to a reflection in $\mathbb{R}^n$ [16]. Recall that a *Householder transformation* is a reflection about the hyperplane normal to some vector $\alpha \in \mathbb{R}^n$ defined by $v \mapsto v - 2\frac{\langle v,\alpha \rangle}{\langle \alpha,\alpha \rangle}\alpha$ [16]. If r is the reflection about the hyperplane normal to $\alpha \in \mathbb{R}^n$ and $M \in \mathrm{O}(n)$, then $M \circ r \circ M^{-1}$ is the reflection about the hyperplane normal to $M\alpha$ [16, Prop. 1.2]. As a special case, v and $-v$ define the same reflection.

The goal of this section is to construct a presentation for the Weyl group of the E_8 lattice in terms of Toffoli-K gates. Recall that the Weyl group for any lattice $L \subseteq \mathbb{R}^n$ is the finite reflection group generated by reflections about the roots of L (see [16, Sec. 2.9]). That is, given a root system Φ for L, the group $W(L)$ is generated by $\{r_\alpha : \alpha \in \Phi\}$ where r_α is the reflection through the hyperplane normal to α. Consequently, $W(E_8)$ is a Coxeter group. A root system and the corresponding Coxeter matrix for $W(E_8)$ are given in Figure 1.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1/2 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1/2 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1/2 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/2 \end{bmatrix}$$

(a) E_8 Root System.

$$\begin{bmatrix} 1 & 3 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 1 & 3 & 2 & 2 & 2 & 2 & 2 \\ 2 & 3 & 1 & 3 & 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & 1 & 3 & 2 & 2 & 2 \\ 2 & 2 & 2 & 3 & 1 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 3 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 3 & 2 & 1 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 & 3 & 1 \end{bmatrix}$$

(b) $W(E_8)$ Coxeter Matrix.

Figure 1: The root system and Coxeter matrix for $W(E_8)$. Note that the root system consists of 8 vectors and are presented as the columns of an 8×8 matrix.

$$\begin{tikzcd}
\langle \Sigma_{E8} \mid R_{E8} \rangle \arrow[r, "\textbf{Gen}(+)"] \arrow[dd, dotted, "\cong"] & \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{D(E8)} \rangle \arrow[r, "\textbf{Rel}(+)"] & \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{D(E8)} \cup R_{E8(D)} \rangle \arrow[d, "\textbf{Rel}(+)"] \\
 & & \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{D(E8)} \cup R_{E8(D)} \cup R_D \rangle \arrow[d, "\textbf{Rel}(-)"] \\
\langle \Sigma_D \mid R_D \rangle & \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8(D)} \cup R_D \rangle \arrow[l, "\textbf{Gen}(-)"'] & \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{E8(D)} \cup R_D \rangle \arrow[l, "\textbf{Rel}(-)"']
\end{tikzcd}$$

Figure 2: A diagrammatic summary of the Tietze transformations used to obtain a presentation for $W(E_8)$. Note that in this diagram Σ_{E8} denotes the Coxeter generators, Σ_D denotes the dyadic Toffoli-Hadamard gates, $R_{D(E8)}$ expresses the Σ_D in terms of Σ_{E8}, and $R_{E8(D)}$ expresses Σ_{E8} in terms of Σ_D.

To obtain a presentation in terms of Toffoli-K gates, we begin with the Coxeter presentation of $W(E_8)$. The desired presentation is then obtained through a sequence of Tietze transformations. First, the **Gen**$(+)$ rule is used to introduce the dyadic Toffoli-K gates with their intended semantics. Second, the **Rel**$(+)$ rule is used to rewrite the Coxeter generators in terms of Toffoli-K gates (call these relations $R_{E8(D)}$). Third, the **Rel**$(+)$ rule is used to introduce well-known relations satisfied by the Toffoli-K gates (see, e.g., [18, 19]). Given these new relations, the **Rel**$(-)$ rule is used to eliminate all defining relations for the Toffoli-K gates. In a similar fashion, the **Rel**$(-)$ rule is also used to eliminate the Coxeter relations of $W(E_8)$. At this point, the Coxeter generators only appear in $R_{E8(D)}$, and can be eliminated using the **Gen**$(-)$ rule. What remains is a presentation of $W(E_8)$ in terms of Toffoli-K gates. All steps of this proof are summarized in Figure 2.

Each step of this proof requires numerous applications of the corresponding Tietze transformation. To establish that each **Gen**$(+)$ and **Rel**$(+)$ transformation holds, an equation of 8×8 matrices must be validated. To establish that each **Rel**$(-)$ transformation holds, a derivational proof must be validated. In both cases, the proof obligation is computational in nature. The validity of our Tietze transformations have been machine-verified by the software package TIETZE[2].

[2] Available at: `https://github.com/meamy/tietze`.

4.1 Introducing the Toffoli-K Gates

The generators of $W(E_8)$ can be written as follows.

$$\begin{aligned}
r_1 &= X_0 \circ X_1 \circ CCX_{0,1} \circ X_1 \circ X_0 & r_2 &= X_0 \circ CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1} \circ X_0 \\
r_3 &= X_0 \circ CCX_{0,1} \circ X_0 & r_4 &= CX_{0,1} \circ CX_{0,2} \circ CCX_{1,2} \circ CX_{0,2} \circ CX_{0,1} \\
r_5 &= X_1 \circ CCX_{0,1} \circ X_1 & r_6 &= CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1} \\
r_7 &= CZ_{0,1} \circ CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1} \circ CZ_{0,1} & r_8 &= K_{1,2} \circ X_1 \circ X_2 \circ CZ_{0,2} \circ CCX_{1,2} \circ CZ_{0,2} \circ X_2 \circ X_1 \circ K_{1,2}
\end{aligned}$$

These equations can be derived from the geometry of $\mathbb{R}^8$. First, note that $CCX_{0,1}$ is a reflection about the hyperplane normal to $|\widehat{b}\rangle = |1\rangle \otimes |1\rangle \otimes |-\rangle$ where $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$. Then for each generator r_j with normal vector $|b_j\rangle$, it suffices to find an element $M \in W(E_8)$ such that $M\,|\widehat{b}\rangle = |b_j\rangle$. The corresponding circuit would be $M \circ CCX_{1,2} \circ M^{-1}$. As an example of this technique, consider the Coxeter generator r_3 defined by the normal vector $|b_3\rangle = |0\rangle \otimes |1\rangle \otimes |-\rangle$. Since $(X_0)\,|\widehat{b}\rangle = |b_3\rangle$ with X_0 self-inverse, then $r_3 = X_0 \circ CCX_{0,1} \circ X_0$. The remaining cases are established in Appendix C.

Next, the Toffoli-K gates are introduced. For simplicity of presentation, we first introduce the swap matrices $\sigma_{j,k} = CX_{j,k} \circ CX_{k,j} \circ CX_{j,k}$ where $\sigma_{j,k}$ permutes the j-th qubit with the k-th qubit. Recall that the Toffoli-K gates correspond to the following matrices[3]:

$$\Sigma_D := \left\{ X_j, CX_{k,l}, CCX_{j,k}, Z_j, CZ_{j,k}, K_{j,j+1}, \sigma_{j,k} \mid j,k,l \in \{0,1,2\}, j < k, j \neq l \neq k \right\}.$$

It turns out that all Toffoli-K gates are generated by X_0, $CX_{1,0}$, $CCX_{1,2}$, and $K_{1,2}$. To derive these primitive gates, it helps to first derive several diagonal matrices over (± 1). These are then used to derive the $CCX_{0,1}$ and X_0 gates. From this, the swap matrices can be derived, after which, it is relatively straightforward to construct the X_0, $K_{1,2}$, and $CX_{0,1}$ gates. This yields four words w_X, w_K, w_{CX}, and w_{CCX}, such that $[\![w_X]\!]^*_{E8} = X_0$, $[\![w_{CX}]\!]^*_{E8} = CX_{1,0}$, $[\![w_{CCX}]\!]^*_{E8} = CCX_{1,2}$, and $[\![w_K]\!]^*_{E8} = K_{1,2}$, as outlined in Appendix D.

The remaining Toffoli-K gates are derived in terms of $K_{1,2}$, $CCX_{1,2}$, X_0, and $CX_{0,1}$. To simplify this process, we note that once a gate has been derived, it may then be used to derive other gates. This is analogous to how the generator X_0 appears in the defining relation for $CX_{0,1}$. Given a set of defining relations, if the dependencies between the generators defined by the relations form an acyclic digraph, then the defining relations arise from a valid sequence of **Gen**$(+)$ transformations (see Appendix B.2). Likewise, the derived generators can be eliminated by a valid sequence of Tietze transformation. The defining relations for the remaining 19 gates are found in Appendix B.2. Since the dependencies among these generators are acyclic, then they must arise from 19 valid applications of the **Gen**$(+)$ rules. Let $R_{D(E8)}$ denote all 23 relations. Then $W(E_8) \cong \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{D(E8)} \rangle$.

4.2 Deriving the $W(E_8)$ Coxeter Generators

Recall the circuit definitions for the $W(E_8)$ generators from Section 4.1. Let $R_{E8(D)}$ denote the set of corresponding relations. For example, the relation corresponding to r_3 is $r_3 \approx X_0 \cdot CCX_{0,1} \cdot X_0$. Since these relations hold by definition, then they may be introduced via 8 applications of **Rel**$(+)$ and consequently $W(E_8) \cong \langle \Sigma_{E8} \cup \Sigma_D \mid R_{E8} \cup R_{D(E8)} \cup R_{E8(D)} \rangle$.

4.3 Elimination of the Coxeter Generators

In this section, the relations in R_{E8} and $R_{D(E8)}$ are eliminated. To do this, some additional (Σ_D)-relations are required. For the remainder of this section, let $M_{(x_0,x_1,\ldots,x_k)}$ denote a gate M applied to the qubits x_0

[3] For simplicity, we assume that all K gates are applied to adjacent qubits. This is sufficient, since $K_{0,2} = K_{0,1} \circ K_{1,2}$.

$$
\begin{aligned}
\mathbf{r_1} \cdot r_1 &\rightarrow X_0 \cdot X_1 \cdot CCX_{0,1} \cdot X_1 \cdot X_0 \cdot \mathbf{r_1} \\
&\rightarrow X_0 \cdot X_1 \cdot CCX_{0,1} \cdot X_1 \cdot \mathbf{X_0} \cdot \mathbf{X_0} \cdot X_1 \cdot CCX_{0,1} \cdot X_1 \cdot X_0 \\
&\rightarrow X_0 \cdot X_1 \cdot CCX_{0,1} \cdot \mathbf{X_1} \cdot \mathbf{X_1} \cdot CCX_{0,1} \cdot X_1 \cdot X_0 \\
&\rightarrow X_0 \cdot X_1 \cdot \mathbf{CCX_{0,1}} \cdot \mathbf{CCX_{0,1}} \cdot X_1 \cdot X_0 \\
&\rightarrow X_0 \cdot \mathbf{X_1} \cdot \mathbf{X_1} \cdot X_0 \\
&\rightarrow \mathbf{X_0} \cdot \mathbf{X_0} \\
&\rightarrow \varepsilon
\end{aligned}
$$

(a) Deriving ε from $r_1 \cdot r_1$.

$$
\begin{aligned}
\mathbf{CX_{0,1}} \cdot X_1 &\rightarrow \sigma_{0,1} \cdot CX_{1,0} \cdot \sigma_{0,1} \cdot \mathbf{X_1} \\
&\rightarrow \sigma_{0,1} \cdot CX_{1,0} \cdot \boldsymbol{\sigma_{0,1}} \cdot \boldsymbol{\sigma_{0,1}} \cdot X_0 \sigma_{0,1} \\
&\rightarrow \sigma_{0,1} \cdot \mathbf{CX_{1,0}} \cdot \mathbf{X_0} \cdot \sigma_{0,1} \\
&\rightarrow \sigma_{0,1} \cdot X_0 \cdot CX_{1,0} \sigma_{0,1} \\
&\rightarrow \boldsymbol{\sigma_{0,1}} \cdot \mathbf{X_0} \cdot \boldsymbol{\sigma_{0,1}} \cdot \sigma_{0,1} \cdot CX_{1,0} \cdot \sigma_{0,1} \\
&\rightarrow X_1 \cdot \boldsymbol{\sigma_{0,1}} \cdot \mathbf{CX_{1,0}} \cdot \boldsymbol{\sigma_{0,1}} \\
&\rightarrow X_1 \cdot CX_{0,1}
\end{aligned}
$$

(b) Deriving $X_1 \cdot CX_{0,1}$ from $CX_{0,1} \cdot X_1$.

Figure 3: Examples of derivations proofs which appear in the proof that $W(E_8) \cong \langle \Sigma_D \mid R_0 \rangle$.

through to x_k. For example, if M is a doubly-controlled X gate, then $M_{(1,2,0)}$ would correspond to $CCX_{1,2}$. Using this notation, we introduce the following families of relations, denoted R_D.

- **Bifunctoriality**. $M_S \cdot N_T = N_T \cdot M_S$, for all $M_S, N_T \in \Sigma_D$ with $S \cap T = \varnothing$.
- **Symmetry**. $\sigma_{i,j} \cdot M_S \cdot \sigma_{i,j} = M_{\sigma_{i,j}(S)}$, for all $M_S \in \Sigma_D$ and integers $0 \leq i < j \leq 3$.
- **Order**. $M \cdot M = \varepsilon$, for all $M \in \Sigma_D$.
- **Commutators**. $M_S \cdot N_T = N_T \cdot w$, for all $M_S, N_T \in \Sigma_D$ with $S \cap T \neq \varnothing$ and $w \in (\Sigma_D)^*$ minimal.

The relation $\sigma_{1,2} \approx CZ_{1,2} \cdot K_{1,2} \cdot CZ_{1,2} \cdot K_{1,2} \cdot CZ_{1,2} \cdot K_{1,2}$ from [7] is also included for simplicity. From these relations, all elements of $R_{D(E8)}$ and R_{E8} can be derived. Since all elements of Σ_{E8} are self-inverse, then it suffices to consider only the upper half of the Coxeter matrix for $W(E_8)$. As an example, consider the relation $r_1 \cdot r_1 \approx \varepsilon$ in R_{E8}. The proof proceeds as in Figure 3a. Then ε can be derived from $r_1 \cdot r_1$ using the relations in R_D. Similar methods can be used to eliminate the remaining (R_{E8})-relations. All derivations, for both $R_{D(E8)}$ and R_{E8} can be found in the supplement to this paper [5].

Theorem 4.1. $W(E_8) \cong \langle \Sigma_D \mid R_D \rangle$

It follows immediately from Theorem 4.1 that given any circuit C over Σ_D, there exists a minimal word w over the alphabet $\{r_1, r_2, \ldots, r_8\}$ such $[\![C]\!]_D^* = [\![w]\!]_{E8}^*$. Then by the decompositions of Section 4.1, there exists a circuit C' over Σ_D such that $[\![w]\!]_{E8}^* = [\![C']\!]_D^*$ such that C' contains exactly $|w|$ Toffoli gates. By [16, Thm. 1.8], every minimal word in $W(E_8)$ has length at most n, where n is the cardinality of the positive root system associated with $W(E_8)$. By [16], the positive root system associated with $W(E_8)$ has cardinality 120. Therefore, C' contains at most 120 Toffoli gates. This provides an upper-bound on the Toffoli count for circuits over Σ_D, which can be thought of as a measure of computational complexity for these three-qubit circuits.

Corollary 4.2. *If $C \in \Sigma_D^*$, then there exists $C' \in \Sigma_D$ with Toffoli count at most* 120 *such that* $[\![C]\!]_D^* = [\![C']\!]_D^*$.

4.4 A Reduced Set of Relations for $W(E_8)$

The relations R_D from Section 4.3 are far from minimal. For example, the family of commutator relations contains all relations of the form $CX_{j,k} \cdot X_k \approx X_k \cdot CX_{j,k}$. However, given all symmetry relations, it suffices to include only $CX_{1,0} \cdot X_0 \approx X_0 \cdot CX_{1,0}$. The remaining commutator relations can be derived, as illustrated in Figure 3b. Furthermore, many of the relations in R_D do not appear in any derivations of the supplement. For example, the relation $CX_{2,0} \cdot X_0 \approx X_0 \cdot CX_{2,0}$ does not appear, and therefore Theorem 4.1 holds with respect to the relation set $R_D \setminus \{CX_{2,0} \cdot X_0 \approx X_0 \cdot CX_{2,0}\}$. Using both techniques, a new relation set R_0 is obtained, as illustrated in Figure 4. All derivations can be found in the supplement to this paper [5].

Corollary 4.3. $W(E_8) \cong \langle \Sigma_D \mid R_0 \rangle$

$$CZ_{0,1} \approx K_{1,2} \cdot CX_{0,1} \cdot K_{1,2} \tag{1}$$
$$X_1 \approx CX_{0,1} \cdot X_0 \cdot CX_{0,1} \cdot X_0 \tag{2}$$
$$Z_0 \approx CZ_{0,1} \cdot CX_{0,1} \cdot CZ_{0,1} \cdot CX_{0,1} \tag{3}$$
$$Z_1 \approx K_{1,2} \cdot X_1 \cdot K_{1,2} \tag{4}$$
$$CX_{2,0} \approx X_1 \cdot CCX_{1,2} \cdot X_1 \cdot CCX_{1,2} \tag{5}$$
$$CX_{2,1} \approx CX_{2,0} \cdot CX_{0,1} \cdot CX_{2,0} \cdot CX_{0,1} \tag{6}$$
$$CX_{1,2} \approx K_{1,2} \cdot CX_{2,1} \cdot K_{1,2} \tag{7}$$
$$\sigma_{1,2} \approx CX_{1,2} \cdot CX_{2,1} \cdot CX_{1,2} \tag{8}$$
$$CX_{0,2} \approx \sigma_{1,2} \cdot CX_{0,1} \cdot \sigma_{1,2} \tag{9}$$
$$\sigma_{0,2} \approx CX_{0,2} \cdot CX_{2,0} \cdot CX_{0,2} \tag{10}$$
$$K_{0,1} \approx \sigma_{0,2} \cdot K_{1,2} \cdot \sigma_{0,2} \tag{11}$$
$$CX_{1,0} \approx K_{0,1} \cdot CX_{0,1} \cdot K_{0,1} \tag{12}$$
$$\sigma_{0,1} \approx CX_{0,1} \cdot CX_{1,0} \cdot CX_{0,1} \tag{13}$$
$$CCX_{0,2} \approx \sigma_{0,1} \cdot CCX_{1,2} \cdot \sigma_{0,1} \tag{14}$$
$$X_2 \approx \sigma_{0,2} \cdot X_0 \cdot \sigma_{0,2} \tag{15}$$
$$Z_2 \approx \sigma_{0,2} \cdot Z_0 \cdot \sigma_{0,2} \tag{16}$$
$$CCX_{0,1} \approx K_{1,2} \cdot CCX_{0,2} \cdot K_{1,2} \tag{17}$$
$$CZ_{0,2} \approx \sigma_{1,2} \cdot CZ_{0,1} \cdot \sigma_{1,2} \tag{18}$$
$$CZ_{1,2} \approx \sigma_{0,1} \cdot CZ_{0,2} \cdot \sigma_{0,1} \tag{19}$$
$$\sigma_{1,2} \approx CZ_{1,2} \cdot K_{1,2} \cdot CZ_{1,2} \cdot K_{1,2} \cdot CZ_{1,2} \cdot K_{1,2} \tag{20}$$
$$X_0 \cdot X_0 \approx \varepsilon \tag{21}$$
$$CX_{0,1} \cdot CX_{0,1} \approx \varepsilon \tag{22}$$
$$K_{1,2} \cdot K_{1,2} \approx \varepsilon \tag{23}$$
$$CCX_{1,2} \cdot CCX_{1,2} \approx \varepsilon \tag{24}$$
$$K_{0,1} \cdot K_{0,1} \approx \varepsilon \tag{25}$$
$$CX_{1,2} \cdot X_0 \approx X_0 \cdot CX_{1,2} \tag{26}$$
$$X_0 \cdot K_{1,2} \approx K_{1,2} \cdot X_0 \tag{27}$$
$$X_1 \approx \sigma_{0,1} \cdot X_0 \cdot \sigma_{0,1} \tag{28}$$
$$CX_{2,0} \approx \sigma_{0,2} \cdot CX_{0,2} \cdot \sigma_{0,2} \tag{29}$$
$$CX_{1,2} \approx \sigma_{0,1} \cdot CX_{0,2} \cdot \sigma_{0,1} \tag{30}$$
$$CX_{2,1} \approx \sigma_{0,1} \cdot CX_{2,0} \cdot \sigma_{0,1} \tag{31}$$
$$CCX_{0,1} \approx \sigma_{0,2} \cdot CCX_{1,2} \cdot \sigma_{0,2} \tag{32}$$
$$CCX_{0,1} \approx \sigma_{1,2} \cdot CCX_{0,2} \cdot \sigma_{0,2} \tag{33}$$
$$Z_1 \approx \sigma_{0,1} \cdot Z_0 \cdot \sigma_{0,1} \tag{34}$$
$$K_{0,1} \approx \sigma_{0,1} \cdot K_{0,1} \cdot \sigma_{0,1} \tag{35}$$
$$CCX_{1,2} \cdot CX_{1,0} \approx CX_{1,0} \cdot CCX_{1,2} \tag{36}$$
$$X_0 \cdot CCX_{1,2} \approx CCX_{1,2} \cdot X_0 \tag{37}$$
$$X_0 \cdot CX_{1,0} \approx CX_{1,0} \cdot X_0 \tag{38}$$
$$K_{0,1} \cdot K_{1,2} \approx K_{1,2} \cdot K_{0,1} \tag{39}$$
$$CZ_{0,1} \cdot CZ_{1,2} \approx CZ_{1,2} \cdot CZ_{0,1} \tag{40}$$
$$K_{0,1} \cdot Z_0 \approx X_0 \cdot K_{0,1} \tag{41}$$
$$X_0 \cdot CCX_{0,1} \approx CCX_{0,1} \cdot CX_{1,2} \cdot X_0 \tag{42}$$
$$CX_{0,1} \cdot CZ_{1,2} \approx CZ_{1,2} \cdot CZ_{0,2} \cdot CX_{0,1} \tag{43}$$
$$CX_{1,2} \cdot CCX_{1,2} \approx CCX_{1,2} \cdot CX_{1,0} \cdot CX_{1,2} \tag{44}$$
$$CCX_{1,2} \cdot CX_{0,1} \approx CX_{0,1} \cdot CCX_{0,2} \cdot CCX_{1,2} \cdot CCX_{0,2} \tag{45}$$
$$CCX_{0,1} \cdot CCX_{0,2} \approx CCX_{0,2} \cdot CCX_{0,1} \cdot CCX_{0,2} \cdot CCX_{0,1} \tag{46}$$

Figure 4: Relations for $W(E_8)$, denoted R_0.

4.5 A Minimal Generating Set for $W(E_8)$

Define $\Sigma_0 = \{X_0, CX_{0,1}, CCX_{1,2}, K_{1,2}\}$. From Section 4.1, it is clear that Σ_0 generates $W(E_8)$. In fact, Σ_0 is minimal in the sense that every proper subset of Σ_0 generates a proper subgroup of $W(E_8)$. In other words, no proper subset of Σ_0 generates $W(E_8)$. To show that Σ_0 is a minimal generating set for $W(E_8)$, it suffices to show that for every maximal proper subset Σ' of Σ_0, there exists some 8×8 dyadic matrix M such that M commutes with the elements of Σ' but does not commute with the elements of Σ_0. Intuitively, the subgroup generated by Σ' commutes with M, whereas the subgroup generated by Σ_0 does not commute with M. This claim is proven in Appendix E, and the matrices are constructed.

Theorem 4.4. *Σ_0 is a minimal generating set for $W(E_8)$.*

5 Extending to a Presentation of $\mathrm{O}(8,\mathbb{D})$

Li *et al.* [18] introduced a presentation for $\mathrm{O}(8,\mathbb{D})$ using m-level operators. Let $n > 0$, I be the $n \times n$ identity matrix, and $[m] = \{0, 1, \ldots, m-1\}$. Then given an $m \times m$ matrix M with $m < n$, and a strictly increasing sequence $(a_0, \ldots, a_{m-1})$ over $[m]$, define $M_{[a_0,\ldots,a_{m-1}]}$ to be the $n \times n$ matrix such that:

1. For each pair of elements (j,k) over $[m]$, the component (a_j, a_k) of $M_{[a_0,\ldots,a_{m-1}]}$ is equal to the component (a_j, a_k) of M;
2. For each pair of elements (j,k) over $[n] \setminus \{a_0, a_1, \ldots, a_m\}$, the component (j,k) of $M_{[a_0,a_1,\ldots,a_{m-1}]}$ is equal to the component (a_j, a_k) of I.

$$X_{[a,b]}{}^2 \approx \varepsilon \tag{47}$$
$$(-1)_{[a]}{}^2 \approx \varepsilon \tag{48}$$
$$K_{[a,b,c,d]}{}^2 \approx \varepsilon \tag{49}$$
$$X_{[a,b]} \cdot X_{[c,d]} \approx X_{[c,d]} \cdot X_{[a,b]} \tag{50}$$
$$X_{[a,b]} \cdot (-1)_{[c]} \approx (-1)_{[c]} \cdot X_{[a,b]} \tag{51}$$
$$X_{[a,b]} \cdot K_{[c,d,e,f]} \approx K_{[c,d,e,f]} \cdot X_{[a,b]} \tag{52}$$
$$(-1)_{[a]} \cdot K_{[b,c,d,e]} \approx K_{[b,c,d,e]} \cdot (-1)_{[a]} \tag{53}$$
$$(-1)_{[a]} \cdot (-1)_{[b]} \approx (-1)_{[b]} \cdot (-1)_{[a]} \tag{54}$$
$$K_{[a,b,c,d]} \cdot K_{[e,f,g,h]} \approx K_{[e,f,g,h]} \cdot K_{[a,b,c,d]} \tag{55}$$
$$X_{[a,c]} \cdot X_{[a,b]} \approx X_{[c,b]} \cdot X_{[a,c]} \tag{56}$$
$$X_{[b,c]} \cdot X_{[a,b]} \approx X_{[a,c]} \cdot X_{[b,c]} \tag{57}$$
$$X_{[a,b]} \cdot (-1)_{[a]} \approx (-1)_{[b]} \cdot X_{[a,b]} \tag{58}$$
$$X_{[a,e]} \cdot K_{[a,b,c,d]} \approx K_{[e,b,c,d]} \cdot X_{[a,e]} \tag{59}$$
$$X_{[b,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,e,c,d]} \cdot X_{[b,e]} \tag{60}$$
$$X_{[c,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,e,d]} \cdot X_{[c,e]} \tag{61}$$
$$X_{[d,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,c,e]} \cdot X_{[d,e]} \tag{62}$$
$$X_{[a,b]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,c,d]} \cdot X_{[a,b]} \cdot (-1)_{[b]} \cdot (-1)_{[d]} \tag{63}$$
$$X_{[b,c]} \cdot K_{[a,b,c,d]} \approx (-1)_{[a]} \cdot K_{[a,b,c,d]} \cdot (-1)_{[a]} \cdot K_{[a,b,c,d]} \cdot (-1)_{[a]} \tag{64}$$
$$X_{[c,d]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,c,d]} \cdot X_{[b,d]} \tag{65}$$
$$K_{[a,b,c,d]} \cdot K_{[b,d,e,f]} \approx K_{[b,d,e,f]} \cdot K_{[a,b,c,d]} \tag{66}$$
$$(-1)_{[a]} \cdot (-1)_{[e]} \cdot X_{[a,e]} \cdot \rho_{a,b,c,d,e,f,g,h} \approx \rho_{a,b,c,d,e,f,g,h} \cdot X_{[a,e]} \cdot (-1)_{[e]} \cdot (-1)_{[a]} \tag{67}$$

Figure 5: The relations in $\mathscr{R}_n$ from [18], for all valid choices of $a,b,c,d,e,f,g,h \in \mathbb{Z}$. We write $\rho_{a,b,c,d,e,f,g,h}$ for the substring $K_{[e,f,g,h]} \cdot K_{[a,b,c,d]} \cdot X_{[d,e]} \cdot K_{[a,b,c,d]} \cdot K_{[e,f,g,h]}$.

We say that $M_{[a_0,\dots,a_{m-1}]}$ is an *m-level operator of type M*. When $n = 8$ for example, $CCX_{0,1} = X_{[6,7]}$, $CCZ_{0,1} = (-1)_{[7]}$, and $K_{[4,5,6,7]}$ is a controlled K-gate. Define the following for $n > 3$.

$$\mathscr{G}_n = \left\{ (-1)_{[a]}, X_{[a,b]}, K_{[a,b,c,d]} \mid a,b,c,d \in \mathbb{Z} \text{ and } 0 \leq a < b < c < d < n \right\}$$

It was shown in [18] that $\mathrm{O}(n,\mathbb{D}) \cong \langle \mathscr{G}_n \mid \mathscr{R}_n \rangle$, where $\mathscr{R}_n$ is given in Figure 5. The goal of this section is to construct a sequence of Tietze transformations, starting from $\langle \mathscr{G}_8 \mid \mathscr{R}_8 \rangle$, such that the generators and relations describing the subgroup $W(E_8)$ are replaced by Σ_D and R_0, respectively. This process follows similarly to Section 4. However, one should note that $|\mathscr{R}_8| = 2113$ (see Appendix F.1). Inspection of $\mathscr{R}_n$ reveals that many of these relations are either definitional, or obtained through permutations of indices. For this reason, $\mathscr{R}_n$ is partially reduced before carrying out the aforementioned Tietze transformations. First, the permutations are eliminated via a sequence of **Rel(−)** transformations to obtain $\mathscr{R}_n^1$. Next, some redundant commutator relations are eliminated via a sequence of **Rel(−)** transformations to obtain $\mathscr{R}_n^2$. Finally, the derived generators are eliminated to obtain $\mathscr{R}_n^3$. All proofs can be found in Appendix F.

5.1 Permutation Groups and Reindexing

Let $[n] = \{0,1,\dots,n-1\}$ and $\mathrm{S}(n)$ denote the group of permutations on $[n]$. For $j,k \in [n]$, let $\tau_{j,k}$ denote the permutation that swaps j and k. For example, $\tau_{0,1}(0) = 1$, $\tau_{0,1}(1) = 0$, and $\tau_{0,1}(2) = 2$. The group $\mathrm{S}(n)$ is a finite reflection group generated by the *transpositions* $\{\tau_{j,j+1} \mid j \in [n]\}$ (see [16]). The *braiding relations*, which state that $\tau_{j,j+1} \circ \tau_{j+1,j+2} \circ \tau_{j,j+1} = \tau_{j+1,j+2} \circ \tau_{j,j+1} \circ \tau_{j+1,j+2}$ for all $j \in [n-2]$, together with the order relations are sound and complete for $\mathrm{S}(n)$ (see [17]). The standard representation of $\mathrm{S}(n)$ as a reflection group sends each $\tau_{j,k}$ to $X_{[j,k]}$. This means that every two-level operator of type X can be decomposed into sequence of transpositions. Intuitively, each $X_{[j,k]}$ acts by permuting the standard basis vectors $|j\rangle$ and $|k\rangle$, which can be achieved through a sequence of transpositions of basis vectors. Clearly, $\mathrm{S}(8) \hookrightarrow W(E_8) \leq O(8,\mathbb{D})$.

Many relations in $\mathscr{R}_n$ are related via permutation of indices. The *formal application of* σ to a word over $\mathscr{G}_n$ is defined inductively as follows.

$$\sigma(\varepsilon) = \varepsilon \qquad \sigma(X_{[a,b]} \cdot w) = X_{[\sigma(a),\sigma(b)]} \cdot \sigma(w)$$
$$\sigma((-1)_{[a]} \cdot w) = (-1)_{[\sigma(a)]} \cdot \sigma(w) \qquad \sigma(K_{[a,b,c,d]} \cdot w) = K_{[\sigma(a),\sigma(b),\sigma(c),\sigma(d)]} \cdot \sigma(w)$$

Note that $\sigma(w)$ may yield m-level operators with invalid indices. For example, $\tau_{1,2}(X_{[1,2]})$ yields $X_{[2,1]}$, which is not a valid two-level operator since $2 > 1$. The permutation σ is a *valid reindexing for w* if all symbols in $\sigma(w)$ are well-formed multi-level operators. If σ is valid for v and w, then σ is valid for $v \cdot w$. Conversely, if σ is valid for w, then σ is valid for all subwords in w. Consider, for example, the word $w = K_{[2,3,4,5]} \cdot K_{[3,5,6,7]}$ which appears on the left-hand side of an instance of Relation (55). Let $\sigma \in \mathrm{S}(8)$ be the cyclic permutation $7 \mapsto 5 \mapsto 3 \mapsto 1 \mapsto 6 \mapsto 4 \mapsto 2 \mapsto 0 \mapsto 7$. Then σ is a valid reindexing for w since $\sigma(w) = K_{[0,1,2,5]} \cdot K_{[1,3,4,5]}$. In Appendix F.2, we show that all valid reindexings are derivable using only the relations in $\mathscr{R}_\sigma = \{$Relations (47), (51), (52), (56), (57), (58), (59), (60), (61) and (62)$\}$.

5.2 Selecting Representative Relations for $\mathrm{O}(n, \mathbb{D})$

As a consequence of Section 5.1, many relations in $\mathscr{R}_n$ can be replaced by representative instances. For example, let r denote instance $(-1)_{[6]} \cdot (-1)_{[7]} \approx (-1)_{[7]} \cdot (-1)_{[6]}$ of Relation (54). Clearly $\sigma = \tau_{0,6} \circ \tau_{1,7}$ is a valid reindexing for r, where $\sigma(r)$ is $(-1)_{[0]} \cdot (-1)_{[1]} \approx (-1)_{[1]} \cdot (-1)_{[0]}$ Then by Appendix F.2, it is possible to derive $\sigma(r)$ from r using $\mathscr{R}_n \setminus \{\sigma(r)\}$. Then $\langle \mathscr{G}_n \mid \mathscr{R}_n \rangle \cong \langle \mathscr{G}_n \mid \mathscr{R}_n \setminus \{\sigma(r)\} \rangle$ by **Rel(−)**.

This process can be repeated, until all instances of Relation (54) have been eliminated, except for the representative relation r. In a similar fashion, Relations (48), (49), (53), (55), (63), (64), (65), (66) and (67) can be eliminated, since these relations do not appear in $\mathscr{R}_\sigma$. Then $\mathrm{O}(n, \mathbb{D}) \cong \langle \mathscr{G}_n \mid \mathscr{R}_n^1 \rangle$ where $\mathscr{R}_n^1$ is the set of representative relations (see Appendix F.3).

5.3 Selecting Representative Generators for $\mathrm{O}(n, \mathbb{D})$

Define the new generator set,

$$\mathscr{G}_n^1 = \left\{ X_{[a,b]}, \mid a, b, \in \mathbb{Z} \text{ and } 0 \leq a < b < n \right\} \cup \left\{ K_{[0,1,2,3]} \right\} \cup \left\{ (-1)_{[0]} \right\}.$$

Many of the generators in $\mathscr{G}_n$ are redundant in the sense that they may be constructed using only the generators in $\mathscr{G}_n^1$. This is because $\mathrm{S}(n) \hookrightarrow \mathscr{G}_n^1$, with $\mathscr{G}_n^1 \setminus \mathscr{G}_n$ consisting of valid indexings of either $K_{[0,1,2,3]}$ or $(-1)_{[0]}$. Furthermore, these reindexings follow from relations in $\mathscr{R}_n^1$. As an example, consider the instance $X_{[0,7]} \cdot (-1)_{[0]} \approx (-1)_{[7]} \cdot X_{[0,7]}$ of Relation (58). Using the order relation for $X_{[0,7]}$, the following derivation holds.

$$(-1)_{[7]} \leftarrow (-1)_{[7]} \cdot {X_{[0,7]}}^2 \leftarrow X_{[0,7]} \cdot (-1)_{[0]} \cdot X_{[0,7]}$$

Similarly, the original relation can be obtained from this new relation using the order relation for $X_{[0,7]}$. Then through a **Rel(+)** transformation followed by a **Rel(−)** transformation, the commutator relation $X_{[0,7]} \cdot (-1)_{[0]} \approx (-1)_{[7]} \cdot X_{[0,7]}$ can be replaced by the definitional relation $(-1)_{[7]} = X_{[0,7]} \cdot (-1)_{[0]} \cdot X_{[0,7]}$. This process can be repeated for all instances of Relation (58).

To derive the four-level operators of type K, it suffices to note that the following family of relations are valid with respect to $[\![\cdot]\!]_O^*$.

$$K_{[a,b,c,d]} \approx X_{[0,a]} \cdot X_{[1,b]} \cdot X_{[2,c]} \cdot X_{[3,d]} \cdot K_{[0,1,2,3]} \cdot X_{[3,d]} \cdot X_{[2,c]} \cdot X_{[1,b]} \cdot X_{[0,a]}$$

The cases where $\{a, b, c, d\} \cap \{0, 1, 2, 3\} \neq \varnothing$ can be handled using the techniques of Appendix F.2. These relations are introduced using a sequence of **Rel(+)** relations to obtain a new relation set R. In this relation set, all multi-level operators of type (-1) and K are defined in terms of $(-1)_{[0]}$ and $K_{[0,1,2,3]}$, respectively. As outlined in Appendix B.2, these defining relations can be used to eliminate all generators in $\mathscr{G}_n^1 \setminus \mathscr{R}_n^0$ via a finite sequence of Tietze transformations.

$$X_{[a,a+1]}{}^2 \approx \varepsilon \tag{68}$$
$$(-1)_{[0]}{}^2 \approx \varepsilon \tag{69}$$
$$K_{[0,1,2,3]}{}^2 \approx \varepsilon \tag{70}$$
$$X_{[b,b+1]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[b,b+1]} \tag{71}$$
$$X_{[c,c+1]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[c,c+1]} \tag{72}$$
$$(-1)_{[4]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot (-1)_{[4]} \tag{73}$$
$$(-1)_{[0]} \cdot (-1)_{[4]} \approx (-1)_{[4]} \cdot (-1)_{[0]} \tag{74}$$
$$K_{[0,1,2,3]} \cdot K_{[4,5,6,7]} \approx K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \tag{75}$$
$$X_{[a,a+1]} \cdot X_{[a,a+2]} \approx X_{[a+1,a+2]} \cdot X_{[a,a+1]} \tag{76}$$
$$X_{[a+1,b]} \cdot X_{[a,a+1]} \approx X_{[a,b]} \cdot X_{[a+1,b]} \tag{77}$$
$$X_{[0,1]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[0,1]} \cdot (-1)_{[1]} \cdot (-1)_{[3]} \tag{78}$$
$$X_{[1,2]} \cdot K_{[0,1,2,3]} \approx (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \tag{79}$$
$$X_{[2,3]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[1,3]} \tag{80}$$
$$K_{[0,1,2,3]} \cdot K_{[1,3,4,5]} \approx K_{[1,3,4,5]} \cdot K_{[0,1,2,3]} \tag{81}$$
$$(-1)_{[0]} \cdot (-1)_{[4]} \cdot X_{[0,4]} \cdot \rho \approx \rho \cdot X_{[0,4]} \cdot (-1)_{[4]} \cdot (-1)_{[0]} \tag{82}$$

Figure 6: The reduced relations in $\mathscr{R}_n^3$, for all valid choices of $a,b,c \in \mathbb{Z}$ where $b > 0$ and $c > 3$. We write ρ for the substring $K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \cdot X_{[3,4]} \cdot K_{[0,1,2,3]} \cdot K_{[4,5,6,7]}$.

The elimination process works as follows. Let $M \in \mathscr{G}_n^1 \setminus \mathscr{G}_n$. Then M appears in some defining relation $M \approx w$. If M appears in some relation $r \in R$, then every instance of M will be replaced by w. For example, Relation (54) will be replaced by the following relation.

$$(-1)_{[0]} \cdot X_{[0,5]} \cdot (-1)_{[0]} \cdot X_{[0,5]} \approx X_{[0,5]} \cdot (-1)_{[0]} \cdot X_{[0,5]} \cdot (-1)_{[0]}$$

We introduce the following abbreviations for simplicity of presentation.

$$(-1)_{[c]} = X_{[0,c]} \cdot (-1)_{[0]} \cdot X_{[0,c]} \qquad K_{[0,1,2,d]} = X_{[3,d]} \cdot K_{[0,1,2,3]} \cdot X_{[3,d]} \qquad K_{[0,1,c,d]} = X_{[2,c]} \cdot K_{[0,1,2,d]} \cdot X_{[2,c]}$$
$$K_{[0,b,c,d]} = X_{[1,b]} \cdot K_{[0,1,c,d]} \cdot X_{[1,b]} \qquad K_{[a,b,c,d]} = X_{[0,a]} \cdot K_{[0,b,c,d]} \cdot X_{[0,a]}$$

Denote this new set of relations $\mathscr{R}_n^2$. Then $\mathrm{O}(n,\mathbb{D}) \cong \langle \mathscr{G}_n^1 \mid \mathscr{R}_n^2 \rangle$.

5.4 Eliminating Redundant Relations

It will now shown that many relations in $\mathscr{R}_n^2$ are redundant. First, the braiding relations and order relations are used according to Appendix F.2 to eliminate all other relations over the two-level operators of type X. This reduced relation set is then used to show that all instances of Relations (51) and (52) can be derived using transpositions in place of swaps. Finally, it is shown that the relations Relations (59), (60), (61) and (62) are entirely redundant. All derivations can be found in Appendix F.4. This new set of relations is denoted $\mathscr{R}_n^3$, and can be found in Figure 6. Then via a sequence of **Rel(−)** transformations, the following presentation is obtained.

Theorem 5.1. $\mathrm{O}(n,\mathbb{D}) \cong \langle \mathscr{G}_n^1 \mid \mathscr{R}_n^3 \rangle$.

5.5 Introducing the $W(E_8)$ Generators

It this section, the circuit generators and relations for $W(E_8)$ are introduced. Since $CCX_{1,2} = X_{[6,7]}$, then without loss of generality, every instance of $X_{[6,7]}$ in $\mathscr{R}_8^3$ can be replaced by $CCX_{1,2}$. Next, the generators X_0 and $CX_{0,1}$ are introduced. This yields the following relations.

$$(r_X) : X_0 \approx X_{[0,4]} \cdot X_{[1,5]} \cdot X_{[2,6]} \cdot X_{[3,7]} \qquad (r_{CX}) : CX_{0,1} \approx X_{[2,6]} \cdot X_{[3,7]}$$

It turns out that the $K_{1,2}$ gate decomposes into a word over X_0 and $K_{[0,1,2,3]}$. This is because $K_{[0,1,2,3]}$ is a $K_{1,2}$ gate which is applied when qubit 0 is in state $|1\rangle$, and $X_0 \circ K_{[0,1,2,3]} \circ X_0$ is a $K_{1,2}$ gate which is applied

$$(-1)_{[0]}{}^2 \approx \varepsilon \tag{83}$$

$$K_{[0,1,2,3]}{}^2 \approx \varepsilon \tag{84}$$

$$X_{[1,2]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[1,2]} \tag{85}$$

$$X_{[2,3]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[2,3]} \tag{86}$$

$$X_{[3,4]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[3,4]} \tag{87}$$

$$X_{[4,5]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[4,5]} \tag{88}$$

$$X_{[5,6]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[5,6]} \tag{89}$$

$$X_{[6,7]} \cdot (-1)_{[0]} \approx (-1)_{[0]} \cdot X_{[6,7]} \tag{90}$$

$$X_{[4,5]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[4,5]} \tag{91}$$

$$X_{[5,6]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[5,6]} \tag{92}$$

$$X_{[6,7]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[6,7]} \tag{93}$$

$$(-1)_{[4]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot (-1)_{[4]} \tag{94}$$

$$(-1)_{[0]} \cdot (-1)_{[4]} \approx (-1)_{[4]} \cdot (-1)_{[0]} \tag{95}$$

$$K_{[0,1,2,3]} \cdot K_{[4,5,6,7]} \approx K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \tag{96}$$

$$X_{[0,1]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[0,1]} \cdot (-1)_{[1]} \cdot (-1)_{[3]} \tag{97}$$

$$X_{[1,2]} \cdot K_{[0,1,2,3]} \approx (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \tag{98}$$

$$X_{[2,3]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[1,3]} \tag{99}$$

$$K_{[0,1,2,3]} \cdot K_{[1,3,4,5]} \approx K_{[1,3,4,5]} \cdot K_{[0,1,2,3]} \tag{100}$$

$$(-1)_{[0]} \cdot (-1)_{[4]} \cdot X_{[0,4]} \cdot \rho \approx \rho \cdot X_{[0,4]} \cdot (-1)_{[4]} \cdot (-1)_{[0]} \tag{101}$$

Figure 7: The relations in $\mathscr{R}_8^4$, sufficient to extend from $W(E_8)$ to $O(8,\mathbb{D})$.

when qubit 0 is in state $|0\rangle$. Together, these two words compose to a $K_{1,2}$ gate without any controls. This yields the following relation.

$$(r_K) : K_{1,2} \approx K_{[4,5,6,7]} \cdot X_0 \cdot K_{[4,5,6,7]} \cdot X_0$$

The relations r_X, r_{CX}, and r_K can be validated with respect to $[\![\cdot]\!]_O$. These relations do not depend on one-another, so the generators in Σ_0 may be introduced via a sequence of **Gen(+)** transformations, as outlined in Appendix B.2. Likewise, the derived generators in $\Sigma_D \setminus \Sigma_0$ may be introduced via a sequence of **Gen(+)** transformations, as outlined in Appendix B.2. Finally, the relations in R_0 may be introduced, since Section 4.1 established the validity of these relations in $W(E_8)$, which is a subgroup of $O(8,\mathbb{D})$. This sequence of transformations yields $\mathrm{O}(8,\mathbb{D}) \cong \left\langle \mathscr{G}_8^1 \cup \Sigma_D \mid \mathscr{R}_8^3 \cup R_0 \cup \{r_X, r_{CX}, r_K\} \right\rangle$.

5.6 Eliminating the Multi-Level Operators

Using the generators in Σ_D and the relations in R_0, it is possible to eliminate all two-level operators of type X. As a first step, it must be shown that the two-level operators can be decomposed into circuits over Σ_D. This follows from the fact that Σ_D generates $W(E_8)$, and $\mathrm{S}(n) \hookrightarrow W(E_8)$.

$$X_{[0,1]} = X_0 \circ X_1 \circ CCX_{0,1} \circ X_1 \circ X_0 \qquad X_{[1,2]} = X_0 \circ CCX_{0,1} \circ CCX_{0,2} \circ CCX_{0,1} \circ X_0$$

$$X_{[2,3]} = X_0 \circ CCX_{0,1} \circ X_0 \qquad X_{[3,4]} = X_0 \circ X_2 \circ CCX_{0,1} \circ X_0 \circ CCX_{1,2} \circ CCX_{0,2} \circ CCX_{1,2} \circ X_0 \circ CCX_{0,1} \circ X_2 \circ X_0$$

$$X_{[4,5]} = X_1 \circ CCX_{0,1} \circ X_1 \qquad X_{[5,6]} = CCX_{0,1} \circ CCX_{0,2} \circ CCX_{0,1}$$

These relations can be validated with respect to $[\![\cdot]\!]_O$, and consequently introduced via a sequence of **Gen(+)** operations. These definitional relations can then be used to eliminate the two-level operators of type X, as outlined in Appendix B.2.

In this new presentation, then relations Relations (47), (56) and (57) are replaced by relations over Σ_D. Since R_0 is complete for $W(E_8)$, then these relations can be derived from R_0. Consequently, these relations can be eliminated with a sequence of **Rel(−)** transformations. This yields a new set of relations, denoted $\mathscr{R}_8^4$, which can be found in Figure 7. For simplicity of presentation, we used $X_{[0,1]}$ through to $X_{[6,7]}$ as abbreviations for the circuits given above. Furthermore, we take $CCZ = (-1)_{[7]}$ to be a generator with $(-1)_{[0]}$ an alias for $X_0 \cdot X_1 \cdot CCZ \cdot X_1 \cdot X_0$. Then define $\Sigma_1 = \Sigma_D \cup \{K_{[0,1,2,3]}, CCZ\}$ and $R_1 = R_0 \cup \mathscr{R}_8^4$, where $K_{[0,1,2,3]}$ corresponds to a negatively controlled K gate.

Theorem 5.2. $\mathrm{O}(8,\mathbb{D}) \cong \langle \Sigma_1 \mid R_1 \rangle$.

$$H_2 \cdot X_0 \approx X_0 \cdot H_2 \tag{102}$$
$$H_2 \cdot CX_{0,1} \approx CX_{0,1} \cdot H_2 \tag{103}$$
$$H_2 \cdot CCX_{1,2} \approx K_{0,1} \cdot K_{1,2} \cdot CCZ \cdot K_{1,2} \cdot K_{0,1} \cdot H_2 \tag{104}$$
$$H_2 \cdot CCZ \approx CCX_{0,1} \cdot H_2 \tag{105}$$
$$H_2 \cdot K_{1,2} \approx K_{1,2} \cdot H_2 \tag{106}$$
$$H_2 \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot H_2 \tag{107}$$
$$H_2 \cdot H_2 \approx \varepsilon \tag{108}$$

Figure 8: Additional relations for $\mathsf{TofH}(3)$.

5.7 A Minimal Generating Set for $\mathrm{O}(8,\mathbb{D})$

It turns out that CCZ and $K_{[0,1,2,3]}$ can be defined in terms of one-another, given that generators in Σ_D. The decompositions are as follows.

$$CCZ = K_{1,2} \circ CZ_{1,2} \circ X_0 \circ K_{[0,1,2,3]} \circ X_0 \circ CZ_{1,2} \circ K_{[0,1,2,3]} \circ X_{[5,6]} \qquad K_{[0,1,2,3]} = (K_{1,2} \circ CCZ)^3 \circ X_{[5,6]}$$

Given this observation, it seems natural to eliminate the $K_{[0,1,2,3]}$, given that it is not a common generator in quantum computation. However, the set $\Sigma_D \cup \{CCZ\}$ is minimal, whereas the set $\Sigma_D \cup \{K_{[0,1,2,3]}\}$ is not. In other words, choosing the generator $K_{[0,1,2,3]}$ enables a smaller generating set, whereas choosing the generator CCZ allows for more conventional circuit decompositions. For this reason, we choose to keep both $K_{[0,1,2,3]}$ and CCZ in our presentation. The minimality of these generating sets are proven in Appendix E, using the same techniques as in Section 4.5.

Theorem 5.3. *The following generating sets are minimal for* $\mathrm{O}(8,\mathbb{D})$.

1. $\Sigma_K = \left\{X_0, CX_{0,1}, CCX_{1,2}, K_{[0,1,2,3]}\right\}$.
2. $\Sigma_Z = \{X_0, CX_{0,1}, CCX_{1,2}, K_{1,2}, CCZ\}$

6 Extending to the 3-Qubit Toffoli-Hadamard Circuits

We now give a presentation of $\mathsf{TofH}(3)$, by leveraging the presentation of $\mathrm{O}(8,\mathbb{D})$ found in Section 5. The argument in this section closely follows [18, Section 5]. From [3], it is known that $\mathsf{TofH}(3)$ is obtained by adding the generator H_2 to $\mathrm{O}(8,\mathbb{D})$. Let $\Sigma_2 = \Sigma_1 \cup \{H_2\}$ and R_2 extend the set R_1 with all relations found in Figure 8. Using the relations in Figure 8, the generator H_2 can be moved from the left-hand side to the right-hand side of any word over Σ_1. Since H_2 is self-inverse, this is sufficient to decide equality in $\mathsf{TofH}(3)$.

The proof proceeds as follows. In Lemma 6.1, it is shown that H_2 commutes with every word in over Σ_1 using only the relations in R_2. This is used in Lemma 6.2, to show that every word over Σ_2 can be rewritten as a word over Σ_1, followed by at most one H_2 gate. Since $R_1 \subseteq R_2$ is a complete equational theory for $\mathrm{O}(8,\mathbb{D})$ with every element of $\mathsf{TofH}(3)$ of the form described in Lemma 6.2, it follows that R_2 is a complete equational theory for $\mathsf{TofH}(3)$ (see Theorem 6.3).

Lemma 6.1. *If $w \in \Sigma_1^*$, then there exists a $w' \in \Sigma_1^*$ such that $H_2 \cdot w \sim_{R2} w' \cdot H_2$.*

Proof. The proof follows by induction on $|w|$.

- **Base Case**. If $|w| = 0$, then $H_2 \cdot w = w \cdot H_2$. Then $H_2 \cdot w \sim_{R2} w \cdot H_2$ by the transitivity of $(\sim_{R2})$.
- **Inductive Case**. Assume that for some $k \in \mathbb{N}$, if $|w| = k$, then there exists a $w' \in \Sigma_1^*$ such that $H_2 \cdot w \sim_{R2} w' \cdot H_2$.

- **Inductive Step**. Assume that $|w| = k+1$. Then there exists a $u \in \Sigma_1^*$ and $x \in \Sigma_1$ such that $x \cdot u = w$. It follows by one of Relations (102), (103), (104), (105), (106) and (107), that there exits a $u' \in \Sigma_1^*$ such that $H_2 \cdot w \sim_{R2} u' \cdot H_2 \cdot u$. Since $|u| = k$, then by the inductive hypothesis, there exists a $w' \in \Sigma_1^*$ such that $H_2 \cdot u \sim_{R2} w' \cdot H_2$. Then $u' \cdot H_2 \cdot u \sim_{R2} u' \cdot w' \cdot H_2$. Then $H_2 \cdot u \sim_{R2} u' \cdot w' \cdot H_2$ by the transitivity of $(\sim_{R2})$, and the inductive case holds.

Then by the principle of induction, there exists a $w' \in \Sigma_1^*$ such that $H_2 \cdot w \sim_{R2} w' \cdot H_2$. □

Lemma 6.2. *If $w \in \Sigma_2^*$, then there exists some $w' \in \Sigma_1^*$ and $\ell \in \{0,1\}$ such that $w \sim_{R_2} w' \cdot H_2^\ell$.*

Proof. Let $f : \Sigma_2^* \to \mathbb{N}$ map each word $w \in \Sigma_2^*$ to the number of H_2 symbols in w. The proof follows by induction on $f(w)$.

- **Base Case**. If $f(w) = 0$, then $w \in \Sigma_1^*$. Then $w \sim_{R2} w \cdot H_2^0$ by the reflexivity of $(\sim_{R2})$
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $f(w) = k$, then there exists some $w' \in \Sigma_1^*$ and $\ell \in \{0,1\}$ such that $w \sim_{R2} w' \cdot H_2^\ell$.
- **Inductive Step**. Assume that $f(w) = k+1$. Then there exists $w_1 \in \Sigma_2^*$ and $w_2 \in \Sigma_1^*$ such that $w = w_1 \cdot H_2 \cdot w_2$ with $f(w_1) = f(w) - 1$. Then by Lemma 6.1, $w \sim_{R2} w_1 \cdot w_2' \cdot H_2$ for some $w_2' \in \Sigma_1^*$. Then $f(w_1 \cdot w_2') = f(w_1) = f(w) - 1$. By the inductive hypothesis, there exists some $w_3 \in \Sigma_1^*$ and $\ell \in \{0,1\}$ such that $w_1 \cdot w_2' \sim_{R2} w_3 \cdot H_2^\ell$. Then $w \sim_{R2} w_3 \cdot H_2^{\ell+1}$. If $\ell = 0$, then $w \sim_{R2} w_3 \cdot H_2$ and we are done. Otherwise, if $\ell = 1$, then $w \sim_{R2} w_3$ by Relation (108). In either case, there exists an $\ell' \in \{0,1\}$ such that $w \sim_{R2} w_3 \cdot H_2^{\ell'}$ and the inductive step holds.

Then by the principle of induction, there exists some $w' \in \Sigma_1^*$ and $\ell \in \{0,1\}$ such that $w \sim_{R_2} w' \cdot H_2^\ell$. □

Theorem 6.3. *For all $w_1, w_2 \in \Sigma_2^*$, $[\![w_1]\!]_H^* = [\![w_2]\!]_H^*$ if and only if $w_1 \sim_{R2} w_2$.*

Proof. It follows by matrix multiplication that the relations in Figure 8 are sound. It remains to be shown that the relations in Figure 8 are complete. Let $w_1 \in \Sigma_2^*$ and $w_2 \in \Sigma_2^*$ such that $[\![w_1]\!]_H^* = [\![w_2]\!]_H^*$. By Lemma 6.2, there exists $\ell_1, \ell_2 \in \{0,1\}$ and $w_1', w_2' \in \Sigma_1^*$ such that $w_1 \sim_{R2} w_1' \cdot H_2^{\ell_1}$ and $w_2 \sim_{R2} w_2' \cdot H_2^{\ell_2}$. Since $[\![w_1]\!]_H^* \in \mathrm{O}(8,\mathbb{D}) \cong \langle \Sigma_1, R_1 \rangle$ with $R_1 \subseteq R_2$, then $w_1' \cdot H_2^{\ell_1} \sim_{R2} w_2' \cdot H_2^{\ell_1}$. Assume for the intent of contradiction that $\ell_1 \neq \ell_2$. Then $[\![w_1']\!]_H^* = [\![w_2']\!]_H^* \circ [\![H_2]\!]_H$. Then $[\![w_2']\!]_H^* \circ [\![H_2]\!]_H^* \in \mathrm{O}(8,\mathbb{D})$. However, $[\![w_2']\!]_H^* \circ [\![H_2]\!]_H$ has a denominator of the form $1/(2^k\sqrt{2})$, and therefore $[\![w_2']\!]_H^* \circ [\![H_2]\!]_H \notin \mathrm{O}(8,\mathbb{D})$. By contradiction, $\ell_1 = \ell_2$. Since $\ell_1 = \ell_2$, then $w_1 \sim_{R2} w_2$ by the transitivity and symmetry of $(\sim_{R2})$. Since w_1 and w_2 were arbitrary, then the relations in Figure 8 are complete. □

7 Conclusion

We used the geometry of $W(E_8)$ to obtain a circuit presentation for the 3-qubit Toffoli-K circuits, and then leveraged [18] to obtain a finite presentation of 3-qubit Toffoli-Hadamard circuits. Our presentation contains 65 relations, compared to the 2113 relations of [18]. There are several directions for future work. We hope to simplify our presentation by further reducing the number of relations. In addition, we plan to explore the structural properties of the group of 3-qubit Toffoli-Hadamard circuits and of its subgroups. In particular, it is known that the group $\mathrm{O}(8,\mathbb{D})$ is generated by reflections, but it is not known whether this group can be presented as an (infinite) Coexeter group. From an applied perspective, we also hope to explore applications of these presentations to circuit optimization and equivalence checking.

References

[1] Dorit Aharonov (2003): *A simple proof that Toffoli and Hadamard are quantum universal*. arXiv:quant-ph/0301040.

[2] Matthew Amy, Jianxin Chen & Neil J. Ross (2018): *A Finite Presentation of CNOT-Dihedral Operators*. EPTCS 266, pp. 84–97, doi:10.4204/eptcs.266.5.

[3] Matthew Amy, Andrew Glaudell & Neil Ross (2020): *Number-Theoretic Characterizations of Some Restricted Clifford+T Circuits*. Quantum 4, p. 252, doi:10.22331/q-2020-04-06-252.

[4] Matthew Amy, Andrew N. Glaudell, Sarah Meng Li & Neil J. Ross (2023): *Improved Synthesis of Toffoli-Hadamard Circuits*. In: Reversible Computation, Springer-Verlag, pp. 169–209, doi:10.1007/978-3-031-38100-3_12.

[5] Matthew Amy, Neil J. Ross & Scott Wesley (2024): *Supplement: A Sound and Complete Equational Theory for 3-Qubit Toffoli-Hadamard Circuits*. Available as an ancillary file from the arXiv page of this paper.

[6] Franz Baader & Tobias Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press, doi:10.1017/CBO9781139172752.

[7] Xiaoning Bian & Peter Selinger (2023): *Generators and relations for 2-qubit Clifford+T operators*. EPTCS 394, pp. 13–28, doi:10.4204/eptcs.394.2.

[8] Xiaoning Bian & Peter Selinger (2023): *Generators and relations for 3-qubit Clifford+CS operators*. EPTCS 384, pp. 114–126, doi:10.4204/eptcs.384.7.

[9] Ronald V. Book & Friedrich Otto (1993): *String-Rewriting Systems*. Springer, doi:10.1007/978-1-4613-9771-7.

[10] Maria Luisa Dalla Chiara, Antonio Ledda, Giuseppe Sergioli & Roberto Giuntini (2013): *The Toffoli-Hadamard gate system: an algebraic approach*. Journal of Philosophical Logic 42, pp. 467–481, doi:10.1007/s10992-013-9271-9.

[11] Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix & Benoît Valiron (2023): *A Complete Equational Theory for Quantum Circuits*. In: LiCS, IEEE, pp. 1–13, doi:10.1109/LICS56636.2023.10175801.

[12] J. H. Conway & N. J. A. Sloane (1987): *Sphere-Packings, Lattices, and Groups*. Springer-Verlag, doi:10.1007/978-1-4757-6568-7.

[13] Jörg Endrullis, Herman Geuvers, Jakob Grue Simonsen & Hans Zantema (2011): *Levels of undecidability in rewriting*. Information and Computation 209(2), pp. 227–245, doi:10.1016/j.ic.2010.09.003.

[14] Adam P. Goucher (2020): *Minimalistic Quantum Computation*. `https://cp4space.hatsya.com/2020/05/10/minimalistic-quantum-computation/`. Accessed: 2023-11-26.

[15] Simon Henry & Samuel Mimram (2022): *Tietze Equivalences as Weak Equivalences*. Applied Categorical Structures 30(3), pp. 453–483, doi:10.1007/s10485-021-09662-w.

[16] James E. Humphreys (1990): *Reflection Groups and Coxeter Groups*. Cambridge Studies in Advanced Mathematics, Cambridge University Press, doi:10.1017/CBO9780511623646.

[17] D. L. Johnson (1990): *Presentations of Groups*. Cambridge University Press, doi:10.1017/CBO9781139168410.

[18] Sarah Meng Li, Neil J. Ross & Peter Selinger (2021): *Generators and Relations for the Group On$(\mathbb{Z}[1/2])$*. EPTCS 343, pp. 210–264, doi:10.4204/eptcs.343.11.

[19] Justin Makary, Neil J. Ross & Peter Selinger (2021): *Generators and Relations for Real Stabilizer Operators*. EPTCS 343, p. 14–36, doi:10.4204/eptcs.343.2.

[20] Leonardo de Moura & Nikolaj Bjørner (2008): *Z3: An Efficient SMT Solver*. In: TACAS, Springer-Verlag, pp. 337–340, doi:10.5555/1792734.1792766.

[21] Michel Planat (2011): *Clifford group dipoles and the enactment of Weyl/Coxeter group $W(E8)$ by entangling gates*. Gen. Math. Notes 2(1), pp. 96–113, doi:10.22331/q-2020-04-06-252.

[22] Peter Selinger (2015): *Generators and Relations for n-Qubit Clifford Operators*. *LMCS* 11(2:10), pp. 1–17, doi:10.2168/LMCS-11(2:10)2015.

[23] Yaoyun Shi (2003): *Both Toffoli and controlled-NOT need little help to do universal quantum computing*. *Quantum Info. Comput.* 3(1), pp. 84–92, doi:10.5555/2011508.2011515.

[24] Maryna S. Viazovska (2017): *The sphere packing problem in dimension 8*. *Annals of Mathematics* 185(3), pp. 991–1015, doi:10.4007/annals.2017.185.3.7.

[25] Renaud Vilmart (2019): *A ZX-calculus with triangles for Toffoli-Hadamard, Clifford+T, and beyond*. *EPTCS* 287, pp. 313–344, doi:10.4204/eptcs.287.18.

[26] Renaud Vilmart (2023): *Completeness of sum-over-paths for Toffoli-Hadamard and the dyadic fragments of quantum computation*. In: *CSL*, *LIPIcs* 252, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 36:1–36:17, doi:10.4230/LIPIcs.CSL.2023.36.

A Semantic Tietze Transformations

This section recalls what it means for a function to induce a monoid homomorphism. This is then used to prove the soundness and completeness of Tietze transformations with respect to semantic interpretations. The majority of this section is dedicated to proving that all induced homomorphisms are isomorphisms, and in the case of **Gen**$(+)$, the extension is unique. The uniqueness of this extension is necessary to prove that each generator in Σ_D has the intended matrix semantics.

A.1 Induced Monoid Homomorphisms

Let Σ be an alphabet and M a monoid. For each function $f : \Sigma \to M$, define the function $f^* : \Sigma^* \to M$ such that $f^*(x_1 \cdot x_2 \cdots x_n) = 1_M \circ f(x_1) \circ f(x_2) \circ \cdots \circ f(x_n)$ for all $x_1 \cdot x_2 \cdots x_n \in \Sigma^*$. It can then be shown that f^* is the unique monoid homomorphism such that $f^*(x) = f(x)$ for all $x \in \Sigma$ [17]. Given a set of relations $R \subseteq \Sigma^* \times \Sigma^*$, it can then be asked whether f induces a monoid homomorphism between $G = \langle \Sigma \mid R \rangle$ and M. This question is answered by the following theorem.

Theorem A.1 ([9])**.** *Let M and $G = \langle \Sigma \mid R \rangle$ be monoids with $f : \Sigma \to M$. There there exists a unique monoid homomorphism $\varphi : G \to M$ such that $f^* = \varphi \circ \pi_G$ if and only if $f^*(q) = f^*(r)$ for all $q \approx_R r$. In this case,* $\mathrm{Im}(\varphi) = \langle \varphi(\pi_G(\Sigma)) \rangle$.

Theorem A.1 characterizes when f induces a monoid homomorphism, and how to construct this induced homomorphism φ. It can then be asked how the construction of φ interacts with the introduction or elimination of generators. As outlined by the following theorems, the elimination of generators corresponds to certain restrictions of φ, whereas the introduction of generators corresponds to certain unique extensions of φ.

Lemma A.2. *Let Σ be an alphabet, $x \in \Sigma$, $\Sigma' = \Sigma \setminus \{x\}$, and $D = \{x \approx w\}$ for some $w \in (\Sigma')^*$. If $q \in \Sigma^*$, then there exists a $q' \in (\Sigma')^*$ such that $q \sim_D q'$.*

Proof. Let $f : \Sigma^* \to \mathbb{N}$ count the x symbols in a word. Then the proof follows by induction on $f(q)$.

- **Base Case**. If $f(q) = 0$, then $q \in (\Sigma')^*$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $f(q) = k$, then there exists some $q' \in (\Sigma')^*$ such that $q \sim_D q'$.
- **Inductive Step**. Assume $f(q) = k+1$. Since $f(q) > 1$, then there exists some $u, v \in \Sigma^*$ such that $q = u \cdot x \cdot v$. Since $k+1 = f(q) = f(u \cdot x \cdot v) = f(u) + f(x) + f(v) = f(u) + f(v) + 1$, then $k = f(u) + f(v)$. Since $x \approx_D w$, then $q \sim_D u \cdot w \cdot v$ with $f(u \cdot w \cdot v) = f(u) + f(v) = k$. Then by the inductive hypothesis, there exists some $q' \in (\Sigma')^*$ such that $u \cdot w \cdot v \sim_D q'$. Then $q \sim q'$.

By the principle of induction, there exists some $q' \in (\Sigma')^*$ such that $q \sim_D q'$. □

Theorem A.3. *Let M and $G = \langle \Sigma \mid R \rangle$ be monoids with $f : \Sigma \to M$, and $H = \langle \Sigma \cup \{x\} \mid R \cup \{x \approx w\} \rangle$ be a monoid for some $x \notin \Sigma$ and $w \in \Sigma^*$. Define $g : \Sigma \cup \{x\} \to M$ such that $g|_\Sigma = f$ and $g : x \mapsto f^*(w)$. If f induces a monoid homomorphism from G to M, then g is the unique extension of f to induce a monoid homomorphism from H to M. Furthermore, if f induces an injection (resp. surjection) from G to M, then g induces an injection (resp. surjection) from H to M.*

Proof. Let $\Pi = \Sigma \cup \{x\}$ and $Q = R \cup \{x \approx w\}$. Assume that f induces a homomorphism from G to M.

- (**Induced Hom**). Let $q \approx_Q r$. Then either $q \approx_R r$ or $(q,r) = (x,w)$. First, assume that $q \approx_R r$. Then $f^*(q) = f^*(r)$ by Theorem A.1. Then $g^*(q) = (g|_\Sigma)^*(q) = f^*(q) = f^*(r) = (g|_\Sigma)^*(r) = g^*(r)$. Next, assume that $(q,r) = (x,w)$. Then $g^*(x) = g(x) = f^*(w) = (g|_\Sigma)^*(w) = g^*(w)$. In either case $g^*(q) = g^*(r)$. Since $q \approx_R r$ was arbitrary, then $g^*(q) = g^*(r)$ for all $q \approx_R r$. Then g induce a monoid homomorphism from H to M by Theorem A.1.

- (**Uniqueness**). Assume that $k : \Sigma \cup \{x\} \to M$ is an extension of f which induces a monoid homomorphism. Since k induces a monoid homomorphism and $x \approx_Q w$, then $k^*(x) = k^*(w)$ by Theorem A.1. Then $k(x) = k^*(x) = k^*(w) = (k|_\Sigma)^*(w) = f^*(w) = g(x)$ by construction of g. Moreover, since $k|_\Sigma = f = g|_\Sigma$, then $k = g$. Since k was arbitrary, then g is unique.

- (**Injectivity**). Let f induce φ and g induce ρ. Assume that φ is injective. Let $q, r \in \Pi^*$ such that $\rho(\pi_H(q)) = \rho(\pi_H(r))$. Since $x \approx_Q w$, then by Lemma A.2 there exists $q', r' \in \Sigma^*$ such that $q \sim_Q q'$ and $r \sim_Q r'$. Then $\pi_H(q) = \pi_H(q')$ and $\pi_H(r) = \pi_H(r')$. Then $\rho(\pi_H(q')) = \rho(\pi_H(r'))$. Then $g^*(q') = g^*(r')$. Since $q', r' \in \Sigma^*$, then $f^*(q') = f^*(r')$. Then $\varphi(\pi_G(q')) = \varphi(\pi_G(r'))$. Since φ is injective, then $\pi_G(q') = \pi_G(r')$. Then $q' \sim_R r'$. Since $R \subseteq Q$, then $q' \sim_Q r'$. Then $\pi_H(q') = \pi_H(r')$. Then $\pi_H(q) = \pi_H(r)$. Since q and r were arbitrary, then ρ is injective.

- (**Surjectivity**). Let f induce φ and g induce ρ. Assume that φ is surjective. Since $\varphi(\pi_G(y)) = f^*(y) = g^*(y) = \rho(\pi_H(y))$ for each $y \in \Sigma$, then $\varphi(\pi_G(\Sigma)) \subseteq \mathrm{Im}(\rho)$. Since φ is surjective, then $H = \mathrm{Im}(\varphi) = \langle \varphi(\pi_G(\Sigma)) \rangle \leq \mathrm{Im}(\rho) \leq H$ and ρ is surjective.

Therefore, g is the unique extension of f to induce a monoid homomorphism from H to M, with g inducing an injection (resp. surjection) whenever f induces an injection (resp. surjection). □

Theorem A.4. *Let M and $G = \langle \Sigma \mid R \rangle$ be monoids with $f : \Sigma \to M$ and $H = \langle \Pi \mid Q \rangle$ where $\Pi = \Sigma \setminus \{x\}$ for some $x \in \Sigma$ and $Q = R \setminus \{x \approx w\}$ for some $x \approx_R w$. If $Q \subseteq \Pi^* \times \Pi^*$ and f induces a monoid homomorphism from G to M, then $f|_\Pi$ induces a monoid homomorphism from H to M. Furthermore, if f induces an injection (resp. surjection) from G to M, then $f|_\Pi$ induces an injection (resp. surjection) from H to M.*

Proof. Assume that f induces a homomorphism from G to M.

- (**Induced Hom**). Since f induces a monoid homomorphism, then $f^*(q) = f^*(r)$ for all $q \approx_R r$ by Theorem A.1. Since $Q \subseteq R$ and $Q \subseteq \Pi^* \times \Pi^*$, then $f|_\Pi^*(q) = f^*(q) = f^*(r) = f|_\Pi^*(r)$ for all $q \approx_Q r$. Then $f|_\Pi$ induces a monoid homomorphism from H to M by Theorem A.1.

- (**Injectivity**). Let f induce φ and $f|_\Pi$ induce ρ. Assume that φ is injective. Let $q, r \in \Pi^*$ such that $\rho(\pi_H(q)) = \rho(\pi_H(r))$. Then $f|_\Pi^*(q) = f|_\Pi^*(r)$. Then $f^*(q) = f^*(r)$. Then $\varphi(\pi_G(q)) = \varphi(\pi_G(r))$. Since φ is an injective, then $\pi_G(q) = \pi_G(r)$. Then $q \sim_R r$. Since $Q = R \setminus \{x \approx w\}$, $Q \subseteq \Pi^* \times \Pi^*$, and $q, r \in \Pi^*$, then $q \sim_Q r$. Then $\pi_H(q) = \pi_H(r)$. Since q and r were arbitrary, then ρ is injective.

- (**Surjectivity**). Let f induce φ and $f|_\Pi$ induce ρ. Assume that φ is surjective. Since $\varphi(\pi_G(y)) = f^*(y) = f|_\Pi^*(y) = \rho(\pi(y)) \in \mathrm{Im}(\rho)$ for each $y \in \Pi$, then $\varphi(\pi_G(\Pi)) \subseteq \mathrm{Im}(\rho)$. Since $x \approx_R w$, then $\pi_G(x) = \pi_G(w)$, and consequently $\varphi(\pi_G(x)) = \varphi(\pi_G(w)) = f^*(w) = f|_\Pi^*(w) = \rho(\pi_H(w)) \in \mathrm{Im}(\rho)$. Then $\varphi(\pi_G(\Sigma)) \subseteq \mathrm{Im}(\rho)$. Since φ is surjective, then $H = \mathrm{Im}(\varphi)\langle \varphi(\pi_G(\Sigma)) \leq \mathrm{Im}(\rho) \leq H$ and ρ is surjective.

Therefore, $f|_\Pi$ induces a monoid homomorphism from H to M with $f|_\Pi$ inducing an injection (resp. surjection) whenever f induces an injection (resp. surjection). □

A.2 Semantic Interpretations and Relations

Let $G = \langle \Sigma \mid R \rangle$ be a monoid presentation with an interpretation $[\![\cdot]\!]_G : \Sigma \mapsto H$. The **Rel(+)** transformation states that if $r \in \Sigma^*$ and $q \in \Sigma^*$ with $r \sim_R q$, then $\langle \Sigma \mid R \rangle \cong \langle \Sigma \mid R \cup \{r\} \rangle$. In practice, deriving q from r can be challenging, and on a theoretical level, this is known to be undecidable [13]. However, it is rarely the case that one would try to prove $r \sim_R q$ without some intuition that $\pi_G(r) = \pi_G(q)$. In the case of this paper, this intuition comes from knowledge about operators in $O(8, \mathbb{D})$. For example, if $M \circ N = A \circ B$, then for any complete set of relations R, it must be the case that $M \cdot N \sim_R A \cdot B$. More generally, if $[\![r]\!]_\Sigma^* = [\![q]\!]_\Sigma^*$ with $[\![\cdot]\!]_\Sigma^*$ inducing an injection, then $r \sim_R q$. This claim is established by the following theorem, and used freely throughout the paper to simplify derivations.

Definition A.5 (Valid Semantic Interpretation)**.** A semantic interpretation $[\![\cdot]\!]_\Sigma : G \to H$ for a presentation $G = \langle \Sigma \mid R \rangle$ is valid if $[\![q]\!]_\Sigma^* = [\![r]\!]_\Sigma^*$ for all $r \approx_R q$.

Theorem A.6. *Let $G = \langle \Sigma \mid R \rangle$ be a presentation with a valid semantic interpretation $[\![\cdot]\!]_G : \Sigma \mapsto H$. If $[\![\cdot]\!]_G$ is injective and $[\![q]\!]_\Sigma^* = [\![r]\!]_\Sigma^*$, then $q \sim_R r$.*

Proof. Since $[\![\cdot]\!]_\Sigma$ is valid, then by Theorem A.1, $[\![\cdot]\!]_\Sigma$ induces a monoid homomorphism $\varphi : G \to H$ such that $[\![\cdot]\!]_\Sigma^* = \varphi \circ \pi_G$. Assume that $[\![q]\!]_\Sigma^* = [\![r]\!]_\Sigma^*$. Then $\varphi(\pi_G(q)) = [\![q]\!]_\Sigma^* = [\![r]\!]_\Sigma^* = \varphi(\pi_G(r))$. Since φ is injective, then $\pi_G(q) = \pi_G(r)$. Then $q \sim_R r$. □

A.3 Semantics and Generator Introduction

In the previous section, it was assumed that $[\![\cdot]\!]_\Sigma$ inducted an injection. This is a reasonable assumption. For example, if $G = \langle \Sigma \mid R \rangle$ is a presentation for H, then there exists an isomorphism $G \cong H$ from which $[\![\cdot]\!]_\Sigma$ can be extracted.

This can become problematic when trying to translate a known presentation to a desired generator set. Assume that $\langle \Pi \mid Q \rangle$ is a known presentation with a semantic interpretation $[\![\cdot]\!]_\Pi : \Pi \to H$, from with a presentation $\langle \Sigma \mid R \rangle$ is derivable via a sequence of Tietze transformations. One would hope that after each Tietze transformation, the semantic interpretation continues to induce an injection, so that Theorem A.6 continues to hold. Furthermore, one would hope that after all of the Tietze transformations, $[\![\cdot]\!]_\Sigma$ is a valid semantic interpretation.

It will be shown that under reasonable assumptions, all Tietze transformations satisfy these assumptions. The first concern is answered by Theorem A.3 and Theorem A.4, which state that after each **Gen(+)** or **Gen(−)** transformation, $[\![\cdot]\!]_\Pi$ continues to induce a monoid homomorphism (resp. injection, surjection, isomorphism). The only time injectivity might fail is after a new generator is added. In **Gen(+)**, the symbol $x \in \Sigma$ becomes an alias for $w \in \Pi^*$. Then it suffices to check that $[\![x]\!]_\Sigma = [\![w]\!]_\Pi^*$.

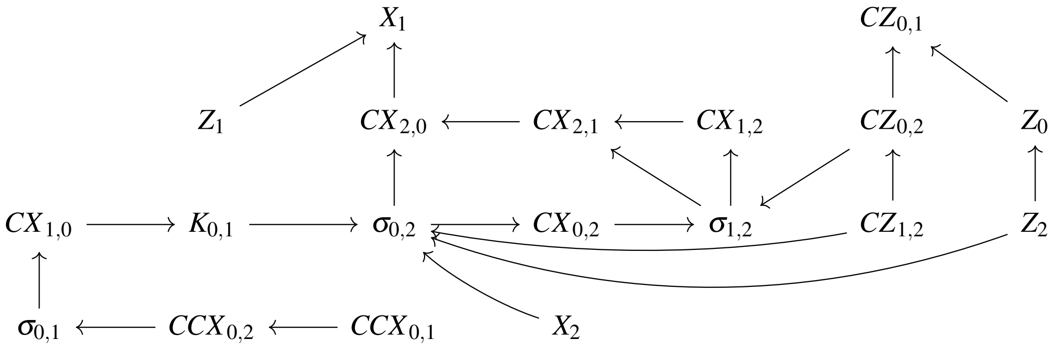


Figure 9: The derived generator graph for the defining relations in R_0.

B From Derivations to Tietze Transformations

This section describes higher-level structures to reason about Tietze transformations. It is proven that each structure corresponds to a valid sequence of Tietze transformations, and is therefore sound for the isomorphism of finite monoid presentations. The structures described in this section are used throughout Section 4 and Section 5.

B.1 Digraphs and Termination

This section reviews the basics of directed graphs. A directed graph is a tuple (V,E) such that $V \subseteq E \times E$. A vertex v is a *child* of u if $(u,v) \in E$. A vertex v is a *parent* of u if $(v,u) \in E$. A *path* in (V,E) is some sequence $(u_0, u_1, \ldots, u_n)$ over V such that $(u_{k-1}, u_k) \in E$ for all $k \in \{1,2,\ldots,n\}$. A directed graph (V,E) is *acyclic* if $u_0 \neq u_n$ for all paths $(u_0, u_1, \ldots, u_n)$ in (V,E).

Lemma B.1 ([6, Sect. 2.2])**.** *If a digraph (V,E) is finite and acyclic, then for every vertex $v \in V$, there exists some path of length n ending (resp. starting) at v such that every path ending (resp. starting) at v has length at most n. In particular, there exists a vertex $v \in V$ such that v has no children (resp. parents) in (V,E).*

Note that the original statement of Lemma B.1 concerned the termination of abstract rewriting systems (which can be modelled using paths through digraphs). In the proofs that follow, this intuition is useful to keep in mind. Indeed, Lemma B.1 is used to argue that certain rewriting procedures terminate, though the rewriting systems are never stated explicitly for simplicity.

B.2 Derived Generators and Tietze Transformations

In Section 4.1, it was claimed that 19 of the generators in Σ_D could be introduced freely, because their defining relations formed an acyclic dependency graph. Later, in Appendix E, the same argument was used to remove these 19 generators from the generating set for $W(E_8)$. Similar arguments appear throughout Section 5. The goal of this section is to establish both directions rigorously. First, formal definitions are given for defining relations and derived generator graphs (i.e., the graph of dependencies between the relations). Then, Theorem B.6 and Theorem B.11 are established to justify the claims of Section 4.1 and Section 4.5 respectively. Finally, it is shown that these theorems apply to the generators in Section 4.

Definition B.2 (Defining Relations)**.** Let Σ be an alphabet. A Σ-*defining relation* for $x \in \Sigma$ is a relation of the form $x \approx w$ where $w \in \Sigma^*$. A *family of* Σ*-defining relations for* $\Pi \subseteq \Sigma$ is a set $\{r_x \mid x \in \Pi\}$ such that for each $x \in \Pi$, r_x is a Σ-defining relation for x.

Definition B.3 (Derived Generator Graph)**.** Let D be a family of Σ-defining relations for $\Pi \subseteq \Sigma$. The *derived generator graph for* D is the digraph $\Gamma_D(D) := (\Pi, E)$ such that $(x, y) \in E$ if and only if there exists $w, w' \in \Sigma^*$ such that $x \approx w \cdot y \cdot w'$ is a relation in D.

The proofs of Theorem B.6 and Theorem B.11 both rely heavily on derived generator graphs. Intuitively, a generator can be introduced (resp. eliminated) if it has no dependencies (resp. dependants) in the derived generator graph. It is always possible to find such a generator, provided the graph is acyclic. However, once a generator has been removed from the defining relations, it is important that the new derived generator graph is also acyclic. In fact, the new derived generator graph is always a subgraph of the previous derived generator graph, as outlined in Lemma B.4.

Lemma B.4. *If D is a family of Σ-defining relations and $D' \subseteq D$, then $\Gamma_D(D')$ is a subgraph of $\Gamma_D(D)$.*

Proof. If D is a family of Σ-defining relations for Π, then there exists $\Pi' \subseteq \Pi$ such that D' is a family of Σ-defining relations for Π'. Then $\Gamma_D(D) = (\Pi, E)$ and $\Gamma_D(D') = (\Pi', E')$ for some $E \subseteq \Pi \times \Pi$ and $E' \subseteq \Pi' \times \Pi$. Let $(x, y) \in E'$, Then there exists some $w, w' \in \Sigma^*$ such that $x \approx_{D'} w \cdot y \cdot w'$. Since $D' \subseteq D$, then $x \approx_D w \cdot y \cdot y'$. Then $(x, y) \in E$. Since (x, y) was arbitrary, then $E' \subseteq E$. Since $\Pi' \subseteq \Pi$ and $E' \subseteq E$, then $\Gamma_D(D')$ is a subgraph of $\Gamma_D(D)$. □

B.2.1 Introduction of Derived Generators

The goal of this section is to prove Theorem B.6. First, Lemma B.5 is introduced to prove that every finite set of defining relations with an acyclic graph must contain at least one defining relation $x \approx w$ whose right-hand side consists only of primitive generators. Since w is consists only of primitive generators, then it may be introduced by a **Gen**$(+)$ transformation. The proof then follows by induction on the number of defining relations, as outlined below.

Lemma B.5. *If D is a family of Σ-defining relations for Π with Π finite and $\Gamma_D(D)$ acyclic, then there exists a relation $x \approx w$ in D such that $w \in (\Sigma \setminus \Pi)^*$.*

Proof. Since Π is finite and $\Gamma_D(D)$ is acyclic, the by Lemma B.1, there exists some vertex $x \in \Pi$ such that x has no children in $\Gamma_D(D)$. Let $x \approx_D w$ be the Σ-defining relation for x in D with $n = |w|$. Let $k \in [n]$. Assume for the intent of contradiction that $w_k \in \Pi$. Then there exists some $w', w'' \in \Sigma^*$ such that $w = w' \cdot w_k \cdot w''$. Then w_k is a child of x by definition. However, x has no children by assumption. Then $w_k \notin \Pi$ by contradiction. Since k was arbitrary, then $w \in (\Sigma \setminus \Pi)^*$. □

Theorem B.6. *Let $\Sigma \subseteq \widehat{\Sigma}$ be an alphabet with $\Pi = \widehat{\Sigma} \setminus \Sigma$ finite and $R \subseteq \Sigma^* \times \Sigma^*$. If D is a family of $\widehat{\Sigma}$-defining relations for Π with $\Gamma_D(D)$ acyclic, then there exists a length $|\Pi|$ sequence of* ***Gen**$(+)$* *transformations between $\langle \Sigma \mid R \rangle$ and $\langle \widehat{\Sigma} \mid R \cup D \rangle$.*

Proof. The proof follows by induction on $|\Pi|$.

- **Base Case**. Assume that $|\Pi| = 0$. Then $|D| = 0$ and $\langle \Sigma \mid R \rangle = \langle \Sigma' \mid R \cup D \rangle$. Then there exists a length 0 sequence of **Gen**$(+)$ transformations between $\langle \Sigma \mid R \rangle$ and $\langle \widehat{\Sigma} \mid R \cup D \rangle$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|\Pi| = k$ and $\Pi_D(D)$ is acyclic, then there exists a length k sequence of **Gen**$(+)$ transformations between $\langle \Sigma \mid R \rangle$ and $\langle \widehat{\Sigma} \mid R \cup D \rangle$.

– **Inductive Step**. Assume that $|\Pi| = k+1$ and $\Gamma_D(D)$ is acylic. Since Π is finite and $\Gamma_D(D)$ is acyclic, then by Lemma B.5 there exists a $\widehat{\Sigma}$-defining relation $x \approx w$ in D. Let $\Lambda = \Sigma \cup \{x\}$ and $Q = R \cup \{x \approx w\}$. Then $D \setminus \{r_x\}$ is a family of $\widehat{\Sigma}$-defining relations for $\Pi \setminus \{x\}$ and $Q \subseteq \Lambda^* \times \Lambda^*$. Since $\Gamma_D(D \setminus \{r_x\})$ is a subgraph of $\Gamma_D(D)$ by Lemma B.4 with $\Gamma_D(D)$ acyclic, then $\Gamma_D(D \setminus \{r_x\})$ is also acyclic. Since $|\Pi \setminus \{x\}| = |\Pi| - 1 = k$ and $\Gamma_D(D \setminus \{r_x\})$ is acyclic, then by the inductive hypothesis, there exists a length k sequence of **Gen**$(+)$ transformations from $\langle \Lambda \mid Q \rangle$ to $\langle \widehat{\Sigma} \mid R \cup D \rangle$. Furthermore, $\langle \Sigma \mid R \rangle \cong \langle \Lambda \mid Q \rangle$ by **Gen**$(+)$. Then there exists a length $k+1$ sequence of **Gen**$(+)$ transformations between $\langle \Sigma \mid D \rangle$ and $\langle \widehat{\Sigma} \mid R \cup D \rangle$.

By the principle of induction, there exists a length $|\Pi|$ sequence of **Gen**$(+)$ transformations between $\langle \Sigma \mid D \rangle$ and $\langle \Sigma' \mid R \cup D \rangle$. □

B.2.2 Tietze Transformations to Exchange Relations

The goal of this section is to introduce a technique necessary to prove that derived generators can be eliminated via finite sequences of Tietze transformations. Using the Tietze transformations discussed so far, it is possible to remove and introduce redundant relation. In practice, one often wishes to replace a relation $q \approx w$ with a relation $q' \approx w'$, where neither $q \approx w$ nor $q' \approx w'$ is redundant without the other relation. The following lemma gives a sufficient condition for when $q \approx w$ can be exchanged with $q' \approx w'$, and provides an upper bound on the number of Tietze transformations required to carry out the exchange.

Lemma B.7. *Let $G = \langle \Sigma \mid R \rangle$ be a presentation with $q \approx w$ in R and $Q = R \setminus \{q \approx w\}$. If there exists $q' \in \Sigma^*$ and $w' \in \Sigma^*$ such that $q \sim_Q q'$ and $w \sim_Q w'$, then there exists a finite sequence of Tietze transformations between G and $\langle \Sigma \mid Q \cup \{q' \approx w'\} \rangle$ of length between 1 and 2.*

Proof. Let $S = R \cup \{q' \approx w'\}$ and $T = Q \cup \{q' \approx w'\}$. If $q' \approx_R w'$, then $\langle \Sigma \mid R \rangle = \langle \Sigma \mid S \rangle$. Assume instead that $q \not\approx_R w$. Since $Q \subseteq R$, then $q \sim_R q'$ and $w \sim_R w'$. Then $q' \sim_R q$ by the symmetry of $(\sim_R)$. Since $q' \sim_R q$, $q \sim_R w$, and $w \sim_R w'$, then $q' \sim_R w'$ by transitivity. Then $\langle \Sigma \mid R \rangle \cong \langle \Sigma \mid S \rangle$ by a single **Rel**$(+)$ transformation. In either case, there exists a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid S \rangle$ of length at most one. Since $Q \subseteq T$, then $q \sim_T q'$ and $w \sim_T w'$. Then $w' \sim_T w$ by the symmetry of $(\sim_T)$. Since $q \sim_T q'$, $q' \sim_T w'$, and $w' \sim_T w$, then $q \sim_T w$ by the transitivity of $(\sim_S)$. Since $S \neq T$, then $\langle \Sigma \mid S \rangle \cong \langle \Sigma \mid T \rangle$ by a single **Rel**$(+)$ transformation. In conclusion, there exists a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid S \rangle$ of length at between 1 and 2. □

B.2.3 Elimination of Derived Generators

The goal of this section is to prove Theorem B.11. The first step in this proof is to show that the set of relations can be replaced by $Q \cup D$ where Q is a set of relations over the primitive generators and D is the set of defining relations. This follows by induction in Lemma B.9, where Lemma B.8 is used to find new relations of the form, and then Lemma B.7 is used to exchange the relations. The second step in this proof is to show that the derived generators can be eliminated through a finite sequence of Tietze transformations. This follows by induction in Theorem B.11, where Lemma B.10 is used to find a derived generator upon which non other derived generator depends. Such a generator is necessarily redundant, and may be eliminated via a **Rel**$(-)$ transformation.

Lemma B.8. *Let $\Sigma \setminus \widehat{\Sigma}$ be an alphabet with $\Pi = \widehat{\Sigma} \setminus \Sigma$. If D is a family of $\widehat{\Sigma}$-defining relations for Π with $\Gamma_D(D)$ acyclic, then for each $w \in \Sigma^*$, there exists some $w' \in (\Sigma')^*$ such that $w \sim_D w'$.*

Proof. The proof follows by induction on the size of Π.

- **Base Case**. Assume that $|\Pi| = 0$ and $w \in \widehat{\Sigma}^*$. Then $\Pi = \varnothing$ and $w \in \Sigma^*$. Since $w \sim_D w$ by the symmetry of $(\sim_D)$, then there exists some $w' \in \Sigma^*$ such that $w \sim_D w'$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|\Pi| = k$, $\Gamma_D(D)$ is acyclic, and $w \in \widehat{\Sigma}^*$, then there exists some $w' \in \Sigma^*$ such that $w \sim_D w'$.
- **Inductive Step**. Assume that $|\Pi| = k+1$ and $\Gamma_D(D)$ is acyclic. Since Π is finite and $\Gamma_D(D)$ is acyclic, then by Lemma B.10, there exists a $\widehat{\Sigma}$-defining relation $x \approx q$ in D such that $D \setminus \{r_x\}$ is a family of $(\widehat{\Sigma} \setminus \{x\})$-defining relations for $\Pi \setminus \{x\}$. Then by Lemma A.2, there exists some $w' \in (\widehat{\Sigma} \setminus \{x\})^*$ such that $w \sim_D w'$. Since $\Gamma_D(D \setminus \{r_x\})$ is a subgraph of $\Gamma_D(D)$ by Lemma B.4 with $\Gamma_D(D)$ acyclic, then $\Gamma_D(D \setminus \{r_x\})$ is also acyclic. Then $D \setminus \{r_x\}$ is a family of $(\widehat{\Sigma} \setminus \{x\})$-defining relations for $\Pi \setminus \{x\}$ with $\Gamma_D(D)$ acyclic and $w' \in (\widehat{\Sigma} \setminus \{x\})^*$. Since $|\Pi \setminus \{x\}| = |\Pi| - 1 = k$, then by the inductive hypothesis, there exists some $w'' \in \Sigma^*$ such that $w' \sim_{D'} w''$. Since $D' \subseteq D$, then $w' \sim_D w''$. Since $\sim_D$ is transitive, then $w \sim_D w''$. Then there exists a $w'' \in \Sigma^*$ such that $w \sim_D w''$.

Since $\Gamma_D(D)$ is acyclic, then by the principle of induction, there exists a $w' \in \Sigma^*$ such that $w \sim_D w'$. □

Lemma B.9. *Let $\Sigma \subseteq \widehat{\Sigma}$ be an alphabet with $\Pi = \widehat{\Sigma} \setminus \Sigma$ and $R \subseteq \widehat{\Sigma}^* \times \widehat{\Sigma}^*$ finite. If $D \subseteq R$ is a family of $\widehat{\Sigma}$-defining relations for Π with $\Gamma_D(D)$ acyclic, then there exists a $Q \subseteq \Sigma^* \times \Sigma^*$ with $|Q| \leq |R \setminus D|$ and a finite sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \Sigma \mid Q \cup D \rangle$ of length between k and $2k$ where $k = |R| - |R \cap (\Sigma^* \times \Sigma^*)| - |D|$.*

Proof. Let $S = R \cap (\Sigma^* \times \Sigma^*)$ and $\overline{S} = R \setminus (S \sqcup D)$. Then $R = S \sqcup D \sqcup \overline{S}$. The proof follows by induction on $|\overline{S}|$ in this decomposition.

- **Base Case**. Assume that R decomposes into $S \sqcup D \sqcup \overline{S}$ with $|\overline{S}| = 0$. Then $\langle \widehat{\Sigma} \mid R \rangle = \langle \widehat{\Sigma} \mid S \cup D \rangle$. Then there exists a length 0 sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \widehat{\Sigma} \mid S \cup D \rangle$. Clearly $|S| = |R \setminus D|$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if R decomposes as $S \sqcup D \sqcup \overline{S}$ with $|\overline{S}| = k$, then there exists some $Q \subseteq \Sigma^* \times \Sigma^*$ with $|Q| \leq |R \setminus D|$ and a sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \widehat{\Sigma} \mid Q \cup D \rangle$ of length at most $2k$.
- **Inductive Step**. Assume that R decomposes into $S \sqcup D \sqcup \overline{S}$ with $|\overline{S}| = k+1$. Let $q \approx w$ in $\overline{S}$. Then by Lemma B.8, there exists $q' \in \Sigma^*$ and $r' \in \Sigma^*$ such that $q \sim_D q'$ and $r \sim_D r'$. Then by Lemma B.7, there exists a finite sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \widehat{\Sigma} \mid R' \rangle$ of length between 1 and 2, where $R' = R \cup \{q' \approx w'\} \setminus \{q \approx w\}$. Then R' decomposes as $S' \sqcup D \sqcup \overline{S}'$ where $S' = S \cup \{q' \approx r'\}$ and $\overline{S}' = \overline{S} \setminus \{q \approx r\}$. Since $q \approx_S r$, then $|\overline{S}'| = |\overline{S}| - 1 = k$. Then the inductive hypothesis holds, and there exists some $Q \subseteq \Sigma^* \times \Sigma^*$ with $|Q| \leq |R' \setminus D|$ and a sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R' \rangle$ and $\langle \widehat{\Sigma} \mid Q \cup D \rangle$ of length between k and $2k$. Then there exists a sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \widehat{\Sigma} \mid Q \cup D \rangle$ of length between $k+1$ and $2(k+1)$. Since $|S'| \leq |S|$ and $|\overline{S}'| = |\overline{S}|$, then $|Q| \leq |S'| + |\overline{S}'| \leq |S| + |\overline{S}| = |R \setminus D|$.

By the principle of induction, there exists a $Q \subseteq \Sigma^* \times \Sigma^*$ with $|Q| \leq |R \setminus D|$ and a sequence of Tietze transformations between $\langle \widehat{\Sigma} \mid R \rangle$ and $\langle \widehat{\Sigma} \mid Q \cup D \rangle$ of length between $|\overline{S}|$ and $2|\overline{S}|$. □

Lemma B.10. *If D is a family of Σ-defining relations for Π with Π finite and $\Gamma_D(D)$ acyclic, then there exists a relation $x \approx w$ in D such that $D \setminus \{x \approx w\}$ is a family of $(\Sigma \setminus \{x\})$-defining relations for $\Pi \setminus \{x\}$.*

Proof. Since Π is finite and $\Gamma_D(D)$ is acyclic, then by Lemma B.1, there exists some vertex $x \in \Pi$ such that x has no parents in $\Gamma_D(D)$. Since x is a vertex in $\Gamma_D(D)$, then there exists a Σ-defining relation $x \approx w$ in D. Let $D' = D \setminus \{x \approx w\}$. Let $y \approx q$ be a Σ-defining relation in D' with $n = |q|$. Let $k \in [n]$. Assume for the intent of contradiction that $q_k = x$. Then there exists some $q', q'' \in \Sigma^*$ such that $q = q' \cdot x \cdot q''$. Then x is a child of y by definition. However, x has no parents by assumption. Then $q_k \neq x$ by contradiction. Since k was arbitrary, then $q \in (\Sigma \setminus \{x\})^*$. Since $y \approx q$ was arbitrary, then D' is a family of $(\Sigma \setminus \{x\})$-defining relations for $\Pi \setminus \{x\}$. □

Theorem B.11. *Let $\Sigma' \subseteq \Sigma$ be an alphabet with $\Pi = \Sigma \setminus \Sigma'$ finite and $R \subseteq \Sigma^* \times \Sigma^*$ finite. If $D \subseteq R$ is a family of defining relations for Π with $\Gamma_D(D)$ acyclic, then there exists a $Q \subseteq (\Sigma')^* \times (\Sigma')^*$ with $|Q| \leq |R \setminus D|$ and a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma' \mid Q \rangle$ of length between $n+k$ and $2n+k$ where $n = |R| - |R \cap ((\Sigma')^* \times (\Sigma')^*)| - |D|$ and $k = |\Pi|$.*

Proof. By Lemma B.9, there exists a $Q \subseteq (\Sigma')^* \times (\Sigma')^*$ with $|Q| \leq |R \setminus D|$ and a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid Q \cup D \rangle$ of length between n and $2n$. The proof follows by induction on $|\Pi|$.

– **Base Case**. If $|\Pi| = 0$, then $|D| = 0$. Then $\langle \Sigma \mid Q \cup D \rangle = \langle \Sigma' \mid Q \rangle$. Then there exists a length 0 sequence of Tietze transformations between $\langle \Sigma \mid Q \cup D \rangle$ and $\langle \Sigma' \mid Q \rangle$.

– **Inductive Case**. Assume that for some $k \in \mathbb{N}$, if $|\Pi| = k$ and $\Pi_D(D)$ is acyclic, then there exists a length k sequence of Tietze transformations between $\langle \Sigma \mid Q \cup D \rangle$ and $\langle \Sigma' \mid Q \rangle$.

– **Inductive Step**. Assume that $|\Pi| = k+1$ and $\Gamma_D(D)$ is acyclic. Since Π is finite and $\Gamma_D(D)$ is acyclic, then by Lemma B.1 there exists some vertex $x \in \Pi$ such that x has no parents in $\Gamma_D(D)$. Define $\Lambda = \Sigma \setminus \{x\}$ and $D' = D \setminus \{r_x\}$. Let $y \approx_{D'} w$ and assume for the intent of contradiction that $w \notin \Lambda^*$. Then there exists $w', w'' \in \Sigma^*$ such that $w = w' \cdot x \cdot w''$. Then x is a child of y in $\Gamma_D(D)$. However, x has no parents by assumption. Then $w \in \Lambda^*$ by contradiction. Since $y \approx_{D'} w$ was arbitrary, then $D' \subseteq \Lambda^* \times \Lambda^*$. Then $Q \cup D' \subseteq \Lambda^* \times \Lambda^*$. It follows that $\langle \Sigma \mid Q \cup D \rangle \cong \langle \Lambda \mid Q \cup D' \rangle$ by **Gen**$(-)$. Furthermore, D' is a family of defining relations for $\Pi \setminus \{x\}$. Since $|\Pi \setminus \{x\}| = k$ and $\Gamma_D(D')$ is acyclic by Lemma B.4, then there exists a length k sequence of Tietze transformations between $\langle \Lambda \mid Q \cup D' \rangle$ and $\langle \Sigma' \mid Q \rangle$ by the inductive hypothesis. Then there exists a length $k+1$ sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma' \mid Q \rangle$.

Then by the principle of induction, there exists a length k sequence of Tietze transformations between $\langle \Sigma \mid Q \cup D \rangle$ and $\langle \Sigma' \mid Q \rangle$. Then there exists a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma' \mid Q \rangle$ of length between $n+k$ and $2n+k$. □

B.2.4 The Derived Generator Graph for $W(E_8)$

The derived generators for $W(E_8)$ are $\Pi := \Sigma_D \setminus \{X_0, CX_{0,1}, CCX_{1,2}, K_{1,2}\}$. The defining relations $D \subseteq R_0$ for $\Pi \subseteq \Sigma_D$ are given by Relation (1) through to Relation (19) in Figure 4. An illustration of the derived generator graph $\Gamma_D(D)$ can be found in Figure 9. Since this graph is acyclic, then Theorem B.6 and Theorem B.11 apply. The derived generator graphs for $O(n, \mathbb{D})$ have paths of length at most one, and are therefore trivially acyclic.

B.3 Derivational Proofs and Tietze Transformation

Assume that $G \cong \langle \Sigma \mid R \rangle$ with semantic interpretation $[\![\cdot]\!]_G$. During proofs based on Tietze transformations, it is often necessary to find a sequence of **Rel**$(-)$ and **Rel**$(+)$ transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid Q \rangle$. For example, this case arises in Section 4, where $R = R_{E8} \cup R_{D(E8)}$ and $Q = R_{E8(D)} \cup R_D$. If $[\![\cdot]\!]_G$ induces an isomorphism and every relation $w \approx_Q w'$ satisfies $[\![w]\!]^* = [\![w']\!]$, then using Theorem A.6 there exists a sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid R \rangle$. Eliminating the relations in R

requires more care. For example, if $r \in R$ is not derivable from $(R \cup Q) \setminus \{r\}$, then $\langle \Sigma \mid R \cup Q \rangle$ is a proper quotient of $\langle \Sigma \mid Q \rangle$. Instead, it must be shown that for each $w \approx_R w'$, it follows that $w \sim_Q w'$. One way to approach this problem is to first derive some auxiliary relations A from Q, and then use $Q \cup A$ to derive R. However, transforming these derivations into a sequence of Tietze transformations is often tedious, and not well-aligned with the process of proof discovery. On the other hand, if the derivations are not transformed into a valid sequence of Tietze transformations, then it is possible to obtain invalid proofs, such as those with cyclic derivations (see Example B.16).

This section formalizes the ad-hoc proof technique described above,and identifies sufficient conditions for when such a family of derivations induces a valid sequence of Tietze transformations between $\langle \Sigma \mid R \rangle$ and $\langle \Sigma \mid Q \rangle$. In the following definitions, $L(\Sigma) = \mathbb{N} \times (\Sigma^* \times \Sigma^*)$ will represent a set of indexed relations over Σ. For example, let $(n,(w,w')) \in L(\Sigma)$. The index n in $(n,(w,w'))$ indicates that $(n,(w,w'))$ is a derived relation, and allows for multiple derivations of the same relation. More concretely, if $n = 3$, $w = a \cdot b$, and $w' = x \cdot y \cdot z$, then $(n,(w,w'))$ is the third derivation that yields $a \cdot b \approx x \cdot y \cdot z$.

Definition B.12 (Derivational Proof)**.** A *derivational proof in* $\langle \Sigma \mid R \rangle$ is a subset $P \subseteq L(\Sigma) \times (L(\Sigma) \cup R)^*$ which satisfies the following conditions.

- **Indexed**. For all $(\ell,d) \in P$ and $(\ell',d') \in P$ distinct, $\ell \neq \ell'$.
- **Well-founded**. For all $(\ell,d) \in P$ and $k \in \{1,2,\dots,|d|\}$, either $d_k \in R$ or $d_k \in L(\Sigma)$ and there exists some $d' \in (L(\Sigma) \cup R)^*$ such that $(d_k,d') \in P$.
- **Valid**. For all $(\ell,d) \in P$ with $(n,(w,w')) = \ell$ and $m = |d|$, there exists some $v \in (\Sigma^*)^{m+1}$ such that $v_1 = w$, $v_{m+1} = w'$, and for all $k \in [m]$ either $d_k \in R$ and $v_k \xrightarrow{d_k} v_{k+1}$ or $(n',r) = d_k$ and $v_k \xrightarrow{r} v_{k+1}$.

A set $Q \subseteq \Sigma^* \times \Sigma^*$ is *entailed by* P, written $P \models Q$, if $Q \subseteq R \cup \{r \mid ((n,r),d) \in P\}$.

Definition B.13 (Proof Substitution)**.** Let P be a proof in $\langle \Sigma \mid R \rangle$. If $(\ell,d) \in P$, $d \in (L(\Sigma) \cup R)^*$, and $P' = (P \setminus \{(\ell,d)\}) \cup \{(\ell,d')\}$ is a proof for $\langle \Sigma \mid R \rangle$, then we say that P' is a *substitution of* P *by* d' *at* ℓ, written $P[\ell \mapsto d']$.

Definition B.14 (Derivation Graph)**.** Let P be a proof in $\langle \Sigma \mid R \rangle$. The *derivation graph for* P is the digraph $\Gamma_D(P) = (V,E)$ such that $V = \{\ell \mid (\ell,d) \in P\}$ and $(\ell,\ell') \in E$ if and only if there exists $(\ell,d) \in P$ and $k \in \{1,2,\dots,|d|\}$ such that $d_k = \ell'$.

Example B.15 (Derivations and Substitutions)**.** Consider $G = \langle x,y \mid x^2 \approx \varepsilon, y^2 \approx \varepsilon \rangle$. It is not hard to show $x \cdot y \cdot x \cdot y^2 \cdot x \cdot y \cdot x \sim \varepsilon$. However, it helps to first prove that $x \cdot y^2 \cdot x \sim \varepsilon$. This can be written as a derivation proof. There will be two derivations, with labels $\ell = (0,(x \cdot y^2 \cdot x, \varepsilon))$ and $\ell' = (0,(x \cdot y \cdot x \cdot y^2 \cdot x \cdot y \cdot x, \varepsilon))$ respectively. Associated with ℓ and ℓ' are two derivations d and d', defined as follows. Let $r = (x^2,\varepsilon)$ and $r' = (y^2,\varepsilon)$.

$$(d): \ x \cdot y \cdot y \cdot x \xrightarrow{r'} x \cdot x \xrightarrow{r} \varepsilon$$

$$(d'): \ x \cdot y \cdot x \cdot y \cdot y \cdot x \cdot y \cdot x \xrightarrow{\ell} x \cdot y \cdot y \cdot x \xrightarrow{\ell} \varepsilon$$

These pieces can be assembled into a derivational proof $P = \{(\ell,d),(\ell',d')\}$. This proof is indexed, since ℓ and ℓ' each appear exactly once as labels. This proof is well-formed, since r, r', d, and d' are all in-scope. The proof is valid, since each step of each derivation follows. Of course, it is possible to expand out d' using the steps of d. This yields a new derivation d'' defined as follows.

$$(d''): \ x \cdot y \cdot x \cdot y \cdot y \cdot x \cdot y \cdot x \xrightarrow{r'} x \cdot y \cdot x \cdot x \cdot y \cdot x \xrightarrow{r} x \cdot y \cdot y \cdot x \xrightarrow{r'} x \cdot x \xrightarrow{r} \varepsilon$$

This new derivational proof corresponds to the substitution $P[\ell' \mapsto d'']$. In this new proof, the derivation of ℓ' no longer relies on the lemma ℓ. Later in this section, lemma eliminating substitutions will be used to extract Tietze transformations from derivational proofs with acyclic derivation graphs. □

Example B.16 (Cyclic Derivations)**.** Consider $G = \langle x, y \mid x^2 \approx \varepsilon, y^2 \approx \varepsilon \rangle$. It is not hard to show that $G \cong \mathbb{Z}_2 \star \mathbb{Z}_2$, where $(\star)$ denotes the free product of groups. It follows that G is non-abelian. In particular, $\pi_G(x \cdot y) \neq \pi_G(y \cdot x)$. Now, consider the proof $P = \{((0,(x \cdot y, y \cdot x)), d), ((0,(x \cdot y^3, y \cdot x \cdot y)), d')\}$, where d and d' are defined as follows. Let $r = (y^2, \varepsilon)$, $\ell = (0,(x \cdot y, y \cdot x))$ and $\ell' = (0,(x \cdot y^3, y \cdot x \cdot y^2))$.

$$(d): \; x \cdot y \xleftarrow{r} x \cdot y^3 \xrightarrow{\ell'} y \cdot x \cdot y^2 \xrightarrow{r} y \cdot x$$

$$(d'): \; x \cdot y^3 \xrightarrow{\ell} y \cdot x \cdot y^2$$

This proof is indexed, well-formed, and valid. However, these derivations suggest that $x \cdot y \sim y \cdot x$. The problem in this proof is that d depends on d' and d' depends on d. In other words, the proof is self-referential. It will be shown later in this section that if $\Gamma_D(P)$ is acyclic, then P is not self-referential. This motivates the requirement that $\Gamma_D(P)$ is acyclic throughout the rest of this section. □

B.3.1 Substitutions and Derivation Graphs

This section relates derivational proofs and substitutions to the derivation graphs they induce. When a derivation (ℓ, d) depends on a derivation (ℓ', d'), we say that ℓ' is a lemma for ℓ. In Lemma B.17, lemma-free derivations are characterized by their vertices in a derivation graph. Similarly, Lemma B.18 characterizes lemma-free proofs in terms of their derivation graphs. As expected, Lemma B.19 shows that if a substitution only eliminates the use of lemmas, such as in Example B.15, then the resulting derivation graph is a subgraph of the original graph. To this end, Lemma B.20 provides sufficient conditions for a valid substitution. Together, these four lemmas give a graph-theoretic characterization of the lemma substitution in Example B.15.

Lemma B.17. *Let P be a proof for $\langle \Sigma \mid R \rangle$. If $(\ell, d) \in P$ and ℓ has no children in $\Gamma_D(P)$, then $d \in R^*$.*

Proof. Let $k \in \{1, 2, \ldots, |d|\}$. Assume for the intent of contradiction that $d_k \notin R^*$. Then (ℓ, d_k) is an edge in $\Gamma_D(P)$. However, ℓ has no children in $\Gamma_D(P)$. Then $d_k \in R$. Since k was arbitrary, then $d \in R^*$. □

Lemma B.18. *If P is a proof for $\langle \Sigma \mid R \rangle$, then $\Gamma_D(P)$ is edgeless if and only if $d \in R^*$ for all $(\ell, d) \in P$.*

Proof. Let $\Gamma_D(P) = (V, E)$. Consider the contrapositive statement. Then there exists an $(\ell, \ell') \in E$. This is true if and only if there exists a $(\ell, d) \in P$ and $k \in \{1, 2, \ldots, |d|\}$ such that $d_k = \ell'$. This is true if and only if there exists an $(\ell, d) \in P$ such that $d \notin R^*$. □

Lemma B.19. *Let P be a proof for $\langle \Sigma \mid R \rangle$ and $(V, E) = \Gamma_D(P)$. If $(\ell, d) \in P$ and there exists some $d' \in R^*$ such that $P' = P[l \mapsto d']$ is also a proof for $\langle \Sigma \mid R \rangle$, then $\Gamma_D(P') = (V, E_\ell)$ where $E_\ell = \{(\ell', \ell'') \in E \mid \ell \neq \ell'\}$.*

Proof. Let $\Gamma_D(P') = (V', E')$.

- **Vertices** $(\subseteq)$. Let $\ell' \in V'$. Then there exists some $\delta \in (L(\Sigma) \cup R)^*$ such that $(\ell', \delta) \in P'$. Then either $(\ell', \delta) \in P$ or $(\ell', \delta) = (\ell, d')$. If $(\ell', \delta) \in P$, then $\ell \in V$. If $(\ell', \delta) = (\ell, d')$, then $\ell' \in V$ since $(\ell, d) \in P$. In either case $\ell' \in V$. Since ℓ' was arbitrary, then $V' \subseteq V$.
- **Vertices** $(\supseteq)$. Let $\ell' \in V$. Then there exists some $\delta \in (L(\Sigma) \cup R)^*$ such that $(\ell', \delta) \in P$. Then either $\ell' = \ell$ or $\ell' \neq \ell$. If $\ell' = \ell$, then $\ell' \in V'$ since $(\ell', d') \in P'$. If $\ell' \neq \ell$, then $\ell' \in V'$ since $(\ell', \delta) \in P \setminus \{(\ell, d)\}$. In either case, $\ell' \in V$, Since ℓ' was arbitrary, then $V' \subseteq V$.

- **Edges** ($\subseteq$). Let $(\ell', \ell'') \in E'$. Then there exists some $\delta \in (L(\Sigma) \cup R)^*$ and $k \in \{1, 2, \ldots, |\delta|\}$ such that $(\ell', \delta) \in P'$ and $\delta_k = \ell''$. Then $\delta \notin R^*$. Consequently, $\delta \neq d'$. Then $\ell \neq \ell'$, since P' is indexed. Then $(\ell', \delta) \in P$. Consequently, $(\ell', \ell'') \in E$. Since $\ell' \neq \ell$, then $(\ell', \ell'') \in E_\ell$. Since (ℓ', ℓ'') was arbitrary, then $E' \subseteq E_\ell$.
- **Edges** ($\supseteq$). Assume that $(\ell', \ell'') \in E_\ell$. Then $\ell' \neq \ell$ and $(\ell', \ell'') \in E$. Then there exists some $\delta \in (L(\Sigma) \cup R)^*$ and $k \in \{1, 2, \ldots, |\delta|\}$ such that $(\ell', \delta) \in P$ and $\delta_k = v$. Since $\ell' \neq \ell$, then $(\ell', \delta) \in P'$. Then $(\ell', \ell'') \in E'$. Since (ℓ', ℓ'') was arbitrary, then $E_\ell \subseteq E'$.

Then $\Gamma_D(P') = (V, E_\ell)$. □

Lemma B.20. *Let P be a proof for $\langle \Sigma \mid R \rangle$ and $V = \{\ell \mid (\ell, d) \in P\}$. If $(\ell, d) \in P$, $d' \in (V \cup R)^*$, and (ℓ, d') is valid, then $P[\ell \mapsto d']$ is a proof for $\langle \Sigma \mid R \rangle$.*

Proof. Let $P' = [\ell \mapsto d']$. It must be shown that P' is indexed, well-formed, and valid.

- **Indexed**. Let $(\ell', \delta) \in P'$ and $(\ell'', \delta') \in P'$ distinct. Without loss of generality, assume $\ell' \neq \ell$. Then $(\ell', \delta) \in P$. Now, these are two cases to consider, depending on whether $\ell'' = \ell$. If $\ell'' = \ell$, then $(\ell'', d) \in P$ and $\ell' \neq \ell''$ since P is indexed. If $\ell'' \neq \ell$, then $(\ell'', \delta') \in P$ and $\ell' \neq \ell''$ since P is indexed. In either case $\ell' \neq \ell'$. Since (ℓ', δ) and (ℓ'', δ') were arbitrary, then P' is indexed.
- **Well-formed**. Let $(\ell', \delta) \in P'$. There are two cases to consider, depending on whether $\ell' = \ell$.
  - If $\ell' = \ell$, then $\delta = d'$, since P' is indexed. Then $\delta \in (V \cup R)^*$. Then for all $k \in \{1, 2, \ldots, |\delta|\}$, either $\delta_k \in R$ or $\delta_k \in V$. If $\delta_k \in V$ and $\delta_k = \ell$, then $(\delta_k, d') \in P'$. If $\delta_k \in V$ and $\delta_k \neq \ell$, then there exists some $\delta' \in (L(\Sigma) \cup R)^*$ such that $(d_k, \delta') \in P$. Since $\delta_k \neq \ell$, then $(\delta_k, \delta') \in P'$. In either case, if $\delta_k \in V$, then there exists some $\delta' \in (L(\Sigma) \cup R)^*$ such that $(d_k, d'') \in P'$ Since k was arbitrary, then (ℓ', δ) is well-formed.
  - If $\ell' \neq \ell$, then $(\ell', \delta) \in P$. Let $k \in \{1, 2, \ldots, |\delta|\}$. Since P is well-formed, then either $\delta_k \in R$ or there exists some $\delta' \in (L(\Sigma) \cup R)^*$ such that $(\delta_k, \delta') \in P$. There are two cases to consider, depending on whether $\delta_k = \ell$. If $\delta_k = \ell$, then $(\ell, d') \in P'$. If $\delta_k \neq \ell$, then $(\delta_k, \delta') \in P'$. In either case, there exists some $\delta' \in (L(\Sigma) \cup R)^*$ such that $(\delta_k, \delta') \in P'$. Since k was arbitrary, then (ℓ', δ) is well-formed.

  In either case, (ℓ', δ) is well-formed. Since (ℓ', δ) was arbitrary, then P' is well-formed.
- **Valid**. Let $(\ell', \delta) \in P'$. Now these are two cases to consider, depending on whether $\ell' = \ell$. If $\ell' = \ell$, then $\delta = d'$ since P' is indexed, and consequently δ is valid by assumption. If $\ell' \neq \ell$, then $(\ell', \delta) \in P$ and δ is valid by the validity of P. In either case, (ℓ', δ) is valid. Since (ℓ', δ) was arbitrary, then P' is valid.

Then P' is a proof for $\langle \Sigma \mid R \rangle$. □

B.3.2 From Derivational Proofs to Tietze Transformations

Let $\langle \Sigma \mid R \rangle$ be a monoid presentation. The goal of this section is to prove Theorem B.25, which states that acyclic derivational proofs are sound for the isomorphism of finite monoid presentations. The completeness of acyclic derivational proofs follows immediately from the fact that every statement of the form $w \sim_R w'$ corresponds to at least one finite derivation. The proof proceeds as follows. First, Lemma B.21 shows that the lemma substitutions outlined in Example B.15 preserve the structure of derivational proofs. This is used in Lemma B.22 to show that if a derivation (ℓ, d) in a proof P depends on a lemma (ℓ', d') which follows directly from R, then d can be rewritten so that all dependencies on ℓ'

are removed without introducing any new dependencies. This is used repeated in Lemma B.23 to show that any derivation (ℓ, d) which depends only on lemmas which follow directly from R, can be rewritten to also follow directly from R. This is extended in Lemma B.24 to show that any finite and acyclic derivational proof P can be written into a proof P' such that every derivation depends only on R. Then each derivation in P' follows from $(\sim_R)$, and Theorem B.25 follows immediately by induction.

Lemma B.21. *Let P be a proof for $\langle \Sigma \mid R \rangle$. If (ℓ, d) and (ℓ', d') are derivations in P and there exists $\delta, \delta' \in (L(\Sigma) \cup R)^*$ such that $d = \delta \cdot \ell' \cdot \delta'$, then $P[\ell \mapsto \delta \cdot d' \cdot \delta']$ is a proof for $\langle \Sigma \mid R \rangle$.*

Proof. Let $L_P = \{\ell \mid (\ell, d) \in P\}$, $d'' = \delta \cdot d' \cdot \delta'$, and $(x, (q, r)) = \ell$. Since $\delta, d', \delta' \in (L_p \cup R)^*$, then $d'' \in (L_p \cup R)^*$. Since P is valid, then there exists $v \in (\Sigma^*)^{n+1}$ such that $v_1 = q$, $v_{n+1} = r$, and $v_k \xrightarrow{d_k} v_{k+1}$ for all $k \in [n]$, where $n = |d|$. Let $m = |\delta|$. Since P is valid and $v_m \xrightarrow{\ell'} v_{m+1}$, then there exists some $u \in (\Sigma^*)^{s+1}$ such that $u_1 = v_m$, $u_{s+1} = v_{m+1}$, and $u_k \xrightarrow{d'_k} u_{k+1}$ for all $k \in [s]$, where $s = |d'|$. Define $v' = (v_1, \ldots, v_m, u_1, \ldots, u_s, v_{m+1}, \ldots, v_n)$. Let $k \in \{1, 2, \ldots, |d''|\}$. There are five cases to consider.

- If $k < m$, then $v'_k = v_k$, $v'_{k+1} = v_{k+1}$, and $d''_k = \delta_k = d_k$. Then $v'_k \xrightarrow{d''_k} v'_{k+1}$.
- If $k = m$, then $v'_k = v_m$, $v'_{k+1} = u_1 = v_{m+1}$, and $d''_k = \delta_k$. Then $v'_k \xrightarrow{d''_k} v'_{k+1}$.
- If $m < k < m + s$, then $v'_k = u_{k-m}$, $v'_{k+1} = u_{k-m+1}$, and $d''_k = d'_{k-m}$. Then $v'_k \xrightarrow{d''_k} v'_{k+1}$.
- If $k = m + s$, then $v'_k = u_s$, $v'_{k+1} = v_{m+1} = u_{s+1}$, and $d''_k = d'_s$. Then $v'_k \xrightarrow{d''_k} v'_{k+1}$.
- If $k > m + s$, then $v'_k = u_{k-s}$, $v'_{k+1} = u_{k+1-s}$, and $d''_k = \delta'_{k-m-s}$. Then $v'_k \xrightarrow{d''_k} v'_{k+1}$.

In each case, $v'_k \xrightarrow{d''_k} v'_{k+1}$. Since k was arbitrary, then (ℓ, d'') is valid. Then $P[\ell \mapsto d'']$ is a proof by Lemma B.20. □

Lemma B.22. *Let P be a proof for $\langle \Sigma \mid R \rangle$. If (ℓ, ℓ') is a maximal path rooted at ℓ in $\Gamma_D(P)$, then there exists a $\hat{d} \in (L(\Sigma) \cup R)^*$ such that $P' = P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$ and ℓ has one less child in $\Gamma_D(P')$.*

Proof. Let $f : (L(\Sigma) \cup R)^* \to \mathbb{N}$ count the number of occurrences of ℓ' in a derivation. Given a proof Q for $\langle \Sigma \mid R \rangle$, let $C_Q : L(\Sigma) \to \mathscr{P}(L)$ map each $\ell \in L(\Sigma)$ to its children in $\Gamma_D(Q)$. Since (ℓ, ℓ') is a maximal path, then ℓ' has no children in $\Gamma_D(P)$. The proof follows by induction on the number of occurrences of ℓ' in the derivation.

- **Base Case**. Let $\hat{d} \in (L(\Sigma) \cup R)^*$ with $P' = P[\ell \mapsto \hat{d}]$ a proof for $\langle \Sigma \mid R \rangle$ and $C_P(\ell) = C_{P'}(\ell) \cup \{\ell'\}$. Assume that $f(\hat{d}) = 0$. Then $\hat{d}_j \neq \ell'$ for all $k \in [n]$, where $n = |\hat{d}|$. Then $(\ell, \ell') \notin \Gamma_D(P')$. Then $\ell' \notin C_{P'}(\ell)$. Then $C_{P'}(\ell) = C_P(\ell) \setminus \{\ell'\}$. Since $\ell' \in C_P(\ell)$, then $|C_{P'}(\ell)| = |C_P(\ell)| - 1$.
- **Inductive Hypothesis**. Let $d' \in (L(\Sigma) \cup R)^*$ such that $P' = P[\ell \mapsto d']$ a proof for $\langle \Sigma \mid R \rangle$ and $C_P(\ell) = C_{P'}(\ell) \cup \{\ell'\}$. Assume that for some $k \in \mathbb{N}$, if $f(d') = k$, then exists a $\hat{d} \in (L(\Sigma) \cup R)^*$ such that $P'' = P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$ and $|C_{P''}(\ell)| = |C_P(\ell)| - 1$.
- **Inductive Step**. Under the conditions of the inductive hypothesis, assume that $f(d') = k + 1$. Then there exists some $\delta, \delta' \in (L(\Sigma) \cup R)^*$ such that $d' = \delta \cdot \ell' \cdot \delta'$. Since ℓ' is a vertex in $\Gamma_D(P)$, then there exists some $d'' \in (L(\Sigma) \cup R)^*$ such that $(\ell, d'') \in P$. Define $\hat{d} = \delta \cdot d'' \cdot \delta'$. By Lemma B.21, $P'' = P[l \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$. Let $\ell'' \in C_P(\ell) \setminus \{\ell'\}$. Then there exists some $k \in \{1, 2, \ldots, |\delta|\}$ such that $d_k = \ell''$. Since $\ell'' \neq \ell$, then without loss of generality $k \leq |\delta|$ and $\hat{d}_k = \delta_k = \ell''$. Then $\ell'' \in C_{P''}(\ell)$. Since ℓ'' was arbitrary, then $C_P(\ell) \setminus \{\ell'\} \subseteq C_{P''}(\ell)$. Next, let $\ell'' \in C_{P''}(\ell)$. Then there

exists some $k \in \{1,2,\dots,|\hat{d}|\}$ such that $\hat{d}_k = \ell''$. Since $d \in (R^*)$, then without loss of generality $k \leq |\delta|$ and $\delta_k = \hat{h}_k = \ell''$. Then $\ell'' \in C_P(\ell)$. Since ℓ'' was arbitrary, then $C_{P''}(\ell) \subseteq C_P(\ell)$. Then $C_P(\ell) = C_{P''}(\ell) \cup \{\ell'\}$. Since $k+1 = f(d') = f(\delta) + 1 + f(\delta')$, then $f(\hat{d}) = f(\delta) + f(\delta') = k$. Then by the inductive hypothesis, there exists some $\hat{d}' \in (L(\Sigma) \cup R)^*$ such that $Q = P'[\ell \mapsto \hat{d}']$ is a proof for $\langle \Sigma \mid R \rangle$ and $|C_Q(\ell)| = |C_{P'}(\ell)| - 1 = |C_P(\ell)| - 1$. Since $Q = P[\ell \mapsto \hat{d}']$ by definition, then the inductive step holds.

It follows by definition that $d \in (L(\Sigma) \cup R)^*$ and $P = P[\ell \mapsto d]$. Then by the principle of induction, there exists a $\hat{d} \in (L(\Sigma) \cup R)^*$ such that $P' = P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$ and $|C_{P''}(\ell)| = |C_P(\ell)| - 1$. □

Lemma B.23. *Let P be a proof for $\langle \Sigma \mid R \rangle$. If $(\ell, d) \in P$ and all paths rooted at ℓ in $\Gamma_D(P)$ have length at most one, then there exists a $\hat{d} \in R^*$ such that $P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$.*

Proof. Let $\Gamma_D(P) = (V, E)$. Since $|V| = |P|$ and all paths rooted at ℓ have length at most one, then the number of paths rooted at ℓ in $\Gamma_D(P)$ is finite. The proof follows by induction on the number of paths rooted at ℓ in $\Gamma_D(P)$.

- **Base Case**. Assume that $\Gamma_D(P)$ has zero paths rooted at ℓ. Then $d \in R^*$ by Lemma B.17. Then $P = P[l \mapsto d]$ with $d \in R^*$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $\Gamma_D(P)$ has k paths rooted at ℓ, then there exists a $\hat{d} \in R^*$ such that $P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$.
- **Inductive Step**. Assume that $\Gamma_D(P)$ has $k+1$ paths rooted at ℓ. Then there exists at least one path rooted at ℓ in $\Gamma_D(P)$. Since all paths rooted at ℓ in $\Gamma_D(P)$ have length one, then there exists some path (ℓ, ℓ') in $\Gamma_D(P)$ such that ℓ' has no children in $\Gamma_D(P)$. Then by Lemma B.22, there exists some $d' \in (L(\Sigma) \cup R)^*$ such that $P' = P[\ell \mapsto d']$ is a proof for $\langle \Sigma \mid R \rangle$ and ℓ has k children in $\Gamma_D(P')$. Since all paths rooted at ℓ have length one, then $\Gamma_D(P')$ has k paths rooted at ℓ. Then by the inductive hypothesis, then there exists a $\hat{d} \in R^*$ such that $P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$. Then the inductive step holds.

Then by the principle of induction, there exists a $\hat{d} \in R^*$ such that $P[\ell \mapsto \hat{d}]$ is a proof for $\langle \Sigma \mid R \rangle$. □

Lemma B.24. *Let P be a finite proof for $\langle \Sigma \mid R' \rangle$ with $R' \subseteq R \subseteq \Sigma^* \times \Sigma^*$. If $P \models R$ and $\Gamma_D(P)$ is acyclic, then there exists a proof P' for $\langle \Sigma \mid R' \rangle$ such that $|P'| = |P|$, $P' \models R$, and $d \in (R')^*$ for all $(\ell, d) \in P'$.*

Proof. Let $f : L(\Sigma) \times (L(\Sigma) \cup R')^* \to \mathbb{N}$ count the number of vertices with children in the derivation graph of a proof. Since $\Gamma_D(P)$ has $|P|$ vertices, then $f(P) \leq |P|$. Since P is finite, then $f(P)$ is also finite. The proof follows by induction on $f(P)$.

- **Base Case**. Assume that $f(P) = 0$. Then there are no edges in $\Gamma_D(P)$. Then by Lemma B.18, $d \in (R')^*$ for all $(\ell, d) \in P$.
- **Inductive Hypothesis** Let Q be a proof for $\langle \Sigma \mid R' \rangle$. Assume that for some $k \in \mathbb{N}$, if $f(Q) = k$ with $P \models R$ and $\Gamma_D(Q)$ is acyclic, then there exists a proof P' for $\langle \Sigma \mid R \rangle$ such that $|P'| = |Q|$, $P' \models R$, and $d \in (R')^*$ for all $(\ell, d) \in P'$.
- **Inductive Step**. Let Q be a proof for $\langle \Sigma \mid R' \rangle$. Assume that $f(Q) = k+1$ with $Q \models R$ and $\Gamma_D(Q)$ acyclic. Since $f(Q) > 0$, then there exists at least one edge (ℓ, ℓ') in $\Gamma_D(Q)$. By Lemma B.1, there exists some path $(\ell_0, \dots, \ell_n)$ in $\Gamma_D(Q)$ such that $\ell_0 = \ell$ every path rooted at ℓ has length at most n. Since (ℓ, ℓ') is a path of length one in $\Gamma_D(Q)$ rooted at ℓ, then $n \geq 1$. Assume for the intent of contradiction that there exists a path of length at least 2 rooted at ℓ_{n-1}. Then there exists a

path (ℓ_{n-1}, x, y) in $\Gamma_D(Q)$. Then $(\ell_0, \ldots, \ell_{n-1}, x, y)$ is a path of length $n+1$ in $\Gamma_D(Q)$ rooted at ℓ. However, all paths rooted at ℓ have length at most n. Therefore, all paths rooted at ℓ_{n-1} have length at most one. Since ℓ_{n-1} is a vertex in $\Gamma_D(Q)$, then there exists some $d \in (L(\Sigma) \cup R')^*$ such that $(\ell_{n-1}, d) \in Q$. Then by Lemma B.23, there exists some $d' \in (R')^*$ such that $P' = Q[\ell_{n-1} \mapsto d']$ is a proof for $\langle \Sigma \mid R' \rangle$. Let $(z_{n-1}, r_{n-1}) = \ell_{n-1}$. Since $Q \models R$ with respect to $\langle \Sigma \mid R' \rangle$, it follows that $R \subseteq R' \cup \{r \mid ((m,r),d) \in P\}$. Then,

$$R \subseteq R' \cup \{r \mid ((z,r),d) \in P \setminus \{\ell_{n-1}, d)\} \cup \{r_{n-1}\} \subseteq R' \cup \{r \mid ((m,r),d) \in P'\}.$$

Then $P' \models R$ with respect to $\langle \Sigma \mid R' \rangle$. By Lemma B.19, $\Gamma_D(P')$ is also a subgraph of $\Gamma_D(Q)$ with $f(P') = f(Q) - 1 = k$. Since $\Gamma_D(P')$ is a subgraph of $\Gamma_D(Q)$ with $\Gamma_D(Q)$ acyclic, then $\Gamma_D(P')$ is also acyclic. Then by the inductive hypothesis, there exists a proof P'' for $\langle \Sigma \mid R \rangle$ such that $|P''| = |P'|$, $P'' \models R$, and $d \in (R')^*$ for all $(\ell, d) \in P''$. Then $|P''| = (|Q| - 1) + 1 = |Q|$, since Q is indexed. Then the inductive step holds.

Then by the principle of induction, there exists a proof P' for $\langle \Sigma \mid R \rangle$ such that $|P'| = |P|$, $P' \models R$, and $d \in (R')^*$ for all $(\ell, d) \in P$. □

Theorem B.25. *Let P be a finite proof for $\langle \Sigma \mid R' \rangle$ with $R' \subseteq R \subseteq \Sigma^* \times \Sigma^*$. If $P \models R$ and $\Gamma_D(P)$ is acyclic, then there exists a length $|R \setminus R'|$ sequence of **Rel**(+) transformations between $\langle \Sigma \mid R' \rangle$ and $\langle \Sigma \mid R \rangle$.*

Proof. By Lemma B.24, there exists a proof P' for $\langle \Sigma \mid R' \rangle$ with $P' \models R$ and $d \in (R')^*$ for all $(\ell, d) \in P$. The proof follows by induction on $|R \setminus R'|$.

- **Base Case**. If $|R \setminus R'| = 0$, then $R \subseteq R' \subseteq R$. Then $R = R'$ and $\langle \Sigma \mid R \rangle = \langle \Sigma \mid R' \rangle$. Then there exists a length zero sequence of **Rel**(+) transformations between $\langle \Sigma \mid R' \rangle$ and $\langle \Sigma \mid R \rangle$.
- **Inductive Hypothesis**. Let P' be a proof with respect to $\langle \Sigma \mid Q \rangle$ with $Q \subseteq R$ and $P' \models R$. Assume that for some $k \in \mathbb{N}$, if $|R \setminus R'| = k$, then there exists a length k sequence of **Rel**(+) transformations between $\langle \Sigma \mid Q \rangle$ and $\langle \Sigma \mid R \rangle$.
- **Inductive Step**. Under the conditions of the inductive hypothesis, assume that $|R \setminus R'| = k+1$. Then there exists some $r \in R \setminus R'$, say $(w, w') = r$. Since $P' \models R$ and $r \notin R'$, then there exists some $x \in \mathbb{N}$ and $d \in (L(\Sigma) \cup R')^*$ such that $((x,r),d) \in P'$. Since $d \in (R')^*$, then $w \sim_{R'} w'$ by validity of P'. Let $Q = R' \cup \{r\}$. Then $\langle \Sigma \mid R' \rangle \cong \langle \Sigma \mid Q \rangle$ by **Rel**(+). Since $R' \subseteq Q$ and $P' \models R$ with respect to $\langle \Sigma \mid R' \rangle$, then
$$R \subseteq R' \cup \{r \mid ((n,r),d) \in P'\} \subseteq Q \cup \{r \mid ((n,r),d) \in P'\}.$$
Then $P' \models R$ with respect to $\langle \Sigma \mid Q \rangle$. Since $r \in R \setminus R'$, then $|R \setminus Q| = |R \setminus R'| - 1 = k$. Then by the inductive hypothesis, there exists a length k sequence of **Rel**(+) transformations between $\langle \Sigma \mid Q \rangle$ and $\langle \Sigma \mid R \rangle$. Then there exists a length $k+1$ sequence of **Rel**(+) transformations between $\langle \Sigma \mid R' \rangle$ and $\langle \Sigma \mid R \rangle$. Then the inductive step holds.

Then by the principle of induction, there exists a length $|R \setminus R'|$ sequence of **Rel**(+) transformations between $\langle \Sigma \mid R' \rangle$ and $\langle \Sigma \mid R \rangle$. □

C Circuit Decompositions of Coxeter Generators

In Section 4.1, the Coxeter generator r_3 was decomposed into a circuit over Σ_D. In this section, the remaining 7 Coxeter generators are decomposed into circuits over Σ_D. Scalar multiples of the normal vectors are used freely. Recall that $CCX_{0,1}$ is a reflection about the normal vector $|\hat{b}\rangle = |1\rangle \otimes |1\rangle \otimes |-\rangle$. Similarly, $CX_{1,2}$ is a reflection about the normal vector $|\bar{b}\rangle = |-\rangle \otimes |1\rangle \otimes |1\rangle$.

r_1. This generator is defined by the normal vector $|b_1\rangle = |0\rangle \otimes |0\rangle \otimes |-\rangle$. Since $(X_0 \circ X_1)\,|\hat{b}\rangle = |b_1\rangle$ with $(X_0 \circ X_1)^{-1} = X_1 \circ X_0$, then $r_1 = X_0 \circ X_1 \circ CCX_{0,1} \circ X_1 \circ X_0$.

r_2. This generator is defined by the normal vector $|b_2\rangle = |0\rangle \otimes (|1\rangle \otimes |0\rangle - |0\rangle \otimes |1\rangle)/\sqrt{2}$. Since $|b_2\rangle$ and $-|b_2\rangle$ define the same hyperplane, then $-|b_2\rangle$ also defines the same generator. Recall that r_3 is a reflection about the normal vector $|b_3\rangle = |0\rangle \otimes |1\rangle \otimes |-\rangle$. Then $(CX_{2,1})\,|b_3\rangle = -|b_2\rangle$. Since $CX_{2,1}$ is self-inverse, then $r_2 = CX_{2,1} \circ r_3 \circ CX_{2,1}$. Since $r_3 = X_0 \circ CCX_{0,1} \circ X_0$ with X_0 and $CX_{2,1}$ commuting, then $r_2 = X_0 \circ CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1} \circ X_0$.

r_4. This generator is defined by the normal vector $|b_4\rangle = (|0\rangle \otimes |1\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle \otimes |0\rangle)/\sqrt{2}$. Since $|\bar{b}\rangle = (|0\rangle \otimes |1\rangle \otimes |1\rangle - |1\rangle \otimes |1\rangle \otimes |1\rangle)/\sqrt{2}$, then $(CX_{0,1} \circ CX_{0,2})\,|\bar{b}\rangle = |b_4\rangle$. Furthermore, since $(CX_{0,1} \circ CX_{0,2})^{-1} = CX_{0,2} \circ CX_{0,1}$, then $r_4 = CX_{0,1} \circ CX_{0,2} \circ CCX_{1,2} \circ CX_{0,2} \circ CX_{0,1}$.

r_5. This generator is defined by the normal vector $|b_5\rangle = |1\rangle \otimes |0\rangle \otimes |-\rangle$. Since $(X_1)\,|\hat{b}\rangle = |b_5\rangle$ with X_1 self-inverse, then $r_5 = X_1 \circ CCX_{0,1} \circ X_1$.

r_6. This generator is defined by the normal vector $|b_6\rangle = |1\rangle \otimes (|0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle)/\sqrt{2}$. Since $(CX_{2,1})\,|\hat{b}\rangle = |b_6\rangle$ with $CX_{2,1}$ self-inverse, then $r_6 = CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1}$.

r_7. This generator is defined by the normal vector $|b_7\rangle = |1\rangle \otimes (|0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle)/\sqrt{2}$. Recall that r_6 is a reflection about the normal vector $|b_6\rangle = |1\rangle \otimes (|0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle)/\sqrt{2}$. It follows that $(CZ_{0,1})\,|b_6\rangle = |b_7\rangle$. Since $CZ_{0,1}$ is self-inverse, then $r_7 = CZ_{0,1} \circ r_6 \circ CZ_{0,1}$. Furthermore, since $r_6 = CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1}$, then $r_7 = CZ_{0,1} \circ CX_{2,1} \circ CCX_{0,1} \circ CX_{2,1} \circ CZ_{0,1}$.

r_8. The generator is defined by the normal vector $|b_8\rangle = |+\rangle \otimes |+\rangle \otimes |+\rangle$. First, define the operator $M = K_{1,2} \circ X_1 \circ X_2 \circ CZ_{0,2}$. Clearly, $M^{-1} = CZ_{0,2} \circ X_2 \circ X_1 \circ K_{1,2}$. Furthermore,

$$M\,|\bar{b}\rangle = (K_{1,2} \circ X_1 \circ X_2)\,(|+\rangle \otimes |1\rangle \otimes |1\rangle) = K_{1,2}\,(|+\rangle \otimes |0\rangle \otimes |0\rangle) = |b_8\rangle\,.$$

Therefore, $r_8 = K_{1,2} \circ X_1 \circ X_2 \circ CZ_{0,2} \circ CCX_{1,2} \circ CZ_{0,2} \circ X_2 \circ X_1 \circ K_{1,2}$.

This establishes all decompositions of the Σ_{E8} in terms of Σ_D.

D Constructing the Generators for $W(E_8)$

The section walks through the construction of X_0, $CX_{0,1}$, $CCX_{1,2}$, and $K_{1,2}$ using the Coxeter generators for $W(E_8)$. As suggested in Section 4.1, this construction begins by deriving several diagonal matrices over (± 1).

$$w_1 = r_6 \cdot r_7 \qquad w_2 = r_6 \cdot r_5 \cdot w_1 \cdot r_5 \cdot r_6 \qquad w_3 = r_5 \cdot r_4 \cdot w_2 \cdot r_4 \cdot r_5$$
$$w_4 = r_4 \cdot r_3 \cdot w_3 \cdot r_3 \cdot r_4 \qquad w_5 = r_3 \cdot r_2 \cdot w_4 \cdot r_2 \cdot r_3 \qquad w_6 = r_2 \cdot r_1 \cdot w_5 \cdot r_1 \cdot r_2$$

For example, $[\![w_1]\!]^*_{E8} = CZ_{0,1} \circ CZ_{0,2}$. It is then possible to derive $CCX_{0,1}$ and X_2.

$$w_7 = r_7 \cdot r_8 \cdot r_6 \cdot w_6 \cdot w_4 \cdot w_2 \cdot r_8 \cdot w_6 \cdot w_4 \cdot w_2 \cdot r_6 \cdot r_8 \cdot r_7 \qquad w_8 = r_1 \cdot r_3 \cdot r_5 \cdot w_7$$

Then $[\![w_7]\!]^*_{E8} = CCX_{0,1}$ and $[\![w_8]\!]^*_{E8} = X_2$. Using $CCX_{0,1}$, it is then possible to derive $K_{1,2}$.

$$w_9 = r_6 \cdot w_7 \cdot w_1 \cdot w_7 \cdot r_6 \qquad w_{10} = r_2 \cdot r_6 \cdot w_5 \cdot w_3 \cdot w_2 \cdot r_8 \cdot w_9 \cdot r_8 \cdot w_5 \cdot w_3 \cdot w_2 \cdot w_9$$

Then $[\![w_9]\!]^*_{E8}$ is a diagonal matrix over (± 1) and $[\![w_{10}]\!]^*_{E8} = K_{1,2}$. Next, the permutations are derived.

$$w_{11} = w_{10} \cdot r_4 \cdot w_8 \cdot r_4 \cdot w_{10} \cdot w_8 \qquad w_{12} = r_2 \cdot r_6 \qquad w_{13} = w_{11} \cdot w_{12} \cdot w_{11}$$

It can be validated that $[\![w_{11}]\!]^*_{E8} = \sigma_{1,2}$, $[\![w_{12}]\!]^*_{E8} = \sigma_{0,1}$, and $[\![w_{13}]\!]^*_{E8} = \sigma_{0,2}$. As an immediate consequence, $[\![w_{13} \cdot w_7 \cdot w_{13}]\!]^*_{E8} = CCX_{1,2}$ and $[\![w_{13} \cdot w_8 \cdot w_{13}]\!]^*_{E8} = X_0$. Then by three applications of **Gen(+)**, the generators $K_{1,2}$, $CCX_{1,2}$, and X_0 are introduced, alongside the following relations.

$$K_{1,2} \approx w_{10} \qquad CCX_{1,2} \approx w_{13} \cdot w_7 \cdot w_{13} \qquad X_0 \approx w_{13} \cdot w_8 \cdot w_{13}$$

Next, define $w_{14} = w_{12} \cdot X_0 \cdot w_{10} \cdot X_0 \cdot w_{12}$. It can be validated directly that $[\![w_{14}]\!]^*_{E8} = CX_{0,1}$. Then by application of **Gen(+)**, the generator $CX_{0,1}$ is introduced, alongside the relation $CX_{0,1} \approx w_{14}$.

E Establishing the Minimality of $W(E_8)$ and $\mathrm{O}(8,\mathbb{D})$ Generators

This section establishes the minimality of certain generating sets for $W(E_8)$ and $\mathrm{O}(8,\mathbb{D})$. First, a general result about minimal generating sets is established. This result is then applied to the generating sets of interest, to prove their minimality.

E.1 Two Results on Minimal Generating Sets

Theorem E.1. *Let G be a group with $\Sigma' \subseteq \Sigma \subseteq G$. If there exists a $g \in G$ such that g commutes with the elements of Σ' and g does not commute with the elements of Σ, then $\langle \Sigma' \rangle$ is a proper subgroup of $\langle \Sigma \rangle$.*

Proof. Assume that $g \in G$, g commutes with every element of Σ', and $\langle \Sigma' \rangle = \langle \Sigma \rangle$. It follows by induction on the length of an element in $\langle \Sigma' \rangle$, that g commutes with every element in $\langle \Sigma' \rangle$. As a base case, if $h \in \langle \Sigma' \rangle$ corresponds to a word of length 0, then h is the identity and $g \circ h = g = h \circ g$. As an inductive hypothesis, assume that for some $n \in \mathbb{N}$, if $h_1, h_2, \dots, h_n \in \Sigma'$ and $h = h_1 \circ h_2 \circ \cdots \circ h_n$, then $g \circ h = h \circ g$. To show that the inductive step holds, let $h_1, h_2, \dots, h_n, h_{n+1} \in \Sigma'$ and $h = h_1 \circ h_2 \circ \cdots h_{n+1}$. By the inductive hypothesis, $g \circ h' = h' \circ g$ where $h' = h_1 \circ h_2 \circ \cdots \circ h_n$. Then $g \circ h = g \circ h' \circ h_{n+1} = h' \circ g \circ h_{n+1} = h' \circ h_{n+1} \circ g = h \circ g$. Then the inductive step holds, and g commutes with every element of $\langle \Sigma' \rangle$. In particular, g commutes with Σ. By the contrapositive, if g does not commute with Σ, then $\langle \Sigma' \rangle \neq \langle \Sigma \rangle$. However, $\langle \Sigma' \rangle \leq \langle \Sigma \rangle$ since $\Sigma' \subseteq \Sigma$. Therefore, $\langle \Sigma' \rangle$ is a proper subgroup of $\langle \Sigma \rangle$. □

Lemma E.2. *Let G be a group with $\Sigma \subseteq G$. If for every maximal proper subset Σ' of Σ, $\langle \Sigma' \rangle$ is a proper subgroup of $\langle \Sigma \rangle$, then Σ is a minimal generating set for $\langle \Sigma \rangle$.*

Proof. Let Σ' be a proper subset of Σ. Then there exists some maximal proper subset Π of Σ such that $\Sigma' \subseteq \Pi \subseteq \Sigma$. Then $\langle \Sigma' \rangle \leq \langle \Pi \rangle \leq \langle \Sigma \rangle$. Since Π is maximal, then by assumption, $\langle \Pi \rangle$ is a proper subgroup of $\langle \Sigma \rangle$. Consequently, $\langle \Sigma' \rangle$ is a proper subgroup of $\langle \Sigma \rangle$. Since Σ' was arbitrary, then Σ is a minimal generating set for $\langle \Sigma \rangle$. □

E.2 Minimality for $W(E_8)$

It must be shown that for every maximal proper subset Σ' of Σ_0, there exists some 8×8 dyadic matrix M such that M commutes with Σ' but does not commute with Σ_0. The first three cases can be solved by inspection. In fact, these matrices follow from well-known circuit relations.

1. Z_2 commutes with $\{X_0, CX_{0,1}, CCX_{1,2}\}$ but does not commute with $K_{1,2}$.
2. H_2 commutes with $\{X_0, CX_{0,1}, K_{1,2}\}$ but does not commute with $CCX_{1,2}$.
3. X_0 commutes with $\{X_0, CCX_{1,2}, K_{1,2}\}$ but does not commute with $CX_{0,1}$.

The final case is less obvious, but can be reduced to solving a linear integer program. Assume that there exists such a matrix M. Since M is dyadic, then there exists some integer matrix N and integer k such that $M = N/2^k$. Clearly, M and N commute with the same matrices. Then M is characterized by the following four equations.

$$X_0 \circ N \neq N \circ X_0 \qquad CX_{0,1} \circ N = N \circ CX_{0,1} \qquad CCX_{1,2} \circ N = N \circ CCX_{1,2} \qquad K_{1,2} \circ N = N \circ K_{1,2}$$

Without loss of generality, $K_{1,2}$ can be replaced by its integral scalar multiple $2 \cdot K_{1,2}$. Then the entries of N can be thought of as 64 integer variables, with each equation yielding 64 linear constraints. Using

Z3 [20] as a solver, the following solution is obtained.

$$N_{0,0} = \begin{bmatrix} 4 & 2 & 2 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad N_{0,1} = N_{1,0} = N_{1,1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad N = \begin{bmatrix} N_{0,0} & M_{0,1} \\ N_{1,0} & M_{1,1} \end{bmatrix}$$

This establishes Theorem 4.4.

E.3 Minimality of Σ_K for $\mathrm{O}(8,\mathbb{D})$

First, it will be shown that Σ_K is a minimal generating set for $\mathrm{O}(8,\mathbb{D})$. To see that Σ_K generates $\mathrm{O}(8,\mathbb{D})$ simply note that $\Sigma_K \cup \{K_{1,2}\}$ generates $\mathrm{O}(8,\mathbb{D})$ with $K_{1,2} = K_{[0,1,2,3]} \circ X_0 \circ K_{[0,1,2,3]} \circ X_0$. It remains to be shown that for every maximal proper subset Σ' of Σ_K, there exists some 8×8 dyadic matrix M such that M commutes with Σ' but does not commute with Σ_K. The first three cases are also solved by inspection, using well-known circuit relations.

1. Z_2 commutes with $\{X_0, CX_{0,1}, CCX_{1,2}\}$ but does not commute with $K_{[0,1,2,3]}$.
2. H_2 commutes with $\{X_0, CX_{0,1}, K_{[0,1,2,3]}\}$ but does not commute with $CCX_{1,2}$.
3. $X_2 \circ CZ_{0,2} \circ X_2$ commutes with $\{CX_{0,1}, CCX_{1,2}, K_{[0,1,2,3]}\}$ but does not commute with X_0.

Using Z3 as in the $W(E_8)$, it is then possible to find an integer matrix L such that L commutes with $\{X_0, CCX_{1,2}, K_{[0,1,2,3]}\}$ but does not commute with $CX_{0,1}$. The solution is as follows.

$$L_0 = \begin{bmatrix} 1 & 2 & 2 & 0 \\ 2 & 0 & -1 & 0 \\ 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -3 \end{bmatrix} \qquad L_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} L_0 & L_1 \\ L_1 & L_0 \end{bmatrix}$$

This establishes the first claim of Theorem 5.3.

E.4 Minimality of Σ_Z for $\mathrm{O}(8,\mathbb{D})$

The proof the Σ_Z is minimal proves more challenging. However, the minimality of Σ_K can be used to simplify this argument significantly. Clearly, $\Sigma_D = \Sigma_Z \setminus \{CCZ\}$ does not generate $\mathrm{O}(8,\mathbb{D})$, since $\langle \Sigma_D \rangle = W(E_8) < \mathrm{O}(8,\mathbb{D})$. Three of the remaining four cases can be solved by inspection.

1. Recall N from Appendix E.2. By construction, this matrix commutes with $\{CX_{0,1}, CCX_{1,2}, K_{1,2}\}$ and does not commute with X_0. Furthermore, $CCZ \circ N = N \circ CCZ$, since the 8-th row and 8-th column of N contain only zeros.
2. $\sigma_{1,2}$ commutes with $\{X_0, CCX_{1,2}, K_{1,2}, CCZ\}$ but does not commute with $CX_{0,1}$.
3. Z_2 commutes with $\{X_0, CX_{0,1}, CCX_{1,2}, CCZ\}$ but does not commute with $K_{1,2}$.

The case of $CCX_{1,2}$ requires more care. There is no obvious operation which commutes with all generators except for $CCX_{1,2}$. Furthermore, Z3 fails to find an solution to the corresponding integer program. Instead, consider the automorphism $f : M \mapsto H_2 \circ M \circ H_2$ of $\mathrm{O}(8,\mathbb{D})$. Since f fixes $\{X_0, CX_{0,1}, K_{1,2,}\}$ and maps CCZ to $CCX_{0,1}$, then f induces an isomorphism between the subgroups $\langle X_0, CX_{0,1}, K_{1,2}, CCZ \rangle$ and $\langle X_0, CX_{0,1}, K_{1,2}, CCX_{0,1} \rangle$ of $\mathrm{O}(8,\mathbb{D})$. Since $X_0, CX_{0,1}$, $K_{1,2,}, CCX_{0,1} \in W(E_8)$ with $W(E_8)$ finite, then

$$|\langle X_0, CX_{0,1}, K_{1,2}, CCZ \rangle| = |\langle X_0, CX_{0,1}, K_{1,2}, CCX_{1,2} \rangle| \le |W(E_8)| < \infty.$$

Since $\mathrm{O}(8,\mathbb{D})$ is an infinite group, then $\langle X_0, CX_{0,1}, K_{1,2}, CCZ \rangle < \mathrm{O}(8,\mathbb{D})$. This establishes the second claim of Theorem 5.3.

F Proof Details for a Presentation of $\mathrm{O}(8,\mathbb{D})$

In Section 5, many informal claims were made about the relations in $\mathscr{G}_n$, and the derivations that are possible using these relations. This section restates each claim as a lemma or theorem, provides a proof for each claim, and then explains how these claims establish the lemmas and theorems in Section 5.

F.1 Counting the Relations in $\mathscr{R}_n$

This section validates the claim that $\mathscr{G}_8$ contains 2039 relations. We say that two relations $(q,r) \in \mathscr{G}_8$ and $(q',r') \in \mathscr{G}_8$ are distinct if $q \neq q'$ or $r \neq r'$. This means, for example, that the relations $K_{[0,1,2,3]} \cdot K_{[4,5,6,7]} \approx K_{[4,5,6,7]} \cdot K_{[4,5,6,7]}$ and $K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot K_{[4,5,6,7]}$ are distinct The techniques used in this section can be generalized to count the number of relations in $\mathscr{G}_n$.

First, consider the relations whose parameters are linearly ordered. If a relation schema r has m linearly ordered parameters, then each choice of m distinct numbers in $[n]$ corresponds to a unique instance of r. It follows that a relation schema with m linearly ordered parameters corresponds to $\binom{n}{m}$ unique relations. For each choice of m, we compute $\binom{8}{m}$ and count the number of relations with m linearly ordered parameters.

- If $m = 1$, then there are $\binom{8}{1} = 8$ instances. The only relation with a single parameter is Relation (48). Then this case contributes 8 relations.
- If $m = 2$, then there are $\binom{8}{2} = 28$ instances. The only relations with two parameters, all linearly ordered, are Relations (47) and (58). Then this case contributes 56 relations.
- If $m = 3$, then there are $\binom{8}{3} = 56$ instances. The only relations with three parameters, all linearly ordered, are Relations (56) and (57). Then this case contributes 102 relations.
- If $m = 4$, then there are $\binom{8}{4} = 70$ instances. The relations Relations (49), (63), (64) and (65) all have exactly four parameters, which are all linearly ordered. Then this case contributes 280 relations.
- If $m = 5$, then there are $\binom{8}{5} = 56$ instances. The relations Relations (59), (60), (61) and (62) all have exactly five parameters, which are all linearly ordered. This this case contributes 224 relations.
- If $m = 6$, then there are $\binom{8}{6} = 28$ instances. The only relation with a six parameters is Relation (66). Then this case contributes 28 relations.
- If $m = 8$, then there are $\binom{8}{8} = 1$ instances. The only relation with eight parameters is Relation (67). Then this case contributes 1 relations.

In total, the relation schemata with linearly ordered parameters contribute 699 instances.

The remaining six schemata induce a partial order on the parameters. For example, in Relation (55), the term $K_{[a,b,c,d]} \cdot K_{[e,f,g,h]}$ indicates that $a < b < c < d$ and $e < f < g < h$. However, the choices of (a,b,c,d) are independent from the choices of (e,f,g,h), except that all choices must be distinct. In this example, there are $\binom{n}{4}$ ways to select the four indices in the first order. Then $n - m$ indices remain, from which there are $\binom{n-4}{4}$ choices. In general, for two independent linear orders with m and k parameters respectively, there will be $\binom{n}{m} \cdot \binom{n-m}{k}$ choices. The six schemata are described below.

- In Relation (54), $m = 1$ and $k = 1$, resulting in $\binom{8}{1} \cdot \binom{7}{1} = 56$ choices.
- In Relation (51), $m = 1$ and $k = 2$, resulting in $\binom{8}{1} \cdot \binom{7}{2} = 168$ choices.
- In Relation (53), $m = 1$ and $k = 4$, resulting in $\binom{8}{1} \cdot \binom{7}{4} = 280$ choices.
- In Relation (50), $m = 2$ and $k = 2$, resulting in $\binom{8}{2} \cdot \binom{6}{2} = 420$ choices.

- In Relation (52), $m = 2$ and $k = 4$, resulting in $\binom{8}{2} \cdot \binom{6}{4} = 420$ choices.
- In Relation (55), $m = 4$ and $k = 4$, resulting in $\binom{8}{4} \cdot \binom{4}{4} = 70$ choices

In total, the relations partially ordered parameters contribute 1414 instances. Then $|\mathscr{R}_8| = 2113$.

F.2 Correctness of Relation Reindexing

This section justifies the reindexing of relations via permutations. First, recall that every permutation on $[n]$ can be represented by a permutation of the basis vectors in $\mathbb{R}^8$, with $\tau_{a,b}$ corresponding to $X_{[a,b]}$. The intuition is that every σ can be represented by a word w over generators of type X, and that conjugation by w corresponds to formal reindexing when σ is valid.

First, a subset $\mathscr{R}_n^B$ of $\mathscr{R}_n$ is identified, for which all of the order and braiding relations for generators of type X hold. Consequently, $\mathscr{R}_n^B$ is complete for words over generators of type X. Then $\mathscr{R}_n^B$ is extended to a subset $\mathscr{R}_n^\sigma$ of $\mathscr{R}_n$ for which all formal reindexings are derivable. The result is proven first, for individual generators, and then extended to entire words.

F.2.1 Deriving the Braiding Relations

First, define the set of relations,

$$\mathscr{R}_n^\tau = \left\{ X_{[a,a+1]}{}^2 \approx \varepsilon \mid a \in \mathbb{Z} \right\} \cup \left\{ X_{[a,a+1]} \cdot X_{[a,b]} \approx X_{[a+1,b]} \cdot X_{[a,a+1]} \mid a, b \in \mathbb{Z} \text{ with } a+1 < b \right\}.$$

This set is sufficient to decompose all swaps into transpositions, as proven in Lemma F.1. Of interest in this section is the following extension of $\mathscr{R}_n^\tau$,

$$\mathscr{R}_n^B = \mathscr{R}_n^\tau \cup \left\{ X_{[a+1,a+2]} \cdot X_{[a,a+1]} \approx X_{[a,a+2]} \cdot X_{[a+1,a+2]} \mid a \in \mathbb{Z} \right\}.$$

The relations in $\mathscr{R}_n^B$ entail the braiding and order relations for $S(n)$, as shown in Lemma F.2, and are therefore complete for equality of words over generators of type X. Of important note is that $\mathscr{R}_n^B \subseteq \mathscr{R}_n$.

Lemma F.1. *Let R be a set of relations over $\mathscr{G}_n$ which contains all well-formed relations in $\mathscr{R}_n^\tau$. If v is a word over two-level operators of type X, then there exists a word u over transpositions such that $v \sim_R u$.*

Proof. Consider a valid two-level operator $X_{[a,b]}$. The proof follows by induction on $b - a > 0$.

- **Base Case**. If $b - a = 1$, then $X_{[a,b]} = X_{[a,a+1]}$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}_{>0}$, if $b - a = k$, then there exists a word w
- **Inductive Step**. Assume that $b - a = k + 1$. Since $k > 0$, then $a + 1 \neq b$, and the following derivation holds.

  $$X_{[a,b]} \leftarrow X_{[a,a+1]}{}^2 \cdot X_{[a,b]} \rightarrow X_{[a,a+1]} \cdot X_{[a+1,b]} \cdot X_{[a,a+1]}$$

  Since $b - (a+1) = k$, then by the inductive hypothesis, there exists a word w over transpositions such that $X_{[a+1,b]} \sim_R w$. Then $X_{[a,b]} \sim_R X_{[a,a+1]} \cdot w \cdot X_{[a,a+1]}$. Since $X_{[a,a+1]} \cdot w \cdot X_{[a,a+1]}$ is a word over transpositions, then the inductive hypothesis holds.

Then for each symbol M in w, there exists a decomposition of M into transpositions. Then by Appendix B.2, there exists a word v over transpositions such that $w \sim_R v$. □

Lemma F.2. *Let R be a set of relations over $\mathscr{G}_n$ which contains all well-formed relations in $\mathscr{R}_n^B$. If v and w are words over two-level operators of type X and $[\![v]\!]_O^* = [\![w]\!]_O^*$, then $v \sim_{\mathscr{R}_\sigma} w$.*

Proof. Since v and w are words over two-level operators of type X with $\mathscr{R}_n^\tau \subseteq \mathscr{R}_n^B$, then by Lemma F.1 there exists words $\hat{v}$ and $\hat{w}$ over transpositions such that $\hat{v} \sim \hat{w}$. Then $\hat{v}$ and $\hat{w}$ are words in the braid representation of $\mathrm{S}(n)$. If R contains all order and braiding relations for the transpositions in $\mathscr{G}_n$, then R is complete for words over the transpositions in $\mathscr{G}_n$. Let $a \in [n-2]$. Then the following derivation holds using the relations in $\mathscr{R}_n^B$.

$$X_{[a,a+1]} \cdot X_{[a+1,a+2]} \cdot X_{[a,a+1]} \leftarrow {X_{[a,a+1]}}^2 \cdot X_{[a,a+2]} \rightarrow X_{[a,a+2]} \leftarrow X_{[a,a+2]} \cdot {X_{[a+1,a+2]}}^2 \leftarrow X_{[a+1,a+2]} \cdot X_{[a,a+1]} \cdot X_{[a+1,a+2]}$$

Then $X_{[a,a+1]} \cdot X_{[a+1,a+2]} \cdot X_{[a,a+1]} \sim_R X_{[a+1,a+2]} \cdot X_{[a,a+1]} \cdot X_{[a+1,a+2]}$. Since a was arbitrary, then R is complete for $\mathrm{S}(n)$. Since $v \sim_R \hat{v}$ and $w \sim_R \hat{w}$, then $[\![v]\!]_O^* = [\![\hat{v}]\!]_O^*$ and $[\![w]\!]_O^* = [\![\hat{w}]\!]_O^*$. Then $[\![\hat{v}]\!]_O^* = [\![\hat{w}]\!]_O^*$. Then $\hat{v} \sim_R \hat{w}$ by the completeness of R. Then $v \sim_R w$ by the transitivity of $(\sim_R)$. □

F.2.2 Formal Inverses for Self-Inverse Generators

For each $w = w_1 \cdot w_2 \cdots w_n$ over $\mathscr{G}_n$, define $\overline{w} = w_n \cdots w_2 \cdot w_1$. Since each element in $\mathscr{G}_n$ is self-inverse, then $[\![\overline{w}]\!]_O^*$ is the inverse of $[\![w]\!]_O^*$ in $\mathrm{O}(8, \mathbb{D})$. One can prove that given a complete set of relations, both $u \cdot \overline{u}$ and $\overline{u} \cdot u$ always derive to ε. For the purposes of this proof, only the case for X-type generators is necessary.

Lemma F.3. *Let R be a set of relations over $\mathscr{G}_n$ which contains all well-formed relations in the set below.*

$$\{{X_{[a,a+1]}}^2 \approx \varepsilon \mid a \in \mathbb{Z}\} \cup \{X_{[a,a+1]} \cdot X_{[a,b]} \approx X_{[a+1,b]} \cdot X_{[a,a+1]} \mid a, b \in \mathbb{Z}\} \cup \{X_{[a+1,a+2]} \cdot X_{[a,a+1]} \approx X_{[a,a+@]} \cdot X_{[a+1,a+2]} \mid a \in \mathbb{Z}\}$$

If u is a word over two-level operators of type X, then $u \cdot \overline{u} \sim_R \varepsilon$ and $\overline{u} \cdot u \sim_R \varepsilon$. Furthermore, if v is a word over two-level operators of type X and $u \sim_R v$, then $\overline{u} \sim_R \overline{v}$.

Proof. Since $[\![\cdot]\!]_O^*$ maps each generator in $\mathscr{G}_n$ to a self-inverse matrix, then $[\![\overline{u}]\!]_O^*$ is the inverse to $[\![u]\!]_O^*$. Then $[\![\overline{u} \cdot u]\!]_O^* = [\![\varepsilon]\!]_O^* = [\![u \cdot \overline{u}]\!]_O^*$. Since u is a word over two-level operators of type X, then $u \cdot \overline{u} \sim_R \varepsilon$ and $\overline{u} \cdot u \sim_R \varepsilon$ by Lemma F.2. Now assume that v is a word over two-level operators of type X with $u \sim_R v$. Then $[\![u]\!]_O^* = [\![v]\!]_O^*$. Since $\overline{u}$ is the inverse to u and $\overline{v}$ is the inverse to v, then $[\![\overline{u}]\!]_O^* = [\![\overline{v}]\!]_O^*$. Then $\overline{u} \sim_R \overline{v}$ by Lemma F.2. □

F.2.3 Permuting the Indices in Multi-Level Operators

Theorem F.4. *If $\sigma \in \mathrm{S}(n)$ is a valid reindexing for a two-level operator M of type X, then there exists a word v over the two-level operators of type X, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since M is a two-level operator of type X, then there exists an increasing sequence (a,b) over $[n]$ such that $M = X_{[a,b]}$. Let $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_m$ be the decomposition of σ into a sequence of transpositions. Then define $v = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_m]\!]_S$. Clearly v is a word over two-level operators of type X satisfying $[\![v]\!]_O^* = [\![\sigma]\!]_S$. Furthermore, $[\![\sigma(M)]\!]_O^* = [\![\tau_{\sigma(a),\sigma(b)}]\!]_S = [\![\sigma \cdot \tau_{a,b} \cdot \sigma^{-1}]\!]_S^* = [\![v \cdot M \cdot \overline{v}]\!]_O^*$. Since $\sigma(M)$ and $\overline{v} \cdot M \cdot v$ are words over two-level operators of type X, then $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ by Lemma F.2. □

Theorem F.5. *If $\sigma \in \mathrm{S}(n)$ and M is a one-level operator of type (-1), then there exists a word v of transpositions, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since M is a one-level operator of type (-1), then there exists an $a \in [a]$ such that $M = (-1)_{[a]}$. Let v Let $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_m$ be the decomposition of σ into a sequence of transpositions. Then define $v = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_m]\!]_S$. It follows by induction on m that $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.

- **Base Case**. Assume that $m = 0$. Then $v = \overline{v} = \varepsilon$ and $\sigma(M) = M$. Then $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ by the reflexivity of $(\sim_{\mathscr{R}_\sigma})$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $m = k$, then $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.
- **Inductive Step**. Assume that $m = k+1$ and define $u = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_k]\!]_S$. Then by the inductive hypothesis $\sigma(M) \sim_{\mathscr{R}_\sigma} u \cdot \sigma_m(M) \cdot \overline{u}$. Since σ_m is a transposition, then there exists some $j \in [n-1]$ such that $\sigma_m = \tau_{j,j+1}$. Furthermore, $[\![\sigma_m]\!]_S = X_{[j,j+1]}$. If $j = a$, then $\sigma_m(M) = (-1)_{[a+1]}$ then the following derivation holds using only Relations (47) and (58).

  $$(-1)_{[a+1]} \leftarrow (-1)_{[a+1]} \cdot {X_{[j,j+1]}}^2 \leftarrow X_{[j,j+1]} \cdot (-1)_{[a]} \cdot X_{[j+1]}$$

  The case when $j+1 = a$ follows symmetrically. When $j \neq a$ and $j+1 \neq a$, then $\sigma_m(M) = M$ and the following derivation holds using only Relations (47) and (51).

  $$(-1)_{[a]} \leftarrow (-1)_{[a]} \cdot {X_{[j,j+1]}}^2 \leftarrow X_{[j,j+1]} \cdot (-1)_{[a]} \cdot X_{[j+1]}$$

  In either case, $\sigma_m(M) \sim_{\mathscr{R}_\sigma} [\![\sigma_m]\!]_S \cdot M \cdot [\![\sigma_m]\!]_S$. Then $\sigma_m(M) \sim_{\mathscr{R}_\sigma} u \cdot \sigma_m(M) \cdot \overline{u} \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ and the inductive step is established.

Then by the principle of induction, $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$. Clearly $[\![v]\!]_O^* = [\![\sigma]\!]_S$. □

Lemma F.6. *For each four-level operator M of type K, there exists a valid reindexing σ for M and a word v over two-level operators of type X, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $K_{[0,1,2,3]} \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since M is a four-level operator of type K, then there exists an increasing sequence (a_0, a_1, a_2, a_3) over $[n]$ such that $M = (-1)_{[a_0,a_1,a_2,a_3]}$. Since (a_0, a_1, a_3, a_4) is increasing, then $k \leq a_k$ for $k \in [4]$. Then for each $k \in [4]$, define σ_k to be τ_{k,a_k} if $k \neq a_k$, or identity otherwise. Then the following equations hold.

$$\sigma_0(M) = (-1)_{[0,a_1,a_2,a_3]} \qquad \sigma_1(\sigma_0(M)) = (-1)_{[0,1,a_2,a_3]}$$
$$\sigma_2(\sigma_1(\sigma_0(M))) = (-1)_{[0,1,2,a_3]} \qquad \sigma_3(\sigma_2(\sigma_1(\sigma_0(M)))) = (-1)_{[0,1,2,3]}$$

Let $v = [\![\sigma_3]\!]_S \cdot [\![\sigma_2]\!]_S \cdot [\![\sigma_1]\!]_S \cdot [\![\sigma_0]\!]_S$. Then the following derivations hold by Relations (47), (59), (60), (61) and (62). We assume that each σ_k is not the identity, else the derivation is trivial.

$$\begin{aligned}
\sigma_0(M) &\leftarrow {X_{[0,a_0]}}^2 \cdot K_{[0,a_1,a_2,a_3]} \rightarrow X_{[0,a_0]} \cdot K_{[a_0,a_1,a_2,a_3]} \cdot X_{[0,a_0]} \\
\sigma_1(\sigma_0(M)) &\leftarrow {X_{[1,a_1]}}^2 \cdot K_{[0,1,a_2,a_3]} \rightarrow X_{[1,a_1]} \cdot K_{[0,a_1,a_2,a_3]} \cdot X_{[1,a_1]} = X_{[1,a_1]} \cdot \sigma_0(M) \cdot X_{[1,a_1]} \\
\sigma_2(\sigma_1(\sigma_0(M))) &\leftarrow {X_{[2,a_2]}}^2 \cdot K_{[0,1,2,a_3]} \rightarrow X_{[2,a_2]} \cdot K_{[0,1,a_2,a_3]} \cdot X_{[2,a_2]} = X_{[2,a_2]} \cdot \sigma_1(\sigma_0(M)) \cdot X_{[2,a_2]} \\
\sigma_3(\sigma_2(\sigma_1(\sigma_0(M)))) &\leftarrow {X_{[3,a_3]}}^2 \cdot K_{[0,1,2,3]} \rightarrow X_{[3,a_3]} \cdot K_{[0,1,2,a_3]} \cdot X_{[3,a_3]} = X_{[3,a_3]} \cdot \sigma_1(\sigma_0(M)) \cdot X_{[3,a_3]}
\end{aligned}$$

It follows that $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ where $\sigma = \sigma_3 \cdot \sigma_2 \cdot \sigma_1 \cdot \sigma_0$. Clearly $[\![v]\!]_O^* = [\![\sigma]\!]_S$. □

Lemma F.7. *Let $M = K_{[0,1,2,3]}$ be a four-level operator of dimension n. For any increasing sequence (a_0, a_1, a_2, a_3) over $[n]$, there exists a valid reindexing σ for M and a word v over two-level operators of type X, such that $\sigma(M) = K_{[a_0,a_1,a_2,a_3]}$, $[\![v]\!]_O^* = [\![\sigma]\!]_S$, and $K_{[a_0,a_1,a_2,a_3]} \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since (a_0, a_1, a_3, a_4) is increasing, then $k \leq a_k$ for $k \in [4]$. Then for each $k \in [4]$, define σ_k to be τ_{k,a_k} if $k \neq a_k$, or identity otherwise. Then the following equations hold.

$$\sigma_3(M) = (-1)_{[0,1,2,a_3]} \qquad \sigma_2(\sigma_3(M)) = (-1)_{[0,1,a_2,a_3]}$$
$$\sigma_1(\sigma_2(\sigma_3(M))) = (-1)_{[0,a_1,a_2,a_3]} \qquad \sigma_0(\sigma_1(\sigma_2(\sigma_3(M)))) = (-1)_{[a_0,a_2,a_3,a_4]}$$

Let $v = [\![\sigma_0]\!]_S \cdot [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdot [\![\sigma_3]\!]_S$. Then the following derivations hold by Relations (47), (59), (60), (61) and (62). We assume that each σ_k is not the identity, else the derivation is trivial.

$$\begin{aligned} \sigma_3(M) &\leftarrow K_{[0,1,2,3]} \cdot X_{[3,a_3]}{}^2 \leftarrow X_{[3,a_3]} \cdot K_{[0,1,2,a_3]} \cdot X_{[3,a_3]} \\ \sigma_2(\sigma_3(M)) &\leftarrow K_{[0,1,a_2,a_3]} \cdot X_{[2,a_2]}{}^2 \leftarrow X_{[2,a_2]} \cdot K_{[0,1,2,a_3]} \cdot X_{[2,a_2]} = X_{[2,a_2]} \cdot \sigma_3(M) \cdot X_{[2,a_2]} \\ \sigma_1(\sigma_2(\sigma_3(M))) &\leftarrow K_{[0,a_1,a_2,a_3]} \cdot X_{[1,a_1]}{}^2 \leftarrow X_{[1,a_1]} \cdot K_{[0,1,a_2,a_3]} \cdot X_{[1,a_1]} = X_{[1,a_1]} \cdot \sigma_2(\sigma_3(M)) \cdot X_{[1,a_1]} \\ \sigma_0(\sigma_1(\sigma_2(\sigma_3(M)))) &\leftarrow K_{[a_0,a_1,a_2,a_3]} \cdot X_{[0,a_0]}{}^2 \leftarrow X_{[0,a_0]} \cdot K_{[0,a_1,a_2,a_3]} \cdot X_{[0,a_0]} = X_{[0,a_0]} \cdot \sigma_1(\sigma_2(\sigma_3(M))) \cdot X_{[0,a_0]} \end{aligned}$$

It follows that $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ where $\sigma = \sigma_0 \cdot \sigma_1 \cdot \sigma_2 \cdot \sigma_3$. Clearly $[\![v]\!]_O^* = [\![\sigma]\!]_S$. □

Lemma F.8. *If $\sigma \in \mathrm{S}(n)$ is a valid reindexing for a four-level operator M of type K and $\sigma(M) = M$, then there exists a word v over the two-level operators of type X, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $M \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since M is a four-level operator of type K, then there exists an increasing sequence (a,b,c,d) over $[n]$ such that $M = (-1)_{[a,b,c,d]}$. Since $\sigma(M) = M$, then σ fixes $\{a,b,c,d\}$. Then σ restricts to a permutation on $[n] \setminus \{a,b,c,d\}$. Decompose this restriction of σ into a sequence of transpositions $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_m$ on $[n] \setminus \{a,b,c,d\}$. Since σ fixes $\{a,b,c,d\}$, then $\sigma = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_m$ when viewing each σ_j as a permutation on $[n]$. Define $v = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_m]\!]_S$. It follows by induction on m that $M \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.

- **Base Case**. If $m = 0$, then $v = \overline{v} = \varepsilon$. Then $M \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$ by the reflexivity of $(\sim_{\mathscr{R}_\sigma})$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $m = k$, then $M \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.
- **Inductive Step**. Assume that $m = k + 1$ and define $u = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_k]\!]_S$. Then by the inductive hypothesis $M \sim_{\mathscr{R}_\sigma} u \cdot M \cdot \overline{u}$. Since σ_m is a transposition of elements in $[n] \setminus \{a,b,c,d\}$, then there exists some $j,l \in [n] \setminus \{a,b,c,d\}$ such that $\sigma_m = \tau_{j,l}$. Furthermore, $[\![\sigma_m]\!]_S = X_{[j,l]}$. Since $j,l \notin \{a,b,c,d\}$, then the following derivation holds by Relations (47) and (52).

  $$K_{[a,b,c,d]} \leftarrow X_{[j,k]}^2 \cdot K_{[a,b,c,d]} \rightarrow X_{[j,l]} \cdot K_{[a,b,c,d]} \cdot X_{[j,l]}$$

  Then $M \sim_{\mathscr{R}_\sigma} [\![\sigma_m]\!]_S \cdot M \cdot [\![\sigma_m]\!]_S$. Then $M \;\sim_{\mathscr{R}_\sigma} u \cdot M \cdot \overline{u} \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.

Then by the principle of induction, $M \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$. Clearly $[\![v]\!]_O^* = [\![\sigma]\!]_S$. □

Theorem F.9. *If $\sigma \in \mathrm{S}(n)$ is a valid reindexing for a four-level operator M of type K, then there exists a word v over the two-level operators of type X, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \overline{v}$.*

Proof. Since M is a four-level operator of type K, then there exists an increasing sequence (a_0,a_1,a_2,a_3) over $[n]$ such that $M = (-1)_{[a_0,a_1,a_2,a_3]}$. By Lemma F.6, there exists a word u over two-level operators of type X, and a permutation σ_1 such that $\sigma_1(M) = (-1)_{[0,1,2,3]}$, $[\![\sigma_1]\!]_S = [\![u]\!]_S^*$, and $\sigma_1(M) \sim_{\mathscr{R}_\sigma} u \cdot M \cdot \overline{u}$. By Lemma F.7, there exists a word v over two-level operators of type X, and a permutation σ_2 such that $\sigma_2(\sigma_1(M)) = (-1)_{[\sigma(a),\sigma(b),\sigma(b),\sigma(c)]} = \sigma(M)$, $[\![\sigma_2]\!]_S = [\![v]\!]_S^*$, and $\sigma_2(\sigma_1(M)) \sim_{\mathscr{R}_\sigma} v \cdot \sigma_1(M) \cdot \overline{v}$. Then define $\sigma_3 = \sigma \circ \sigma_1^{-1} \circ \sigma_2^{-1}$. Then $\sigma_3(\sigma(a_k)) = \sigma(\sigma_1^{-1}(\sigma_2^{-1}(\sigma(a_k)))) = \sigma(\sigma_1^{-1}(k)) = \sigma(a_k)$ for all $k \in [4]$. Then σ_3 is a valid reindexing for M with $\sigma_3(\sigma_2(\sigma_1(M))) = M$. By Lemma F.8, there exists a word w over two-level operators of type X, such that $[\![\sigma_3]\!]_S = [\![w]\!]_S^*$ and $\sigma(M) \sim_{\mathscr{R}_\sigma} w \cdot \sigma_2(\sigma_1(M)) \cdot \overline{w}$. It follows that $\sigma(M) \sim_{\mathscr{R}_\sigma} w \cdot \sigma_2(\sigma_1(M))\overline{w} \sim_{\mathscr{R}_\sigma} w \cdot u \cdot \sigma_1(M) \cdot \overline{w \cdot u} \sim_{\mathscr{R}_\sigma} w \cdot u \cdot v \cdot M \cdot \overline{w \cdot u \cdot v}$. Moreover, $[\![w \cdot u \cdot v]\!]_O^* = [\![w]\!]_O^* \circ [\![u]\!]_O^* \circ [\![v]\!] = [\![\sigma_3]\!]_S \circ [\![\sigma_2]\!]_S \circ [\![\sigma_1]\!]_S = [\![\sigma]\!]_S$. □

F.2.4 Permuting the Indices in Relations Over Multi-Level Operators

Theorem F.10. *For each $\sigma \in \mathrm{S}(n)$, there exits a set of words L_σ with the following properties.*

1. *If $v_1 \in L_\sigma$ and $v_2 \in L_\sigma$, then $v_1 \sim_{\mathscr{R}_\sigma} v_2$.*
2. *If σ is a valid reindexing for w, then there exists a $v \in L_\sigma$ such that $\sigma(w) \sim_{\mathscr{G}_\sigma} v \cdot w \cdot \bar{v}$.*

Proof. Let L_σ be the set of all words v over the two-level operators of type X, such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$. Let $v_1 \in L_\sigma$ and $v_2 \in L_\sigma$. Then $[\![v_1]\!]_O^* = [\![\sigma]\!]_S = [\![v_2]\!]_O^*$. Then $v_1 \sim_{\mathscr{G}_\sigma} v_2$. Since v_1 and v_2 were arbitrary, then Property (1) holds. Now assume that σ is a valid reindexing for w. Property (2) follows by induction on the length of w.

- **Base Case**. If $|w| = 0$, then $\sigma(w) = w$. Let $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_m$ be a decomposition of σ into transpositions. Define $v = [\![\sigma_1]\!]_S \cdot [\![\sigma_2]\!]_S \cdots [\![\sigma_m]\!]$. Then $[\![\bar{v}]\!]_O^* = [\![\sigma^{-1}]\!]_S$ since $\bar{v}$ is the inverse to v. Since $v \cdot w \cdot \bar{v}$, then $\sigma(w) \sim_{\mathscr{R}_\sigma} v \cdot w \cdot \bar{v}$ by Lemma F.3.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|w| = k$, then there exists a $v \in L_\sigma$ such that $\sigma(w) \sim_{\mathscr{G}_\sigma} v \cdot w \cdot \bar{v}$.
- **Inductive Step**. Assume that $|w| = k+1$. Then there exists some word $\hat{w}$ over $\mathscr{G}_n$ and some $M \in \mathscr{G}_n$ such that $w = \hat{w} \cdot M$ with $|\hat{w}| = k$. Clearly, M is either of type X, type (-1), or type K. In any case, there exists a word v over the two-level operators of type X such that $[\![v]\!]_O^* = [\![\sigma]\!]_S$ and $\sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot M \cdot \bar{v}$. Then $v \in L_\sigma$. By the inductive hypothesis, there exists a $u \in L_\sigma$ such that $\sigma(\hat{w}) \sim_{\mathscr{R}_\sigma} u \cdot \hat{w} \cdot \bar{u}$. Since $u \in L_\sigma$ and $v \in L_\sigma$, then $u \sim_{\mathscr{R}_\sigma} v$ by Property (1). Furthermore $\bar{v} \cdot v \sim_{\mathscr{R}_\sigma} \varepsilon$ by Lemma F.3. Then $\bar{v} \cdot u \sim_{\mathscr{R}_\sigma} \bar{v} \cdot v \sim_{\mathscr{R}_\sigma} \varepsilon$. Since $\sigma(w) = \sigma(\hat{w}) \cdot \sigma(M)$, then it follows $\sigma(w) \sim_{\mathscr{R}_\sigma} v \cdot \hat{w} \cdot \bar{v} \cdot \sigma(M) \sim_{\mathscr{R}_\sigma} v \cdot \hat{w} \cdot \bar{v} \cdot u \cdot M \cdot \bar{u} \sim_{\mathscr{R}_\sigma} v \cdot \hat{w} \cdot M \cdot \bar{u} \sim_{\mathscr{R}_\sigma} u \cdot w \cdot \bar{u}$ and the inductive step is established.

Then by the principle of induction, Property (2) holds. □

Corollary F.11. *Let v and w be words over $\mathscr{G}_n$. If σ is a valid reindexing for u and w, then $\sigma(w)$ is derivable from $\sigma(u)$ using $\mathscr{R}_\sigma \cup \{u \approx w\}$.*

Proof. Let $Q = \mathscr{R}_\sigma \cup \{u \approx w\}$. By Theorem F.10, there exists words v_1 and v_2 over two-level operators of type X, such that $v_1 \sim_Q v_2$, $\sigma(u) \sim_Q v_1 \cdot u \cdot \bar{v}_1$ and $\sigma(w) \sim_Q v_2 \cdot w \cdot \bar{v}_2$. Since $v_1 \sim_Q v_2$, then $[\![v_1]\!]_O^* = [\![v_2]\!]_O^*$. Then $[\![\bar{v}_1]\!]_O^* = [\![\bar{v}_2]\!]_O^*$. Since $\bar{v}_1$ and $\bar{v}_2$ are words over two-level operators of type X, then $\bar{v}_1 \sim_Q \bar{v}_2$ by Lemma F.2. Then the following derivation holds over Q.

$$\sigma(u) \to v_1 \cdot u \cdot \bar{v}_1 \to v_2 \cdot u \cdot \bar{v}_1 \to v_2 \cdot w \cdot \bar{v}_1 \to v_2 \cdot w \cdot \bar{v}_2 \to \sigma(w)$$

Then $\sigma(w)$ is derivable from $\sigma(w)$ using Q. □

F.3 The Set of Representative Relations

In Section 5.2, a set of representative relations were selected from $\mathscr{R}_n$. These relations are illustrated in Figure 10. In some sense, the choice of representative relations were arbitrary, since all choices are equivalent up to permutation. However, preference was given to the parameters [0], [4], [0,1,2,3], [4,5,6,7], since these correspond well to controlled qubit operators.

$$X_{[a,b]}{}^2 \approx \varepsilon \tag{109}$$

$$(-1)_{[0]}{}^2 \approx \varepsilon \tag{110}$$

$$K_{[0,1,2,3]}{}^2 \approx \varepsilon \tag{111}$$

$$X_{[a,b]} \cdot X_{[c,d]} \approx X_{[c,d]} \cdot X_{[a,b]} \tag{112}$$

$$X_{[a,b]} \cdot (-1)_{[c]} \approx (-1)_{[c]} \cdot X_{[a,b]} \tag{113}$$

$$X_{[a,b]} \cdot K_{[c,d,e,f]} \approx K_{[c,d,e,f]} \cdot X_{[a,b]} \tag{114}$$

$$(-1)_{[4]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot (-1)_{[4]} \tag{115}$$

$$(-1)_{[0]} \cdot (-1)_{[4]} \approx (-1)_{[4]} \cdot (-1)_{[0]} \tag{116}$$

$$K_{[0,1,2,3]} \cdot K_{[4,5,6,7]} \approx K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \tag{117}$$

$$X_{[a,a+2]} \cdot X_{[a,a+1]} \approx X_{[a+1,a+2]} \cdot X_{[a,a+2]} \tag{118}$$

$$X_{[a+1,a+2]} \cdot X_{[a,a+1]} \approx X_{[a,a+2]} \cdot X_{[a+1,a+2]} \tag{119}$$

$$X_{[a,b]} \cdot (-1)_{[a]} \approx (-1)_{[b]} \cdot X_{[a,b]} \tag{120}$$

$$X_{[a,e]} \cdot K_{[a,b,c,d]} \approx K_{[e,b,c,d]} \cdot X_{[a,e]} \tag{121}$$

$$X_{[b,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,e,c,d]} \cdot X_{[b,e]} \tag{122}$$

$$X_{[c,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,e,d]} \cdot X_{[c,e]} \tag{123}$$

$$X_{[d,e]} \cdot K_{[a,b,c,d]} \approx K_{[a,b,c,e]} \cdot X_{[d,e]} \tag{124}$$

$$X_{[0,1]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[0,1]} \cdot (-1)_{[1]} \cdot (-1)_{[3]} \tag{125}$$

$$X_{[1,2]} \cdot K_{[0,1,2,3]} \approx (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \cdot K_{[0,1,2,3]} \cdot (-1)_{[0]} \tag{126}$$

$$X_{[2,3]} \cdot K_{[0,1,2,3]} \approx K_{[0,1,2,3]} \cdot X_{[1,3]} \tag{127}$$

$$K_{[0,1,2,3]} \cdot K_{[1,3,4,5]} \approx K_{[1,3,4,5]} \cdot K_{[0,1,2,3]} \tag{128}$$

$$(-1)_{[0]} \cdot (-1)_{[4]} \cdot X_{[0,4]} \cdot \rho \approx \rho \cdot X_{[0,4]} \cdot (-1)_{[4]} \cdot (-1)_{[0]} \tag{129}$$

Figure 10: The representative relations in $\mathscr{R}_n^1$, for all valid choices of $a,b,c,d,e,f \in \mathbb{Z}$. We write ρ for the substring $K_{[4,5,6,7]} \cdot K_{[0,1,2,3]} \cdot X_{[3,4]} \cdot K_{[0,1,2,3]} \cdot K_{[4,5,6,7]} \cdot X_{[0,4]}$.

F.4 Proving the Redundant Relations are Derivable

This section makes use of the braiding relations and the inverse relations, to derive several bifunctoriality and commutator relations. Each proof follows the same structure. First, the special case is proven where all generators of type (-1) or K have consecutive indices starting from 0. In all other cases, there is a generator of type (-1) or K conjugated by a permutation. The braiding relations are used to obtain a convenient decomposition for each permutation. The commutativity and bifunctoriality follow immediately from these decompositions.

Lemma F.12. *If $(0,a,b)$ is an increasing sequence over $[n]$, then $X_{[a,b]} \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot X_{[a,b]}$.*

Proof. Let $\sigma = \tau_{a,b}$. Since $3 < a < b$, then there exists a decomposition $\tau_{c_1,c_1+1} \circ \tau_{c_2,c_2+1} \circ \cdots \tau_{c_m,c_m+1}$ of σ into transpositions such that $0 < c_k$ for all $k \in [n]$. Define $u = X_{[c_1,c_1+1]} \cdot X_{[c_2,c_2+1]} \cdots X_{[c_m,c_m+1]}$. Then $[\![X_{[a,b]}]\!]_O^* = [\![\sigma]\!]_S = [\![u]\!]_O^*$. Then $X_{[a,b]} \sim_{\mathscr{R}_n^3} u$ by Lemma F.2. The proof follows by induction on m.

- **Base Case**. If $|u| = 0$, then $u \cdot (-1)_{[0]} = (-1)_{[0]} \cdot u$. Then $u \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot u$ by reflexivity.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|u| = k$, then $u \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot u$.
- **Inductive Step**. Assume that $m = k+1$. Define $v = X_{[c_1,c_1+1]} \cdot X_{[c_2,c_2+1]} \cdots X_{[c_k,c_k+1]}$. Then by definition $u = v \cdot X_{[c_m,c_m+1]}$. Since $c_m > 0$, then $u \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} v \cdot (-1)_{[0]} \cdot X_{[c_m,c_m+1]}$ by the relation $X_{[c_m,c_m+1]} \cdot (-1)_{[0]} \approx_{\mathscr{R}_n^3} (-1)_{[0]} \cdot X_{[c_m,c_m+1]}$. Furthermore, since $|v| = k$, then by the inductive hypothesis $v \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot v$. Then $v \cdot (-1)_{[0]} \cdot X_{[c_m,c_m+1]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot u$. Then by the transitivity of $(\sim_{\mathscr{R}_n^3})$, $u \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot u$ and the inductive step is established.

Then $u \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot u$ by the principle of induction. Since $X_{[a,b]} \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} u \cdot (-1)_{[0]}$ and $(-1)_{[0]} \cdot u \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot X_{[a,b]}$, then $X_{[a,b]} \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot X_{[a,b]}$ by the transitivity of $(\sim_{\mathscr{R}_n^3})$. □

Theorem F.13. *All instances of Relation (51) are derivable from $\mathscr{R}_n^3$.*

Proof. Let $\{a,b,c\} \in [n]$. Define $\sigma \in \mathrm{S}(n)$ such that σ is τ_{k,c_k} if $c > 0$, or identity otherwise. Likewise, define u to be $X_{[0,c]}$ if $c > 0$, or ε otherwise. Clearly $\tau_{a,b} \circ \sigma = \sigma \circ \tau_{\sigma(a),\sigma(b)}$. Then,

$$[\![X_{[a,b]} \cdot u]\!]_O^* = [\![\tau_{a,b} \circ \sigma]\!]_S = [\![\sigma \circ \tau_{\sigma(a),\sigma(b)}]\!]_S = [\![u \cdot X_{[\sigma(a),\sigma(b)]}]\!]_O^*.$$

Then $X_{[a,b]} \cdot u \sim_{\mathscr{R}_n^3} u \cdot X_{[\sigma(a),\sigma(b)]}$ by Lemma F.2. Likewise, $X_{[\sigma(a),\sigma(b)]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} \overline{u} \cdot X_{[a,b]}$ by Lemma F.3. Since $\{a,b,c\}$ are distinct, then $\sigma(a) > 0$ and $\sigma(b) > 0$. Then $X_{[\sigma a,\sigma b]} \cdot (-1)_{[0]} \sim_{\mathscr{R}_n^3} (-1)_{[0]} \cdot X_{[\sigma a,\sigma b]}$ by Lemma F.12. Then the following derivation holds.

$$X_{[a,b]} \cdot u \cdot (-1)_{[0]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot X_{[\sigma(a),\sigma(b)]} \cdot (-1)_{[0]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot (-1)_{[0]} \cdot X_{[\sigma(a),\sigma(b)]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot (-1)_{[0]} \cdot \overline{u} \cdot X_{[a,b]}$$

Since $\{a,b,c\}$ were arbitrary, then all instances of Relation (51) are derivable from $\mathscr{R}_n^3$. □

Lemma F.14. *If $(3,a,b)$ is an increasing sequence over $[n]$, then $X_{[a,b]} \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[a,b]}$.*

Proof. Let $\sigma = \tau_{a,b}$. Since $3 < a < b$, then there exists a decomposition $\tau_{c_1,c_1+1} \circ \tau_{c_2,c_2+1} \circ \cdots \tau_{c_m,c_m+1}$ of σ into transpositions such that $3 < c_k$ for all $k \in [n]$. Define $u = X_{[c_1,c_1+1]} \cdot X_{[c_2,c_2+1]} \cdots X_{[c_m,c_m+1]}$. Then $[\![X_{[a,b]}]\!]_O^* = [\![\sigma]\!]_S = [\![u]\!]_O^*$. Then $X_{[a,b]} \sim_{\mathscr{R}_n^3} u$ by Lemma F.2. The proof follows by induction on m.

- **Base Case**. If $|u| = 0$, then $u \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot u$ by reflexivity.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|u| = k$, then $u \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot u$.
- **Inductive Step**. Assume that $m = k+1$. Define $v = X_{[c_1,c_1+1]} \cdot X_{[c_2,c_2+1]} \cdots X_{[c_k,c_k+1]}$. Then by definition $u = v \cdot X_{[c_m,c_m+1]}$. Since $c_m > 3$, then $u \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} v \cdot K_{[0,1,2,3]} \cdot X_{[c_m,c_m+1]}$ by the relation $X_{[c_m,c_m+1]} \cdot K_{[0,1,2,3]} \approx_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[c_m,c_m+1]}$. Since $|v| = k$, then by the inductive hypothesis $v \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot v$. Then $v \cdot K_{[0,1,2,3]} \cdot X_{[c_m,c_m+1]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot u$. Then by the transitivity of $(\sim_{\mathscr{R}_n^3})$, $u \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot u$ and the inductive step is established.

Then $u \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot u$ by the principle of induction. Since $X_{[a,b]} \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} u \cdot K_{[0,1,2,3]}$ and $K_{[0,1,2,3]} \cdot u \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[a,b]}$, then $X_{[a,b]} \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[a,b]}$ by the transitivity of $(\sim_{\mathscr{R}_n^3})$. □

Theorem F.15. *All instances of Relation (52) are derivable from $\mathscr{R}_n^3$.*

Proof. Let (c_0,c_1,c_2,c_3) an increasing sequence over $[n]$. Since (c_0,c_1,c_2,c_3) is increasing, then $k \leq c_k$ for $k \in [4]$. Then for each $k \in [4]$, define σ_k to be τ_{k,c_k} if $k \neq c_k$, or identity otherwise, and define $\sigma = \sigma_0 \circ \sigma_1 \circ \sigma_2 \circ \sigma_3$. Likewise, for each $k \in [4]$, define u_k to be $X_{[k,c_k]}$ if $k \neq c_k$, or ε otherwise, and let $u = u_0 \cdot u_1 \cdot u_2 \cdot u_3$. Clearly $\tau_{a,b} \circ \sigma = \sigma \circ \tau_{\sigma(a),\sigma(b)}$. Then,

$$[\![X_{[a,b]} \cdot u]\!]_O^* = [\![\tau_{a,b} \circ \sigma]\!]_S = [\![\sigma \circ \tau_{\sigma(a),\sigma(b)}]\!]_S = [\![u \cdot X_{[\sigma(a),\sigma(b)]}]\!]_O^*.$$

Then $X_{[a,b]} \cdot u \sim_{\mathscr{R}_n^3} u \cdot X_{[\sigma(a),\sigma(b)]}$ by Lemma F.2. Likewise, $X_{[\sigma(a),\sigma(b)]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} \overline{u} \cdot X_{[a,b]}$ by Lemma F.3. Since $a,b \notin \{c_0,c_1,c_2,c_3\}$, then $\sigma(a) > 3$ and $\sigma(b) > 3$. Then $X_{[\sigma a,\sigma b]} \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[\sigma a,\sigma b]}$ by Lemma F.14. Then the following derivation holds.

$$X_{[a,b]} \cdot u \cdot K_{[0,1,2,3]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot X_{[\sigma(a),\sigma(b)]} \cdot K_{[0,1,2,3]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot K_{[0,1,2,3]} \cdot X_{[\sigma(a),\sigma(b)]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} u \cdot K_{[0,1,2,3]} \cdot \overline{u} \cdot X_{[a,b]}$$

Since $\{a,b,c_0,c_1,c_2,c_3\}$ were arbitrary, then all instances of Relation (52) are derivable from $\mathscr{R}_n^3$. □

Lemma F.16. *Let (c_0,a,c_1,c_2,c_3) be an increasing sequence over $[n]$. For each $k \in [4]$, define σ_k to be τ_{k,c_k} if $k \neq c_k$, or identity otherwise. If $\sigma = \sigma_0 \cdot \sigma_1 \cdot \sigma_2 \cdot \sigma_3$ and $\rho = \tau_{0,a} \cdot \sigma_1 \cdot \sigma_2 \cdot \sigma_3$, then there exists an $\alpha \in \mathrm{S}(n)$ such that $\tau_{c_0,a} \circ \sigma = \rho \circ \alpha$ with α fixing $[4]$.*

Proof. By definition, $\rho(0) = a = \tau_{c_0,a}(\sigma(0))$, $\rho(1) = c_1 = \tau_{c_0,a}(\sigma(0))$, $\rho(1) = c_2 = \tau_{c_0,a}(\sigma(0))$, and $\rho(2) = c_2 = \tau_{c_0,a}(\sigma(0))$. Since $\mathrm{S}(n)$ is a group, then there exists an $\alpha \in \mathrm{S}(n)$ such that $\tau_{c_0,a} \circ \sigma = \rho \circ \alpha$. Assume that there exists a $k \in [4]$ such that $\alpha(k) \neq k$. Then $(\tau_{c_0,a} \circ \sigma)(\alpha(k)) \neq (\tau_{c_0,a} \circ \sigma)(k) = \rho(k)$. Then by contradiction, α fixes $[4]$. Then α decomposes into a sequence of transpositions over $[n] \setminus [4]$. □

Lemma F.17. *Let* $w = X_{[a_0,b_0]} \cdot X_{[a_1,b_1]} \cdots X_{[a_m,b_m]}$ *such that* $a_k > 3$ *and* $b_k > 3$ *for all* $k \in [m+1]$*. Then* $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$*.*

Proof. Let proof follows by induction on $|w|$.

- **Base Case**. If $|w| = 0$, then $w = \varepsilon$ and $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$ by the transitivity of $(\sim_{\mathscr{R}_n^3})$.
- **Inductive Hypothesis**. Assume that for some $k \in \mathbb{N}$, if $|w| = k$, then $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$
- **Inductive Step**. Assume that $|w| = k+1$. Define $v = X_{[a_0,b_0]} \cdot X_{[a_1,b_1]} \cdots X_{[a_k,b_k]}$. Then by definition $w = v \cdot X_{[a_m,b_m]}$. Since $a_m > 3$ and $b_m > 3$, then $X_{[a_m,b_m]} \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot X_{[a_m,b_m]}$ by Lemma F.14. As a result, $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} v \cdot K_{[0,1,2,3]} \cdot X_{[a_m,b_m]}$. Then by the inductive hypothesis, $v \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot v$. As a result, $v \cdot K_{[0,1,2,3]} \cdot X_{[a_m,b_m]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$. Then $w \cdot K_{[0,1,2,3]} \sim K_{[0,1,2,3]} \cdot w$ by the transitivity of $(\sim_{\mathscr{R}_n^3})$ and the inductive step is established.

Then by the principle of induction, $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$. □

Theorem F.18. *All instances of Relations (59), (60), (61) and (62) are derivable from* $\mathscr{R}_n^3$*.*

Proof. Let be (c_0, a, c_1, c_2, c_3) an increasing sequence. For each $k \in [4]$, define σ_k to be τ_{k,c_k} if $k \neq c_k$, or identity otherwise. Then define $\sigma = \sigma_0 \circ \sigma_1 \circ \sigma_2 \circ \sigma_3$ and $\rho = \tau_{0,a} \cdot \sigma_1 \cdot \sigma_2 \cdot \sigma_3$. By Lemma F.16, there exists a sequence of transpositions $\alpha = \alpha_0 \circ \alpha_1 \circ \cdots \alpha_m$ such that $\tau_{c_0,a} \circ \sigma = \rho \circ \alpha$. Then define $w = [\![\alpha_0]\!]_S \cdot [\![\alpha_1]\!]_S \cdots [\![\alpha_m]\!]_S$. Next, for each $k \in [4]$, define u_k to be $X_{[k,c_k]}$ if $k \neq c_k$, or ε otherwise. Then define $u = u_0 \cdot u_1 \cdot u_2 \cdot u_3$, $v = X_{[c_0,a]} \cdot u_1 \cdot u_2 \cdot u_3$, and $w = [\![\alpha_0]\!]_S \cdot [\![\alpha_1]\!]_S \cdots [\![\alpha_m]\!]_S$. It follows that $[\![u]\!]_O^* = [\![\tau_{c_0,a} \circ \sigma]\!]_S = [\![\rho \circ \alpha]\!]_S = [\![v \cdot w]\!]$. Since u, v, and w are words over two-level operators of type X, then $X_{[c_0,a]} \cdot u \sim_{\mathscr{R}_n^3} v \cdot w$. Likewise, by Lemma F.3, $\overline{u} \cdot X_{[c_0,a]} \sim_{\mathscr{R}_n^3} \overline{w} \cdot \overline{v}$. Then $X_{[c_0,a]} \cdot u$ acts by conjugation on $K_{[0,1,2,3]}$ as follows.

$$X_{[c_0,a]} \cdot u \cdot K_{[0,1,2,3]} \cdot \overline{u} \cdot X_{[c_0,1]} \sim_{\mathscr{R}_n^3} v \cdot w \cdot K_{[0,1,2,3]} \overline{u} \cdot X_{[c_0,1]} \sim_{\mathscr{R}_n^3} v \cdot w \cdot K_{[0,1,2,3]} \cdot \overline{w} \cdot \overline{v}$$

Then by Lemma F.17, $w \cdot K_{[0,1,2,3]} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w$. Furthermore, $w \cdot \overline{w} \sim_{\mathscr{R}_n^3} \varepsilon$ Lemma F.3. Then w acts by conjugation on $K_{[0,1,2,3]}$ as follows.

$$w \cdot K_{[0,1,2,3]} \cdot \overline{w} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]} \cdot w \cdot \overline{w} \sim_{\mathscr{R}_n^3} K_{[0,1,2,3]}$$

Since ${X_{[c_0,a]}}^2 \sim_{\mathscr{R}_n^3} \varepsilon$ by Lemma F.2, then the following derivation also holds.

$$X_{[c_0,a]} \cdot u \cdot K_{[0,1,2,3]} \cdot \overline{u} \sim_{\mathscr{R}_n^3} X_{[c_0,a]} \cdot u \cdot K_{[0,1,2,3]} \cdot \overline{u} \cdot {X_{[c_0,1]}}^2 \sim_{\mathscr{R}_n^3} v \cdot w \cdot K_{[0,1,2,3]} \cdot \overline{w} \cdot \overline{v} \cdot X_{[c_0,1]} \sim_{\mathscr{R}_n^3} v \cdot K_{[0,1,2,3]} \cdot \overline{v} \cdot X_{[c_0,1]}$$

Since $\{a, b, c_0, c_1, c_2, c_3\}$ were all arbitrary, then all instances of Relation (59) holds. The cases of Relations (60), (61) and (62) follow symmetrically. □

Exact Synthesis of Multiqutrit Clifford-Cyclotomic Circuits

Andrew N. Glaudell
Photonic Inc.
andrewglaudell@gmail.com

Neil J. Ross
Dalhousie University
neil.jr.ross@dal.ca

John van de Wetering
University of Amsterdam
john@vdwetering.name

Lia Yeh
University of Oxford
lia.yeh@cs.ox.ac.uk

It is known that the matrices that can be exactly represented by a multiqubit circuit over the Toffoli+Hadamard, Clifford+T, or, more generally, Clifford-cyclotomic gate set are precisely the unitary matrices with entries in the ring $\mathbb{Z}[1/2, \zeta_k]$, where k is a positive integer that depends on the gate set and ζ_k is a primitive 2^k-th root of unity. In the present paper, we establish an analogous correspondence for qutrits. We define the multiqutrit Clifford-cyclotomic gate set of degree 3^k by extending the classical qutrit gates X, CX, and CCX with the Hadamard gate H and the T_k gate $T_k = \text{diag}(1, \omega_k, \omega_k^2)$, where ω_k is a primitive 3^k-th root of unity. This gate set is equivalent to the qutrit Toffoli+Hadamard gate set when $k = 1$, and to the qutrit Clifford+T_k gate set when $k > 1$. We then prove that a $3^n \times 3^n$ unitary matrix U can be represented by an n-qutrit circuit over the Clifford-cyclotomic gate set of degree 3^k if and only if the entries of U lie in the ring $\mathbb{Z}[1/3, \omega_k]$.

1 Introduction

1.1 Background

In quantum computing, **synthesis** refers to the process of converting a representation of a unitary into a quantum circuit. In **exact synthesis** the unitary is typically given as a matrix, and the goal is to produce a circuit that implements the matrix exactly. This is in contrast to **approximate synthesis**, where the circuit is only required to implement the given matrix up to some prescribed error budget.

A solution to an exact synthesis problem for a gate set $\mathscr{G}$ sometimes characterizes the unitary matrices that can be exactly represented by a circuit over $\mathscr{G}$. For instance, the matrices with entries in the ring $\mathbb{Z}[1/2]$ of dyadic rationals corresponds precisely to the unitary matrices that can be represented using the Toffoli gate and the tensor product $H \otimes H$ of the Hadamard gate with itself [4]. Similarly, Clifford+T circuits correspond to unitary matrices with entries in $\mathbb{Z}[1/2, e^{2\pi i/8}]$ [12]. More generally, it was recently shown that multiqubit circuits over the Clifford-cyclotomic gate set of degree k, which extends the Clifford gate set with a z-rotation by angle $2\pi/2^k$, correspond to unitary matrices with entries the ring $\mathbb{Z}[1/2, e^{2\pi i/2^k}]$ [2].

In this paper, we consider the exact synthesis problem for qutrits. Like for qubits, fault-tolerant universal quantum computation has been theoretically devised for qutrits through magic state distillation [5, 9, 24] or gauge fixing of colour codes [30]. In recent years, qudit operations have been demonstrated on many experimental platforms [17, 20, 31, 33], with error rates competitive to qubit operations [26, 10]. Qutrit exact synthesis problems, however, have received less attention than their qubit counterparts and only a few results exist: a normal form for single-qutrit Clifford+T unitaries [13, 25], a proof that all classically reversible functions on trits can be implemented using Clifford+T circuits [32], and an exact synthesis result for single-qutrit Clifford+R unitaries [19].

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 44–62, doi:10.4204/EPTCS.406.2

Let k be a positive integer and let $\omega_k \in \mathbb{C}$ be the **primitive 3^k-th root of unity** $\omega_k = e^{2\pi i/3^k}$. For simplicity, we write ω for ω_1. The single-qutrit **Pauli** X gate, **Pauli** Z gate, **phase** gate S, and **Hadamard** gate H are defined below.

$$X = \begin{bmatrix} \cdot & \cdot & 1 \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & \omega & \cdot \\ \cdot & \cdot & \omega^2 \end{bmatrix} \qquad S = \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & \omega \end{bmatrix} \qquad H = \frac{-\omega^2}{\sqrt{-3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}$$

The two-qutrit **controlled-X** gate CX is the permutation matrix whose action on the computational basis is defined by $|i\rangle\,|j\rangle \mapsto |i\rangle\,|i+j\rangle$, with addition performed modulo 3. The three-qutrit **doubly-controlled-X** gate CCX (or **Toffoli** gate) is similarly defined by $|i\rangle\,|j\rangle\,|k\rangle \mapsto |i\rangle\,|j\rangle\,|k+ij\rangle$. The gate set $\{H,S,CX\}$ is the **Clifford** gate set. Now define the single-qutrit T_k gate

$$T_k = \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & \omega_k & \cdot \\ \cdot & \cdot & \omega_k^2 \end{bmatrix}.$$

When $k = 2$, T_k is the qutrit T gate [16].

The **Clifford-cyclotomic gate set of degree** 3^k is the gate set $\mathcal{G}_k = \{X, CX, CCX, H, T_k\}$. When $k = 1$, we have $T_1 = Z = HXH^\dagger$, so that the Clifford-cyclotomic gate set of degree 3 is equivalent to the qutrit **Toffoli+Hadamard** gate set [27]. As we will show below, when $k \geq 2$, the gate set $\mathcal{G}_k$ is equivalent (up to a single ancillary qutrit) to the **Clifford+T_k** gate set $\{H, S, CX, T_k\}$. In particular, the Clifford-cyclotomic gate set of degree 9 is equivalent to the well-known qutrit **Clifford+T** gate set [13, 14, 25, 32]. Because $T_{k+1}^3 = T_k$, the Clifford-cyclotomic gate sets form a hierarchy of universal gate sets whose first level is given by the Toffoli+Hadamard gate set, whose second level is given by the Clifford+T gate set, and whose subsequent levels are given by finer and finer extensions of the Clifford gate set.

Now consider the ring $\mathbb{Z}[1/3, \omega_k]$, which can be defined as the smallest unital subring of $\mathbb{C}$ containing $1/3$ and ω_k. Since $-\omega^2/\sqrt{-3} = \omega^2(1-\omega)/3$, the entries of X, CX, CCX, H, and T_k lie in $\mathbb{Z}[1/3, \omega_k]$. Hence, any n-qutrit circuit over $\mathcal{G}_k$ must represent a unitary matrix with entries in $\mathbb{Z}[1/3, \omega_k]$. The purpose of this paper is to show that the converse implication is also true.

1.2 Contributions

We show that a $3^n \times 3^n$ unitary matrix U can be exactly represented by an n-qutrit circuit over the Clifford-cyclotomic gate set of degree 3^k if and only if the entries of U belong to the ring $\mathbb{Z}[1/3, \omega_k]$. Furthermore, we show that no more than $k+1$ ancillae are required for this purpose.

We therefore solve the exact synthesis problem for multiqutrit Toffoli+Hadamard circuits, multiqutrit Clifford+T circuits, and, more generally, multiqutrit Clifford-cyclotomic circuits. To the best of our knowledge, this is the first time that a multiqudit exact synthesis result is established for any prime $d > 2$.

A similar hierarchy of Clifford-cyclotomic gate sets exists for qubits, and the correspondence between Clifford-cyclotomic circuits and matrices with entries in rings of algebraic integers also holds in that case [2]. Following [2], we prove our result inductively. We first show that circuits over $\mathcal{G}_1$ correspond to unitary matrices over $\mathbb{Z}[1/3, \omega]$ by reasoning as in [4, 12, 15]. This serves as the base case of our induction. Then, we use properties of the ring extension $\mathbb{Z}[1/3, \omega_k] \subseteq \mathbb{Z}[1/3, \omega_{k+1}]$ and the theory of catalytic embeddings [1] to establish the inductive step.

1.3 Contents

The paper is organized as follows. We discuss the necessary number-theoretic prerequisites in Section 2. In Section 3, we introduce a convenient generating set for the group $\mathrm{U}_n(\mathbb{Z}[1/3,\omega])$ of n-dimensional unitary matrices with entries in the ring $\mathbb{Z}[1/3,\omega]$, and in Section 4 we show that the elements of this generating set can be represented by Clifford-cyclotomic circuits of degree 3 (explicit circuit decompositions are given in Appendix A). We introduce catalytic embeddings in Section 5. We leverage the results of the previous sections in Section 6 to prove our main result. We comment on the complexity of the produced circuits in Section 7 and we conclude in Section 8.

Disclaimer: After the present work was completed, it was brought to our attention that related results were independently established in [18].

2 Rings and Groups

In this section, we discuss the rings and groups which will be important in the rest of the paper. In what follows, when u, u', and v are elements of a ring R, we write $u \equiv_v u'$ if u is congruent to u' modulo v, i.e., if $u - u' = rv$ for some $r \in R$.

2.1 The Ring $\mathbb{Z}[\omega_k]$

Definition 2.1. Let $k \geq 1$. The **primitive 3^k-th root of unity** $\omega_k \in \mathbb{C}$ is defined as $\omega_k = e^{2\pi i/3^k}$.

We have, for $k > 1$, $\omega_k^3 = \omega_{k-1}$, $\omega_k^{3^k} = 1$, $\omega_k^\dagger = \omega_k^{3^k-1}$, and $\omega_k^0 + \omega_k^1 + \ldots + \omega_k^{3^k-1} = 0$. As mentioned in Section 1, we often write ω for ω_1.

Definition 2.2. Let $k \geq 1$. The ring $\mathbb{Z}[\omega_k]$ of **cyclotomic integers of degree** 3^k is the smallest subring of $\mathbb{C}$ that contains ω_k.

The ring $\mathbb{Z}[\omega_k]$ can be defined in a variety of ways [29]. It will be useful for our purposes to note that $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$, and that, for $k \geq 2$,

$$\mathbb{Z}[\omega_k] = \{a + b\omega_k + c\omega_k^2 \mid a, b, c \in \mathbb{Z}[\omega_{k-1}]\}.$$

Furthermore, the expression of an element of $\mathbb{Z}[\omega_k]$ as a linear combination of elements of $\mathbb{Z}[\omega_{k-1}]$ is unique. The ring $\mathbb{Z}[\omega_k]$ is closed under complex conjugation and, for $k \geq 2$, we have $\mathbb{Z}[\omega_{k-1}] \subseteq \mathbb{Z}[\omega_k]$.

2.2 Properties of $\mathbb{Z}[\omega]$

We now record some useful properties of $\mathbb{Z}[\omega]$. If $u = a + b\omega \in \mathbb{Z}[\omega]$, then

$$u^\dagger u = (a + b\omega)(a + b\omega^2) = a^2 + ab(\omega + \omega^2) + b^2 = a^2 - ab + b^2. \tag{1}$$

In particular, if $u \in \mathbb{Z}[\omega]$, then $u^\dagger u$ is a nonnegative integer, since the Euclidean norm of a complex number is always nonnegative.

Definition 2.3. We define $\lambda \in \mathbb{Z}[\omega]$ as $\lambda = 1 - \omega$.

By Equation (1), we have $\lambda^\dagger \lambda = 3$. Similarly, we have $\lambda^2 = 1 - 2\omega + \omega^2 = -3\omega$, so that $3 = -\lambda^2\omega^2$. Hence, $3 \equiv_\lambda 0$.

Proposition 2.4. *We have*

- $\mathbb{Z}[\omega]/(3) \cong \{0,1,2,\omega,2\omega,1+\omega,1+2\omega,2+\omega,2+2\omega\} \cong \mathbb{Z}/(3)+\omega\mathbb{Z}/(3)$ *and*
- $\mathbb{Z}[\omega]/(\lambda) \cong \{0,1,2\} \cong \mathbb{Z}/(3)$.

Proof. The first item follows from the fact that $3 \equiv_3 0$. The second item follows from the fact that $3 \equiv_\lambda 0$ and the fact that $\omega \equiv_\lambda 1$. □

Proposition 2.5. *If $u \in \mathbb{Z}[\omega]$, then $u^\dagger u \equiv_\lambda 0$ or $u^\dagger u \equiv_\lambda 1$.*

Proof. Let $u = a+b\omega \in \mathbb{Z}[\omega]$. By Proposition 2.4, $\mathbb{Z}[\omega]/(\lambda) \cong \mathbb{Z}/(3)$. By Equation (1),

$$u^\dagger u = a^2 - ab + b^2 \equiv_\lambda a^2 + 2ab + b^2 = (a+b)^2.$$

Hence $u^\dagger u$ is a square modulo λ and therefore cannot be congruent to 2, since 0 and 1 are the only squares in $\mathbb{Z}/(3)$. □

Proposition 2.6. *If $u \in \mathbb{Z}[\omega]$, then $u \not\equiv_\lambda 0$ if and only if $u \equiv_3 \pm\omega^x$ for some $x \in \{0,1,2\}$.*

Proof. The table below lists the elements of $\mathbb{Z}[\omega]/(3)$ as given by Proposition 2.4, together with their residues modulo λ.

$\mathbb{Z}[\omega]/(3)$	$\mathbb{Z}[\omega]/(\lambda)$
0	0
1	1
2	2
ω	1
2ω	2
$1+\omega$	2
$1+2\omega$	0
$2+\omega$	0
$2+2\omega$	1

The statement then follows by inspection of the table, using the fact that $1+\omega = -\omega^2 \equiv_3 -\omega^2$ and $2 \equiv_3 -1$. □

2.3 Denominators

Definition 2.7. Let $k \geq 1$. The ring $\mathbb{Z}[1/3,\omega_k]$ is defined as $\mathbb{Z}[1/3,\omega_k] = \{u/3^\ell \mid u \in \mathbb{Z}[\omega_k] \text{ and } \ell \in \mathbb{N}\}$.

Because the elements of $\mathbb{Z}[\omega_k]$ can be expressed as linear combinations of elements of $\mathbb{Z}[\omega_{k-1}]$, the elements of $\mathbb{Z}[1/3,\omega_k]$ can similarly be expressed as linear combinations of elements of $\mathbb{Z}[1/3,\omega_{k-1}]$. In particular, for $k \geq 2$, every element u of $\mathbb{Z}[1/3,\omega_k]$ can be uniquely written as $u = a+b\omega_k+c\omega_k^2$ with $a,b,c \in \mathbb{Z}[1/3,\omega_{k-1}]$.

The ring $\mathbb{Z}[1/3,\omega_k]$ is the localization of $\mathbb{Z}[\omega_k]$ by the powers of 3. Alternatively, $\mathbb{Z}[1/3,\omega_k]$ can be thought of as the localization of $\mathbb{Z}[\omega_k]$ by the powers of λ. Indeed, since $3 = -\omega^2\lambda^2$, we have $3^{-\ell} = (-\omega^2\lambda^2)^{-\ell} = (-\omega)^\ell(\lambda)^{-2\ell}$. As a result, any element of $\mathbb{Z}[1/3,\omega_k]$ can be written as u/λ^ℓ for some $u \in \mathbb{Z}[\omega_k]$ and some $\ell \in \mathbb{N}$. We leverage this fact to define, in the usual way (see [4, 12, 15]), the notions of **λ-denominator exponent** and **least λ-denominator exponent**.

Definition 2.8. Any nonnegative integer ℓ such that $v \in \mathbb{Z}[1/3,\omega_k]$ can be written as $v = u/\lambda^\ell$ with $u \in \mathbb{Z}[\omega_k]$ is **λ-denominator exponent** of v. The smallest such ℓ is the **least λ-denominator exponent** of v and is denoted $\operatorname{lde}(v)$.

The notions of denominator exponent and least denominator exponent extend to matrices (and therefore to vectors) with entries in $\mathbb{Z}[1/3,\omega_k]$: an integer ℓ is a λ-denominator exponent of a matrix M if it is a λ-denominator exponent of all of the entries of M; the smallest such ℓ is the least λ-denominator exponent of M.

2.4 The Group $\mathrm{U}_n(\mathbb{Z}[1/3,\omega_k])$

Definition 2.9. We write $\mathrm{U}_n(\mathbb{Z}[1/3,\omega_k])$ for the group of n-dimensional unitary matrices with entries in $\mathbb{Z}[1/3,\omega_k]$ and $\mathrm{U}(\mathbb{Z}[1/3,\omega_k])$ for the collection of all unitary matrices with entries in $\mathbb{Z}[1/3,\omega_k]$.

3 Generators for $\mathrm{U}_n(\mathbb{Z}[1/3,\omega])$

Following [4, 12, 15, 23], we use ***m*-level matrices** to define a subset of $\mathrm{U}_n(\mathbb{Z}[1/3,\omega])$ which we will show to be a generating set.

Definition 3.1. The matrices (-1), (ω), X, and H are defined as follows:

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

Definition 3.2. Let M be an $m \times m$ matrix, let $m \le n$, and let $0 \le x_1 < \ldots < x_m \le n-1$. The ***m*-level matrix** $M_{[x_1,\ldots,x_m]}$ is the $n \times n$ matrix whose entries are given as follows

$$M_{[x_1,\ldots,x_m]\,i,j} = \begin{cases} M_{i',j'} & \text{if } i = x_{i'} \text{ and } j = x_{j'},\\ I_{i,j} & \text{otherwise.}\end{cases}$$

For example, for $n = 4$, we have

$$(\omega)_{[1]} = \begin{bmatrix}1 & \cdot & \cdot & \cdot\\ \cdot & \omega & \cdot & \cdot\\ \cdot & \cdot & 1 & \cdot\\ \cdot & \cdot & \cdot & 1\end{bmatrix} \quad \text{and} \quad H_{[0,2,3]} = \frac{-\omega^2}{\lambda}\begin{bmatrix}1 & \cdot & 1 & 1\\ \cdot & \lambda/(-\omega^2) & \cdot & \cdot\\ 1 & \cdot & \omega & \omega^2\\ 1 & \cdot & \omega^2 & \omega\end{bmatrix}.$$

When applied to a vector $|u\rangle$, the matrix $(\omega)_{[1]}$ acts as (ω) on the entry of index 1 and the matrix $H_{[0,2,3]}$ acts as H on the entries of index 0, 2, and 3.

Definition 3.3. We write $\mathscr{S}_n$ for the subset of $\mathrm{U}_n(\mathbb{Z}[1/3,\omega])$ defined as

$$\mathscr{S}_n = \{(-1)_{[x]}, (\omega)_{[x]}, X_{[x,y]}, H_{[x,y,z]} \mid 0 \le x < y < z \le n-1\}.$$

Lemma 3.4. *Let $u_0,u_1,u_2 \in \mathbb{Z}[\omega]$ be such that $u_0 \not\equiv_\lambda 0$, $u_1 \not\equiv_\lambda 0$, and $u_2 \not\equiv_\lambda 0$. Then there exists $x_0,x_1,x_2 \in \{0,1,2\}$ and $y_0,y_1,y_2 \in \{0,1\}$ such that*

$$H(\omega)^{x_0}_{[0]}(\omega)^{x_1}_{[1]}(\omega)^{x_2}_{[2]}(-1)^{y_0}_{[0]}(-1)^{y_1}_{[1]}(-1)^{y_2}_{[2]}\begin{bmatrix}u_0\\ u_1\\ u_2\end{bmatrix} = \begin{bmatrix}u_0'\\ u_1'\\ u_2'\end{bmatrix}$$

for some $u_0',u_1',u_2' \in \mathbb{Z}[\omega]$ such that $u_0' \equiv_\lambda 0$, $u_1' \equiv_\lambda 0$, and $u_2' \equiv_\lambda 0$.

Proof. Let $j \in \{0,1,2\}$. Since $u_j \not\equiv_\lambda 0$, we have, by Proposition 2.6, $u_j \equiv_3 (-1)^{w_j}(\omega)^{z_j}$. Hence, setting $y_j = -w_j$ and $x_j = -z_j$, we get $(\omega)^{x_j}(-1)^{y_j}u_j \equiv_3 1$. Therefore,

$$(\omega)^{x_0}_{[0]}(\omega)^{x_1}_{[1]}(\omega)^{x_2}_{[2]}(-1)^{y_0}_{[0]}(-1)^{y_1}_{[1]}(-1)^{y_2}_{[2]}\begin{bmatrix}u_0\\u_1\\u_2\end{bmatrix} = \begin{bmatrix}v_0\\v_1\\v_2\end{bmatrix}$$

for some $v_0, v_1, v_2 \in \mathbb{Z}[\omega]$ such that $v_0 \equiv_3 v_1 \equiv_3 v_2 \equiv_3 1$. The result then follows by computation, since $v_0 + v_1 + v_2 \equiv_3 v_0 + \omega v_1 + \omega^2 v_2 \equiv_3 v_0 + \omega^2 v_1 + \omega v_2 \equiv_3 0$. □

Lemma 3.5. *Let* $|u\rangle \in \mathbb{Z}[1/3,\omega]^n$ *be a unit vector. If* $\operatorname{lde}|u\rangle = 0$*, then* $|u\rangle = \pm\omega^x|j\rangle$ *for some* $0 \le x \le 2$ *and some* $0 \le j \le n-1$.

Proof. Let $|u\rangle \in \mathbb{Z}[1/3,\omega]^n$. Since $\operatorname{lde}|u\rangle = 0$, we have $|u\rangle \in \mathbb{Z}[\omega]^n$. Since $|u\rangle$ is a unit vector, we also have

$$1 = \langle u|u\rangle = \sum u_j^\dagger u_j$$

with $u_j \in \mathbb{Z}[\omega]$. Because each $u_j^\dagger u_j$ is a nonnegative integer, there must be exactly one j for which $u_j^\dagger u_j = 1$, while $u_{j'}^\dagger u_{j'} = 0$ for all $j' \neq j$. If $u_j^\dagger u_j = 1$ then $a_j^2 - a_j b_j + b_j^2 = 1$, and this equation can only be true if $a = \pm 1$ and $b = 0$, $a = 0$ and $b = \pm 1$, or $a = b = \pm 1$. In the first case, $|u\rangle = \pm|j\rangle$, in the second case, $|u\rangle = \pm\omega|j\rangle$, and in the third case, $|u\rangle = \pm\omega^2|j\rangle$, □

Lemma 3.6. *Let* $|u\rangle \in \mathbb{Z}[1/3,\omega]^n$ *be a unit vector. If* $\operatorname{lde}|u\rangle > 0$*, then there exists* $G_0,\ldots,G_q \in \mathscr{S}_n$ *such that* $\operatorname{lde}(G_q\cdots G_0|u\rangle) < \operatorname{lde}|u\rangle$.

Proof. Write $|u\rangle$ as $|v\rangle/\lambda^\ell$, with $\ell = \operatorname{lde}|u\rangle$. Since $\langle u|u\rangle = 1$ and $\lambda^\dagger\lambda = 3$, we get $3^\ell = \langle v|v\rangle = \sum v_j^\dagger v_j$. Hence, $\sum v_j^\dagger v_j \equiv_\lambda 0$. By Proposition 2.5, $v_j^\dagger v_j$ is either 0 or 1 modulo λ, and by Proposition 2.4, $\mathbb{Z}[\omega]/(\lambda) \cong \mathbb{Z}/(3)$. Thus, the number of v_j such that $v_j \not\equiv_\lambda 0$ must be a multiple of 3. Hence, we can group the entries of $|v\rangle$ into triples and apply Lemma 3.4 to each such triple. This maps $|u\rangle$ to some $|u\rangle'$ of lower least denominator exponent. □

Lemma 3.7. *Let* $|u\rangle \in \mathbb{Z}[1/3,\omega]^n$ *be a unit vector and let* $0 \le j \le n-1$*. Then there exists* $G_0,\ldots,G_q \in \mathscr{S}_n$ *such that* $G_q\cdots G_0|u\rangle = |j\rangle$.

Proof. By induction on $\operatorname{lde}|u\rangle$. If $\operatorname{lde}(|u\rangle) = 0$, then, by Lemma 3.5, $|u\rangle = \pm\omega^x e_{j'}$ for some $0 \le j' \le n-1$ and some $0 \le x \le 2$. We can therefore reduce $|u\rangle$ to $|j\rangle$ by applying $(-1)_{[j']}$, $(\omega)_{[j']}$, and $X_{[j,j']}$ or $X_{[j',j]}$, as needed. If $\operatorname{lde}|u\rangle > 0$, then, by Lemma 3.6, there exists $G_p,\ldots,G_0 \in \mathscr{S}_n$ such that $\operatorname{lde}(G_p\cdots G_0|u\rangle) < \operatorname{lde}(|u\rangle)$. We can then conclude by applying the induction hypothesis to $G_p\cdots G_0|u\rangle$. □

Proposition 3.8. *Let* U *be an* $n\times n$ *matrix. Then* $U \in \mathrm{U}_n(\mathbb{Z}[1/3,\omega])$ *if and only if* U *can be written as a product of elements of* $\mathscr{S}_n$.

Proof. The right-to-left direction is immediate. For the left-to-right direction, consider the matrix $U^\dagger \in \mathrm{U}_n(\mathbb{Z}[1/3,\omega])$. Iteratively applying Lemma 3.7 to the columns of $U^\dagger$ yields a sequence $G_0,\ldots,G_q$ of elements of $\mathscr{S}_n$ such that

$$G_0G_1\cdots GqU^\dagger = I,$$

and we can therefore write U as $U = G_0G_1\cdots G_q$. □

4 Exact Synthesis of Toffoli+Hadamard Circuits

Let $\mathscr{G}$ be a set of quantum gates. A unitary matrix U can be **represented by a circuit over** $\mathscr{G}$ if there exists a circuit C over $\mathscr{G}$ such that, for any state $|u\rangle$, we have $C|u\rangle = U|u\rangle$. The circuit may use ancillary qutrits, but these must start and end the computation in the same state. If that state can be arbitrary, the ancillary qutrits are said to be **borrowed**; if that state is required to be $|0\rangle$, the ancillary qutrits are said to be **fresh**. Unless otherwise specified, ancillae are assumed to be fresh. Note that if a matrix can be represented by a circuit using m borrowed ancillae, then it can also be represented by a circuit using m fresh ancillae.

Recall from Section 1 that the Clifford-cyclotomic gate set $\mathscr{G}_k$ is defined as $\mathscr{G}_k = \{X, CX, CCX, H, T_k\}$. In Appendix A we prove that $\mathscr{G}_1$ is equivalent to the Toffoli+Hadamard gate set, up to two borrowed ancillae and that, when $k \geq 2$, $\mathscr{G}_k$ is equivalent to the Clifford+T_k gate set $\{H, S, CX, T_k\}$, up to a single borrowed ancilla. The next proposition shows that all of the elements of $\mathscr{S}_{3^n}$ can be represented by a circuit over $\mathscr{G}_1$ using no more than 2 borrowed ancillae. The proof of the proposition can be found in Appendix A.

Proposition 4.1. *If $U \in \mathscr{S}_{3^n}$, then U can be represented by a circuit over $\mathscr{G}_1$ using at most 2 borrowed ancillae.*

Using Proposition 4.1 we are now in a position to define an exact synthesis algorithm for multiqutrit Toffoli+Hadamard circuits.

Theorem 4.2. *If $U \in \mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_1])$, then U can be represented by an n-qutrit circuit over $\mathscr{G}_1$ using at most 2 ancillae.*

Proof. By Proposition 3.8, $\mathscr{S}_{3^n}$ generates $\mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_1])$. Hence, it is sufficient to show that the elements of $\mathscr{S}_{3^n}$ can be represented by an n-qutrit circuit over $\mathscr{G}_1$. This follows from Proposition 4.1, since if 2 borrowed ancillae suffice to construct a circuit for U, then 2 fresh ancillae are also sufficient for this purpose. □

5 Catalytic Embeddings

Definition 5.1. Let $\mathscr{U}$ and $\mathscr{V}$ be collections of unitaries. An m**-dimensional catalytic embedding** of $\mathscr{U}$ into $\mathscr{V}$ is a pair $(\phi, |c\rangle)$ of a function $\phi : \mathscr{U} \to \mathscr{V}$ and a vector $|c\rangle \in \mathbb{C}^m$ such that if $U \in \mathscr{U}$ has dimension n then $\phi(U) \in \mathscr{V}$ has dimension nm and

$$\phi(U)(|u\rangle \otimes |c\rangle) = (U|u\rangle) \otimes |c\rangle$$

for every $|u\rangle \in \mathbb{C}^n$. The vector $|c\rangle$ is the **catalyst** of the catalytic embedding $(\phi, |c\rangle)$. We sometimes express the fact that $(\phi, |c\rangle)$ is a catalytic embedding of $\mathscr{U}$ into $\mathscr{V}$ by writing $(\phi, |c\rangle) : \mathscr{U} \to \mathscr{V}$.

Definition 5.2. Let $(\phi, |c\rangle) : \mathscr{U} \to \mathscr{V}$ and $(\phi', |c\rangle') : \mathscr{V} \to \mathscr{W}$ be catalytic embeddings of dimension m and m', respectively. The **concatenation** of $(\phi, |c\rangle)$ and $(\phi', |c\rangle')$ is the $m'm$-dimensional catalytic embedding $(\phi', |c\rangle') \circ (\phi, |c\rangle)$ defined by $(\phi', |c\rangle') \circ (\phi, |c\rangle) = (\phi' \circ \phi, |c\rangle \otimes |c\rangle')$.

The concatenation of catalytic embeddings is associative and the catalytic embedding $(1_{\mathscr{U}}, [1]) : \mathscr{U} \to \mathscr{U}$ acts as an identity for concatenation.

Definition 5.3. Let $k \geq 2$. We define Ω_k and $|c_k\rangle$ as

$$\Omega_k = \begin{bmatrix} \cdot & \cdot & \omega_{k-1} \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix} \qquad \text{and} \qquad |c_k\rangle = \frac{1}{\lambda} \begin{bmatrix} 1 \\ \omega_k^{-1} \\ \omega_k^{-2} \end{bmatrix}.$$

The matrix Ω_k is unitary and the vector $|c_k\rangle$ is an eigenvector of Ω_k for eigenvalue ω_k. Indeed, since $\omega_{k-1} = \omega_k^3$, we have

$$\Omega_k\,|c_k\rangle = \frac{1}{\lambda}\begin{bmatrix} \cdot & \cdot & \omega_{k-1} \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix}\begin{bmatrix} 1 \\ \omega_k^{-1} \\ \omega_k^{-2} \end{bmatrix} = \frac{1}{\lambda}\begin{bmatrix} \omega_k \\ 1 \\ \omega_k^{-1} \end{bmatrix} = \frac{\omega_k}{\lambda}\begin{bmatrix} 1 \\ \omega_k^{-1} \\ \omega_k^{-2} \end{bmatrix} = \omega_k\,|c_k\rangle\,. \tag{2}$$

Now recall from Section 2.3 that, for $k \geq 2$, every $u \in \mathbb{Z}[1/3, \omega_k]$ can be written uniquely as a linear combination of the form $u = a + b\omega_k + c\omega_k^2$, where $a, b, c \in \mathbb{Z}[1/3, \omega_{k-1}]$. Therefore, every matrix U over $\mathbb{Z}[1/3, \omega_k]$ can be uniquely written as $U = A + B\omega_k + C\omega_k^2$, where A, B, and C are matrices over $\mathbb{Z}[1/3, \omega_{k-1}]$. We use this fact below to define a function $\mathrm{U}(\mathbb{Z}[1/3, \omega_k]) \to \mathrm{U}(\mathbb{Z}[1/3, \omega_{k-1}])$.

Proposition 5.4. *Let $k \geq 2$. The assignment*

$$A + B\omega_k + C\omega_k^2 \longmapsto A \otimes I + B \otimes \Omega_k + C \otimes \Omega_k^2$$

defines a function $\phi_k : \mathrm{U}(\mathbb{Z}[1/3, \omega_k]) \to \mathrm{U}(\mathbb{Z}[1/3, \omega_{k-1}])$.

Proof. Let $U \in \mathrm{U}(\mathbb{Z}[1/3, \omega_k])$ and write U as $U = A + B\omega_k + C\omega_k^2$ for some matrices A, B, and C over $\mathbb{Z}[1/3, \omega_{k-1}]$. Now let $U' = A \otimes I + B \otimes \Omega_k + C \otimes \Omega_k^2$. It is clear that U' is a matrix with entries in $\mathbb{Z}[1/3, \omega_{k-1}]$. We now show that U' is unitary. Since U is unitary and since $U = A + B\omega_k + C\omega_k^2$, we can express the equation $U^\dagger U = I$ in terms of A, B, and C. Using $\omega_k^\dagger = \omega_{k-1}^\dagger \omega_k^2$, this yields

$$(A^\dagger A + B^\dagger B + C^\dagger C) + (A^\dagger B + B^\dagger C + C^\dagger A\omega_{k-1}^\dagger)\omega_k + (A^\dagger C + B^\dagger A\omega_{k-1}^\dagger + C^\dagger B\omega_{k-1}^\dagger)\omega_k^2 = I.$$

Hence, $A^\dagger A + B^\dagger B + C^\dagger C = I$ and $A^\dagger B + B^\dagger C + C^\dagger A\omega_{k-1}^\dagger = A^\dagger C + B^\dagger A\omega_{k-1}^\dagger + C^\dagger B\omega_{k-1}^\dagger = 0$. Now note that $\Omega_k^\dagger = \omega_{k-1}^\dagger \Omega_k^2$, so that $U'^\dagger U'$ is equal to

$$(A^\dagger A + B^\dagger B + C^\dagger C) \otimes I + (A^\dagger B + B^\dagger C + C^\dagger A\omega_{k-1}^\dagger) \otimes \Omega_k + (A^\dagger C + B^\dagger A\omega_{k-1}^\dagger + C^\dagger B\omega_{k-1}^\dagger) \otimes \Omega_k^2.$$

Hence, $U'^\dagger U' = I$. Reasoning analogously shows that $U'U'^\dagger = I$, so that U' is indeed unitary. □

Proposition 5.5. *Let $k \geq 2$. The pair $(\phi_k, |c_k\rangle)$ is a 3-dimensional catalytic embedding of* $\mathrm{U}(\mathbb{Z}[1/3, \omega_k])$ *into* $\mathrm{U}(\mathbb{Z}[1/3, \omega_{k-1}])$.

Proof. By Proposition 5.4, $\phi_k : \mathrm{U}(\mathbb{Z}[1/3, \omega_k]) \to \mathrm{U}(\mathbb{Z}[1/3, \omega_{k-1}])$ is a function and, by construction, if $U \in \mathrm{U}(\mathbb{Z}[1/3, \omega_k])$ has dimension n, then $\phi_k(U)$ has dimension $3n$. Moreover, if $|u\rangle \in \mathbb{C}^n$, then

$$\begin{aligned}
\phi_k(U)(|u\rangle \otimes |c_k\rangle) &= (A \otimes I + B \otimes \Omega_k + C \otimes \Omega_k^2)(|u\rangle \otimes |c_k\rangle) \\
&= A \otimes I(|u\rangle \otimes |c_k\rangle) + B \otimes \Omega_k(|u\rangle \otimes |c_k\rangle) + C \otimes \Omega_k^2(|u\rangle \otimes |c_k\rangle) \\
&= A\,|u\rangle \otimes I\,|c_k\rangle + B\,|u\rangle \otimes \Omega_k\,|c_k\rangle + C\,|u\rangle \otimes \Omega_k^2\,|c_k\rangle \\
&= A\,|u\rangle \otimes |c_k\rangle + B\,|u\rangle \otimes \omega_k\,|c_k\rangle + C\,|u\rangle \otimes \omega_k^2\,|c_k\rangle \\
&= A\,|u\rangle \otimes |c_k\rangle + \omega_k B\,|u\rangle \otimes |c_k\rangle + \omega_k^2 C\,|u\rangle \otimes |c_k\rangle \\
&= (A\,|u\rangle + \omega_k B\,|u\rangle + \omega_k^2 C\,|u\rangle) \otimes |c_k\rangle \\
&= (U\,|u\rangle) \otimes |c_k\rangle\,.
\end{aligned}$$

Hence, $(\phi_k, |c_k\rangle)$ is a catalytic embedding. □

Remark 5.6. The catalytic embedding constructed in Proposition 5.4 and Proposition 5.5 takes advantage of the fact that the matrix Ω_k and the algebraic number ω_k have many properties in common. Importantly, the polynomial $x^3 - \omega_{k-1}$ is both the characteristic polynomial of Ω_k and the minimal polynomial of ω_k over the ring $\mathbb{Z}[1/3, \omega_{k-1}]$. This construction generalizes to many other rings of interest (see [1]).

Corollary 5.7. *Let $k \geq 2$. There is a 3^{k-1}-dimensional catalytic embedding $(\phi, |c\rangle) : \mathrm{U}(\mathbb{Z}[1/3, \omega_k]) \to \mathrm{U}(\mathbb{Z}[1/3, \omega])$.*

Proof. Applying Proposition 5.5 repeatedly yields a sequence of catalytic embeddings

$$\mathrm{U}(\mathbb{Z}[1/3,\omega_k]) \xrightarrow{(\phi_k,|c_k\rangle)} \cdots \xrightarrow{(\phi_3,|c_3\rangle)} \mathrm{U}(\mathbb{Z}[1/3,\omega_2]) \xrightarrow{(\phi_2,|c_2\rangle)} \mathrm{U}(\mathbb{Z}[1/3,\omega]).$$

Concatenating the catalytic embeddings in this sequence yields the desired result. □

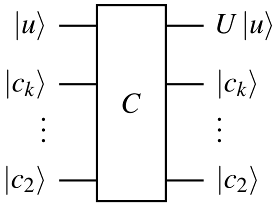
Note that the catalyst $|c\rangle$ in the catalytic embedding $(\phi, |c\rangle)$ of Corollary 5.7 is the product state $|c\rangle = |c_2\rangle \otimes \cdots \otimes |c_k\rangle$.

6 Exact Synthesis of Clifford-Cyclotomic Circuits

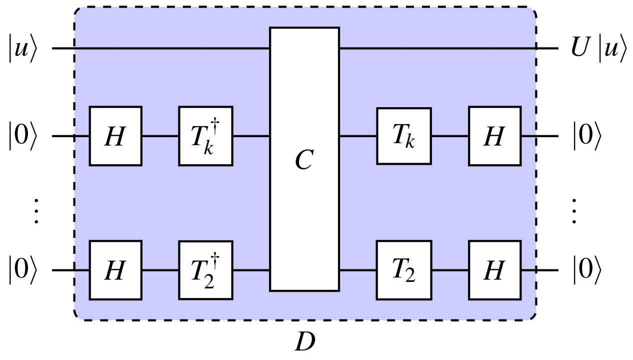
We can now prove our main result, which will follow straightforwardly from the results of Sections 3, 4 and 5.

Proposition 6.1. *Let $k \geq 2$. If $U \in \mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_k])$, then U can be represented by an n-qutrit circuit over $\mathscr{G}_k$ using at most $k+1$ ancillae.*

Proof. Let $U \in \mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_k])$ and let $(\phi, |c\rangle)$ be the catalytic embedding constructed in Corollary 5.7, with $|c\rangle = |c_2\rangle \otimes \cdots \otimes |c_k\rangle$. We then have $\phi(U) \in \mathrm{U}_{3^{n+k-1}}(\mathbb{Z}[1/3, \omega])$, so that, by Theorem 4.2, $\phi(U)$ can be represented by an $(n+k-1)$-qutrit circuit C over $\mathscr{G}_1$ using at most 2 fresh ancillae. By Definition 5.1, the action of $\phi(U)$ on an input of the form $|u\rangle \otimes |c_2\rangle \otimes \cdots \otimes |c_k\rangle$ can be depicted as below (where the ancillary qutrits used in C, if any, are omitted).



But, for $2 \leq \ell \leq k$, we have $|c_\ell\rangle = T_\ell^\dagger H |0\rangle$ and $T_\ell^\dagger = (T_k^\dagger)^{3^{k-\ell}}$. Hence, we can construct the following circuit over $\mathscr{G}_k$.



Since all of the ancillae in D (including the ancillae potentially present in C) start and end the computation in the $|0\rangle$ state, then D is a circuit over $\mathscr{G}_k$ which represents U and uses at most $k+1$ (fresh) ancillae, as desired. □

Remark 6.2. The circuit constructed in Proposition 6.1 actually use $k-1$ fresh ancillae and no more than 2 borrowed ancillae. For brevity, we simply stated the proposition in terms of fresh ancillae. One can amend the constructions in Appendix A to reduce the total ancilla-count from $k+1$ to k, at the cost of requiring all ancillae to be fresh.

Theorem 6.3. *Let $k \geq 1$ and let U be a $3^n \times 3^n$ unitary matrix. Then U can be represented by an n-qutrit circuit over $\mathscr{G}_k$ if and only if $U \in \mathrm{U}_{3^n}(\mathbb{Z}[1/3,\omega_k])$. Moreover, $k+1$ ancillae are always sufficient to construct a circuit for U.*

Proof. The left-to-right direction is a consequence of the fact that the entries of the elements of $\mathscr{G}_k$ lie in the ring $\mathbb{Z}[1/3,\omega_k]$. The right-to-left direction is given by Theorem 4.2 and Proposition 6.1. □

7 Circuit Complexity

The proof of Theorem 6.3 is constructive: it provides an algorithm to construct a circuit for a given matrix. In this section, we briefly discuss the complexity of the resulting circuit, reasoning as in [3, 12]. We start by considering Proposition 3.8 before turning to Theorem 6.3.

Lemma 7.1. *Let $U \in \mathrm{U}_m(\mathbb{Z}[1/3,\omega])$ and let $\ell = \mathrm{lde}(U)$. The algorithm of Proposition 3.8 expresses U as a product of $O(2^m \ell)$ elements of $\mathscr{S}_m$ in the worst case.*

Proof. Consider the first column of U. In the worst case, its least denominator exponent is ℓ. To reduce this least denominator exponent by one requires $O(m)$ operations. Hence, reducing the first column of U completely requires $O(\ell m)$ operations in the worst case. The reduction of the first column may increase the least denominator exponent of the second column from ℓ to 2ℓ, since each entry of the second column may be affected by up to ℓ 3-level matrices in the course of this first reduction. Once the first column has been reduced, the second column may still have $m-1$ nonzero entries. Reducing the second column will hence require $O(2\ell(m-1))$ operations in the worst case. In general, reducing the j-th column will require $O(2^{j-1}\ell(m-j))$ operations in the worst case so that the overall reduction of U requires at most

$$O\left(\sum_{i=0}^{n-1} 2^i \ell(m-i)\right)$$

operations. Simplifying the resulting sum yields a total of $O(2^m \ell)$ operations. □

Theorem 7.2. *Let $U \in \mathrm{U}_{3^n}(\mathbb{Z}[1/3,\omega_k])$ and let $\ell = \mathrm{lde}(U)$. The algorithm of Theorem 6.3 represents U as a circuit of $O((n+k)2^{3^{n+k-1}}\ell)$ gates in the worst case.*

Proof. The algorithm of Theorem 6.3 uses the catalytic embedding $(\phi, |c\rangle)$ of Corollary 5.7 to construct a matrix $\phi(U)$ over $\mathbb{Z}[1/3,\omega]$. The dimension of $\phi(U)$ is 3^{n+k-1} and its least denominator exponent is no more than ℓ. Hence, by Lemma 7.1, the algorithm of Proposition 3.8 will express $\phi(U)$ as a product of no more than $O(2^{3^{n+k-1}}\ell)$ elements of $\mathscr{S}_{3^{n+k-1}}$. It follows from the circuit constructions given in Appendix A, that each element of $\mathscr{S}_{3^{n+k-1}}$ can be represented by a circuit consisting of $O(n+k)$ gates. Hence, the circuit produced by Theorem 6.3 consists of no more than $O((n+k)2^{3^{n+k-1}}\ell)$ gates. □

8 Conclusion

We showed that the matrices that can be exactly represented by an n-qutrit circuit over the Clifford-cyclotomic gate set of degree 3^k are precisely the elements of $\mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_k])$. Moreover, we showed that no more than $k+1$ ancillae are required to construct a circuit for an element of $\mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_k])$.

Our proof contains an algorithm for synthesizing a circuit over $\mathscr{G}_k$, given a matrix in $\mathrm{U}_{3^n}(\mathbb{Z}[1/3, \omega_k])$. However, the circuits constructed in this way are very large and their optimization is a promising direction for future research. It would be interesting to reduce the gate-complexity of the circuits produced by Theorem 6.3. The techniques employed in [3, 22] for the synthesis of multiqubit Toffoli+Hadamard and Clifford+T circuits are likely to apply in the qutrit context as well. Similarly, it would also be interesting to reduce the number of ancillae used by the algorithm. As Appendix A shows, some of the ancillae can be removed by choosing a slightly different gate set, but the bulk of the ancillae come from the use of catalytic embeddings, so a different synthesis technique may be required for more significant savings. Along this line of inquiry, it would be interesting to characterize the matrices that can be represented by ancilla-free circuits. Such characterizations exist for qubit matrices [4, 12], but are likely to be different for qutrits [32].

Finally, a natural generalization of this work would be to consider higher-dimensional qudits. However, preliminary research suggests that the techniques used here and in [2] might not adapt straightforwardly to primes larger than 3. While it stands to reason that some version of our results should continue to hold for larger prime dimensions, proving this to be the case might require new ideas.

Acknowledgements

The authors would like to thank Sarah Meng Li, Ewan Murphy, and the anonymous reviewers of 21st International Conference on Quantum Physics and Logic (QPL 2024) for insightful comments on an earlier version of this paper. LY is funded by a Google PhD Fellowship. The circuit diagrams in the proof of Proposition 6.1 were typeset using Quantikz [21].

References

[1] Matthew Amy, Matthew Crawford, Andrew N. Glaudell, Melissa L. Macasieb, Samuel S. Mendelson & Neil J. Ross (2023): *Catalytic embeddings of quantum circuits*. Preprint available from arXiv:2305.07720.

[2] Matthew Amy, Andrew N. Glaudell, Shaun Kelso, William Maxwell, Samuel S. Mendelson & Neil J. Ross (2023): *Exact Synthesis of Multiqubit Clifford-Cyclotomic Circuits*. Preprint available from arXiv:2311.07741.

[3] Matthew Amy, Andrew N. Glaudell, Sarah Meng Li & Neil J. Ross (2023): *Improved Synthesis of Toffoli-Hadamard Circuits*. In Martin Kutrib & Uwe Meyer, editors: *Reversible Computation*, Springer Nature Switzerland, Cham, pp. 169–209, doi:10.1007/978-3-031-38100-3_12. Also available from arXiv:2305.11305.

[4] 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. Also available from arXiv:1908.06076.

[5] 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. Also available from arXiv:1202.2326.

[6] Alex Bocharov (2016): *A Note on Optimality of Quantum Circuits over Metaplectic Basis*. *Quantum Information and Computation* 18, doi:10.26421/QIC18.1-2-1. Also available from arXiv:1606.02315.

[7] Alex Bocharov, Shawn Cui, Martin Roetteler & Krysta Svore (2016): *Improved Quantum Ternary Arithmetics*. Quantum Information and Computation 16, pp. 862–884, doi:10.26421/QIC16.9-10-8. Also available from arXiv:1512.03824.

[8] 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. Also available from arXiv:1605.02756.

[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. Also available from arXiv:1205.3104.

[10] Yulin Chi, Jieshan Huang, Zhanchuan Zhang, Jun Mao, Zinan Zhou, Xiaojiong Chen, Chonghao Zhai, Jueming Bao, Tianxiang Dai, Huihong Yuan, Ming Zhang, Daoxin Dai, Bo Tang, Yan Yang, Zhihua Li, Yunhong Ding, Leif K. Oxenløwe, Mark G. Thompson, Jeremy L. O'Brien, Yan Li, Qihuang Gong & Jianwei Wang (2022): *A programmable qudit-based quantum processor*. Nature Communications 13(1), p. 1166, doi:10.1038/s41467-022-28767-x.

[11] Shawn X. Cui & Zhenghan Wang (2015): *Universal quantum computation with metaplectic anyons*. Journal of Mathematical Physics 56(3), p. 032202, doi:10.1063/1.4914941. Also available from arXiv:1405.7778.

[12] 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.

[13] Andrew N. Glaudell, Neil J. Ross & Jacob M. Taylor (2019): *Canonical forms for single-qutrit Clifford+T operators*. Annals of Physics 406, pp. 54–70, doi:10.1016/j.aop.2019.04.001. Also available from arXiv:1803.05047.

[14] Andrew N. Glaudell, Neil J. Ross, John van de Wetering & Lia Yeh (2022): *Qutrit Metaplectic Gates Are a Subset of Clifford+T*. 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. 12:1–12:15, doi:10.4230/LIPIcs.TQC.2022.12. Also available from arXiv:2202.09235.

[15] Seth Evenson Murray Greylyn (2014): *Generators and relations for the group* $\mathrm{U}_4(\mathbb{Z}[1/\sqrt{2}, i])$. Master's thesis, Dalhousie University. Available from arXiv:1408.6204.

[16] Mark Howard & Jiri Vala (2012): *Qudit versions of the qubit $\pi/8$ gate*. Phys. Rev. A 86, p. 022316, doi:10.1103/PhysRevA.86.022316. Also available from arXiv:1206.1598.

[17] Pavel Hrmo, Benjamin Wilhelm, Lukas Gerster, Martin W. van Mourik, Marcus Huber, Rainer Blatt, Philipp Schindler, Thomas Monz & Martin Ringbauer (2023): *Native qudit entanglement in a trapped ion quantum processor*. Nature Communications 14(1), p. 2242, doi:10.1038/s41467-023-37375-2. Also available from arXiv:2206.04104.

[18] Amolak Ratan Kalra, Manimugdha Saikia, Dinesh Valluri, Sam Winnick & Jon Yard (2024): *Multi-qutrit exact synthesis*. Preprint available from arXiv:2405.08147.

[19] Amolak Ratan Kalra, Dinesh Valluri & Michele Mosca (2024): *Synthesis and Arithmetic of Single Qutrit Circuits*. Preprint available from arXiv:2311.08696.

[20] Valentin Kasper, Daniel González-Cuadra, Apoorva Hegde, Andy Xia, Alexandre Dauphin, Felix Huber, Eberhard Tiemann, Maciej Lewenstein, Fred Jendrzejewski & Philipp Hauke (2021): *Universal quantum computation and quantum error correction with ultracold atomic mixtures*. Quantum Science and Technology 7(1), p. 015008, doi:10.1088/2058-9565/ac2d39. Also available from arXiv:2010.15923.

[21] Alastair Kay (2018): *Tutorial on the quantikz package*. Preprint available from arXiv:1809.03842.

[22] Vadym Kliuchnikov (2013): *Synthesis of unitaries with Clifford+T circuits*. Preprint available from arXiv:1306.3200.

[23] Michael A. Nielsen & Isaac L. Chuang (2000): *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences, Cambridge University Press, doi:10.1017/CBO9780511976667.

[24] Shiroman Prakash (2020): *Magic state distillation with the ternary Golay code*. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476(2241), p. 20200187, doi:10.1098/rspa.2020.0187. Also available from arXiv:2003.02717.

[25] Shiroman Prakash, Akalank Jain, Bhakti Kapur & Shubangi Seth (2018): *Normal form for single-qutrit Clifford+T operators and synthesis of single-qutrit gates*. *Physical Review A* 98, p. 032304, doi:10.1103/PhysRevA.98.032304. Available from arXiv:1803.05047.

[26] Martin Ringbauer, Thomas R. Bromley, Marco Cianciaruso, Ludovico Lami, W. Y. Sarah Lau, Gerardo Adesso, Andrew G. White, Alessandro Fedrizzi & Marco Piani (2018): *Certification and Quantification of Multilevel Quantum Coherence*. *Phys. Rev. X* 8, p. 041007, doi:10.1103/PhysRevX.8.041007. Also available from arXiv:1707.05282.

[27] Patrick Roy, John van de Wetering & Lia Yeh (2023): *The Qudit ZH-Calculus: Generalised Toffoli+Hadamard and Universality*. *Electronic Proceedings in Theoretical Computer Science* 384, pp. 142–170, doi:10.4204/eptcs.384.9. Also available from arXiv:2307.10095.

[28] Peter Selinger (2016): *Reversible k-valued logic circuits are finitely generated for odd k*. Available from arXiv:1604.01646.

[29] L. C. Washington (1982): *Introduction to Cyclotomic Fields*. Springer New York, NY, doi:10.1007/978-1-4612-1934-7.

[30] Fern H. E. Watson, Earl T. Campbell, Hussain Anwar & Dan E. Browne (2015): *Qudit color codes and gauge color codes in all spatial dimensions*. *Phys. Rev. A* 92, p. 022312, doi:10.1103/PhysRevA.92.022312. Also available from arXiv:1503.08800.

[31] Jordi R. Weggemans, Alexander Urech, Alexander Rausch, Robert Spreeuw, Richard Boucherie, Florian Schreck, Kareljan Schoutens, Jiří Minář & Florian Speelman (2022): *Solving correlation clustering with QAOA and a Rydberg qudit system: a full-stack approach*. *Quantum* 6, p. 687, doi:10.22331/q-2022-04-13-687. Also available from arXiv:2106.11672v3.

[32] Lia Yeh & John van de Wetering (2022): *Constructing all qutrit controlled Clifford+T gates in Clifford+T*. In Claudio Antares Mezzina & Krzysztof Podlaski, editors: *Reversible Computation*, Springer International Publishing, Cham, pp. 28–50, doi:10.1007/978-3-031-09005-9_3. Also available from arXiv:2204.00552.

[33] 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. Also available from arXiv:2003.04879.

[34] Wei Zi, Qian Li & Xiaoming Sun (2023): *Optimal Synthesis of Multi-Controlled Qudit Gates*. In: *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, doi:10.1109/DAC56929.2023.10247925. Also available from arXiv:2303.12979.

A Circuit Constructions

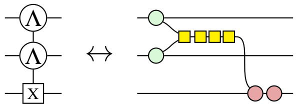
In this appendix, we show that the Clifford-cyclotomic gate set $\mathscr{G}_k$ is equivalent to the Clifford+T_k gate set when $k \geq 2$, we give a construction of the CX gate over the $\{X, CCX, H\}$ gate set, and we provide a proof of Proposition 4.1. In addition, we show that the matrices $(-1)_{[x]}$, $(\omega_k)_{[x]}$, $X_{[x_1,x_2]}$, and $H_{[x_1,x_2,x_3]}$ can be represented by circuits over the $\mathscr{G}_k$ gate set using at most k borrowed ancillae. The constructions in this appendix are exact (i.e., not up to a global or relative phase). Implementations of our constructions, for a fixed number of controls, are available at https://github.com/lia-approves/qutrit-Clifford-cyclotomic.

A.1 Gate Set Equivalences

Recall from Section 1 that the qutrit Toffoli (or CCX) gate acts on computational basis states as

$$|x,y,z\rangle \mapsto |x,y,z+xy\rangle,$$

where the arithmetic operations are performed modulo 3. In higher prime dimension d, the Toffoli gate is defined similarly, except that the arithmetic operations are performed modulo d. The Toffoli gate can be represented in the qupit ZH-calculus [27] as below.



(3)

In Equation (3), Λ denotes the following type of control: if U is a unitary and $|c\rangle$ and $|t\rangle$ are computational basis states, then $\Lambda(U)|c\rangle|t\rangle = |c\rangle \otimes (U^c|t\rangle)$. In particular, $\Lambda(X)$ is the CX gate and $\Lambda(\Lambda(X)) = \Lambda(CX)$ is the CCX gate.

We now recall the definition of the $|0\rangle$**-controlled** X gate, which applies an X gate to its target if and only if its control is in the state $|0\rangle$ [27].

Definition A.1. Let d be a prime. The qudit $|0\rangle$**-controlled** X gate acts on computational basis states as

$$|c,t\rangle \mapsto \begin{cases} |c,t+1\rangle & \text{if } c = 0, \text{ and} \\ |c,t\rangle & \text{otherwise,} \end{cases}$$

where arithmetic is performed modulo d.

Remarkably, when d is a prime greater than 2, the X gate and the $|0\rangle$-controlled X gate suffices to generate all of the d-ary classical reversible gates [27]. Moreover, as was shown in [28, 32], when $d = 3$, no ancillary qutrits are needed for this purpose. In contrast, there is no collection of reversible one and two-qubit gates that suffices to generate all of the binary reversible gates.

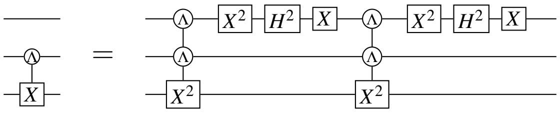
Theorem A.2 ([32], Theorem 2)**.** *Any ternary classical reversible function $f : \{0,1,2\}^n \to \{0,1,2\}^n$ can be represented by an ancilla-free circuit of X and $|0\rangle$-controlled X gates.*

Here, we only need multiply-controlled Toffoli gates, which can be built with a gate count linear in the number of controls, as in [27, 34]. The constructions of [27, 34] use no more borrowed ancillae than there are controls. They can be made into ancilla-free constructions by building Toffoli gates with $n/2$ controls using at most $n/2$ borrowed ancillae. Following [32], one can then combine six of these Toffoli gates with $n/2$ controls to construct a Toffoli gate with $n-1$ controls, and then combine 3 of these Toffoli gates with $n-1$ controls to add the final control.

We now show that the CX gate can be represented by a circuit over $\{X, CCX, H\}$.

Lemma A.3. *The gate sets $\{X, CX, CCX, H\}$ and $\{X, CCX, H\}$ are equivalent up to a single borrowed ancilla.*

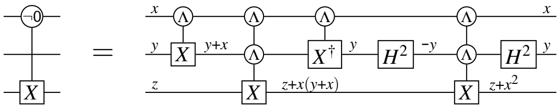
Proof. The circuit below represents the CX gate using a single borrowed ancilla.



□

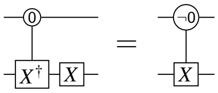
Proposition A.4. *Let $C_{|0\rangle}X$ denote the qutrit $|0\rangle$-controlled X gate. Then the gate sets $\{X, CCX, H\}$ and $\{X, C_{|0\rangle}X, H\}$ are equivalent up to a single borrowed ancilla.*

Proof. The gates X and CCX are ternary classical reversible functions. Hence, by Theorem A.2, they can both be represented by a circuit over $\{X, C_{|0\rangle}X, H\}$. Thus, every matrix that can be represented by a circuit over $\{X, CCX, H\}$ can be represented by a circuit over $\{X, C_{|0\rangle}X, H\}$. Conversely, we have



(4)

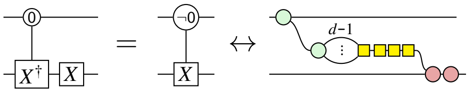
where $x, y, z \in \{0, 1, 2\}$ are input qutrit computational basis states and the basis state on a wire is updated whenever it is changed by the circuit. The $\neg 0$ on the left-hand side of Equation (4) indicates that the X gate is applied when the control is not in the state $|0\rangle$. To see that Equation (4) holds, note that $x^2 = 1$ for $x \neq 0$ so that $z + x^2$ is indeed the desired state. Moreover, we have



(5)

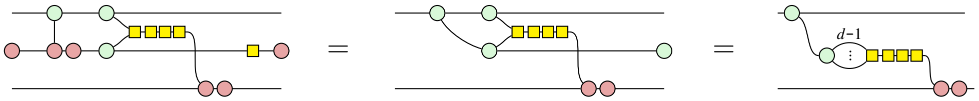
Therefore, multiplying the inverse of the circuit on the right-hand side of Equation (4) by an X gate yields a representation of the $C_{|0\rangle}X$ over the gate $\{X, CCX, H\}$ by Lemma A.3. Hence, every matrix that can be represented by a circuit over $\{X, C_{|0\rangle}X, H\}$ can be represented by a circuit over $\{X, CCX, H\}$ using a single borrowed ancilla. □

Remark A.5. The construction in Proposition A.4 can be explained (and, in fact, was found) using the qupit ZH-calculus [27]. In the qupit ZH-calculus, we have



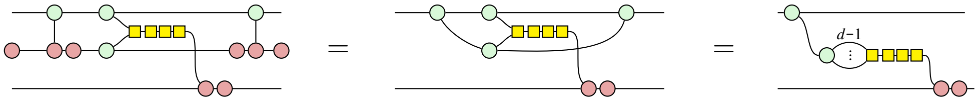
(6)

where the $d-1$ label indicates there are $d-1$ number of wires in parallel. We then get the following construction of the $|\neg 0\rangle$-controlled X gate:



(7)

The post-selected circuit in Equation (7) can be made deterministic by adding a $CX^\dagger$ gate for uncomputation, which yields a construction requiring a fresh ancilla:



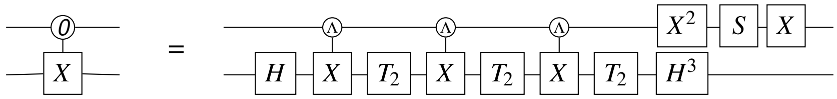
(8)

The construction is then modified in order to work with a borrowed ancilla, which yields the circuit in Equation (4).

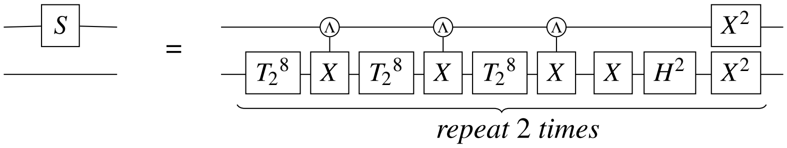
By Lemma A.3 and Proposition A.4, the gate set $\mathcal{G}_1$ is equivalent (up to a borrowed ancilla) to the gate set consisting of the X gate, the $|0\rangle$-controlled X gate, and the Hadamard gate. Hence, by Theorem A.2, any ternary classical reversible function can be represented by a circuit over $\mathcal{G}_1$ using at most one borrowed ancilla.

We now show that, when $k \geq 2$, the Clifford-cyclotomic gate set of degree 3^k is equivalent, up to a borrowed ancilla, to the Clifford+T_k gate set. We take advantage of some constructions from [8] (see, in particular, Figure 6 in [8]).

Lemma A.6. *We have:*



Lemma A.7. *We have:*



Proposition A.8. *When $k \geq 2$, the Clifford-cyclotomic gate set $\mathcal{G}_k$ is equivalent to the Clifford+T_k gate set up to a single borrowed ancilla.*

Proof. Recall that $\mathcal{G}_k = \{X, CX, CCX, H, T_k\}$ and that Clifford+$T_k = \{H, S, CX, T_k\}$. To prove the proposition, we therefore need to show that the S gate can be represented by a circuit over $\mathcal{G}_k$ and that the X and CCX gates can be represented by Clifford+T_k circuits. That the S gate can be represented by a circuit over $\mathcal{G}_k$ follows from Lemma A.7 and the fact that $T_2 = T_k^{3^{k-2}}$. That the X can be represented by a Clifford+T_k circuit simply follows from the fact that $X = H^\dagger T_2^3 H$. That the CCX gate can be represented by a Clifford+T_k circuit follows from Lemma A.6 and Theorem A.2. □

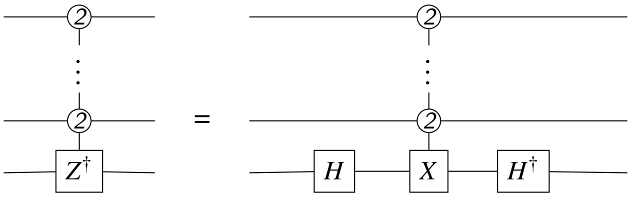
The propositions above show that there is some flexibility in the definition of Clifford-cyclotomic gate sets and, in particular, that the gate set $\{X, CX, CCX, H, T_k\}$ is by no means minimal.

A.2 Circuit Representations for the Elements of $\mathscr{S}_{3^n}$

We now provide explicit constructions for the elements of $\mathscr{S}_{3^n}$. We focus on the matrices in $\mathscr{S}_{3^n}$ where, writing each computational basis state on n qutrits as n trits, the levels are chosen to be those with the greatest value (taking the last qutrit to have the least significant trit). Indeed, these constructions can then be adapted to arbitrary levels by conjugating them by ternary classical reversible circuits using Theorem A.2 and Proposition A.4.

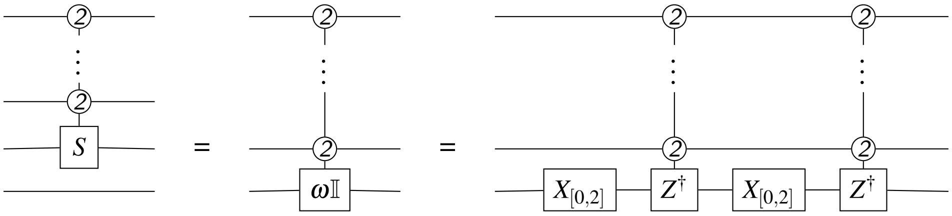
By Theorem A.2 and Proposition A.4, the multiply-controlled X gate can be expressed as a circuit over $\mathcal{G}_1$ using a single borrowed ancilla. We can therefore express the multiply-controlled Z gate as well, since $Z^\dagger = HXH^\dagger$.

Lemma A.9. *We have:*



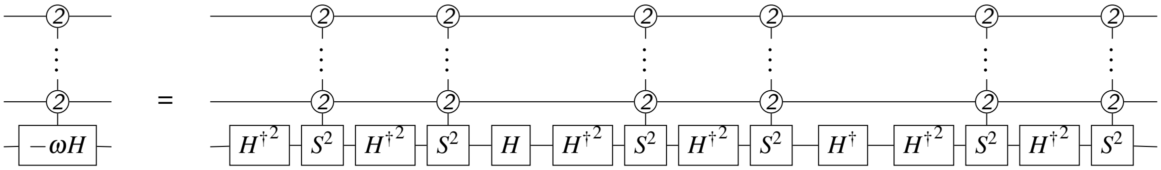
From this and the fact that $(\omega)_{[2]} = S$ when acting on a single qutrit, we can construct the 1-level matrix $(\omega)_{[x]}$ using a single borrowed ancilla.

Lemma A.10. *We have:*

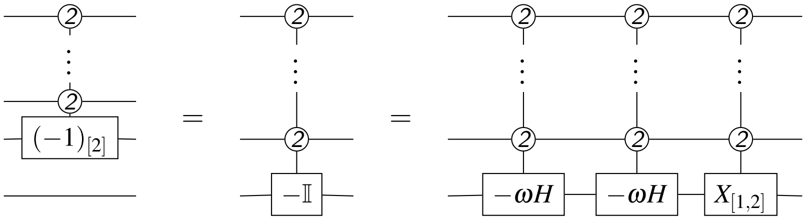


The next two lemmas let us construct the 1-level matrix $(-1)_{[x]}$. When acting on a single qutrit, this is the $(-1)_{[2]} = \mathrm{diag}(1,1,-1)$ gate. This gate is also known as the **metaplectic** gate [6, 8, 11] and in earlier work, we referred to this gate as the R gate [14].

Lemma A.11. *We have:*

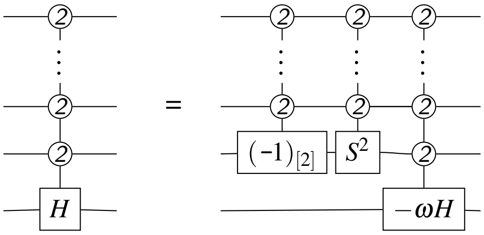


Lemma A.12. *We have:*



We can now synthesize the 3-level matrix $H_{[x_1,x_2,x_3]}$ matrix over $\mathscr{G}_1$. To do this, apply Lemma A.11 as well as the appropriate controlled global phase correction: a product of 1-level $\omega_{[x]}$ matrices and $(-1)_{[x]}$ matrices.

Lemma A.13. *We have:*



We have now constructed all of the required 1-, 2-, and 3-level matrices (up to a permutation). We can therefore prove Proposition 4.1, which we restate below, making the ancilla requirements explicit.

Proposition. *If $U \in \mathscr{S}_{3^n}$, then U can be represented by a circuit over $\mathscr{G}_1$ using at most 2 borrowed ancillae. Explicitly,*

- $(-1)_{[x]}$ *requires 2 borrowed ancillae,*
- $(\omega)_{[x]}$ *requires 1 borrowed ancillae,*
- $X_{[x_1,x_2]}$ *requires 1 borrowed ancilla, and*
- $H_{[x_1,x_2,x_3]}$ *requires 1 borrowed ancillae.*

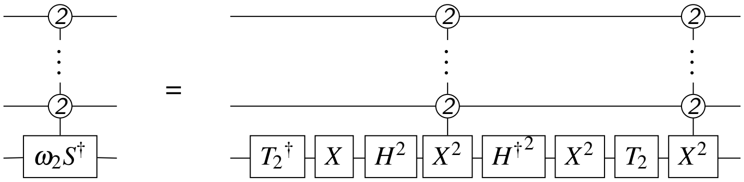
Proof. This follows from Lemmas A.3, A.10, A.12 and A.13, Proposition A.4, and Theorem A.2. □

The number of ancillae required to represent the elements of $\mathscr{S}_{3^n}$ is, to a certain extent, an artifact of the choice of gate set. For example, including the $|0\rangle$-controlled X gate to the gate set would lower the ancilla-count for some of the elements of $\mathscr{S}_{3^n}$.

The proposition above shows that the matrices that can be represented by a multiqutrit circuit over the Clifford+$(-1)_{[2]}$ gate set (also known as the **Clifford+R** or the **metaplectic** gate set) are a subset of those representable by a circuit over $\mathscr{G}_1$. At the time of writing, we do not know whether this inclusion is strict, although the conjecture in [7] that not all ternary classical reversible gates can be exactly represented over the Clifford+$(-1)_{[2]}$ gate set lends credence to this idea.

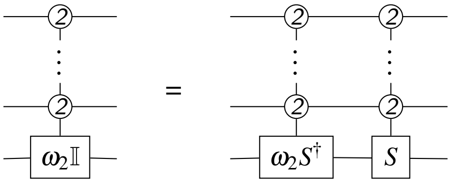
If a matrix can be represented by a circuit over $\mathscr{G}_k$, it can also be represented by a circuit over $\mathscr{G}_{k+1}$. It therefore follows from the proposition above that all of the elements of $\mathscr{S}_{3^n}$ can be represented by a circuit over $\mathscr{G}_2$. We close this appendix by showing that the 1-level matrix $(\omega_2)_{[x]}$ can be represented by a circuit over $\mathscr{G}_2$ and by providing further generalizations of the above constructions. This paves the way for a direct proof of exact synthesis for Clifford+T circuits (rather than the more indirect one using catalytic embeddings, as in Theorem 6.3). Over $\mathscr{G}_2$, the ancilla requirements are lowered, since the $|0\rangle$-controlled X gate can be represented by an ancilla-free circuit by Lemma A.6. To construct $(\omega_2)_{[x]}$, we first build a modification of $(\omega)_{[x]}$ which differs by a controlled global phase of ω_2.

Lemma A.14. *We have:*



We note that unlike the construction in Lemma A.10 which required one (additional) borrowed ancilla, this construction requires no (additional) borrowed ancillae. By combining the construction of Lemma A.14 and that of Lemma A.13, we can therefore represent $H_{[x_1,x_2,x_3]}$ without ancillae. Similarly, by combining the construction of Lemma A.14 and that of Lemma A.12, we can represent $(-1)_{[x]}$ using a single borrowed ancillae. Finally, $(\omega_2)_{[x]}$ can be constructed as in the next lemma using 2 borrowed ancillae.

Lemma A.15. *We have:*



Proposition A.16. *The 1-, 2-, and 3-level matrices $(-1)_{[x]}$, $(\omega_2)_{[x]}$, $X_{[x_1,x_2]}$, and $H_{[x_1,x_2,x_3]}$ can be represented by a circuit over $\mathscr{G}_2$ using at most 2 borrowed ancillae. Explicitly,*

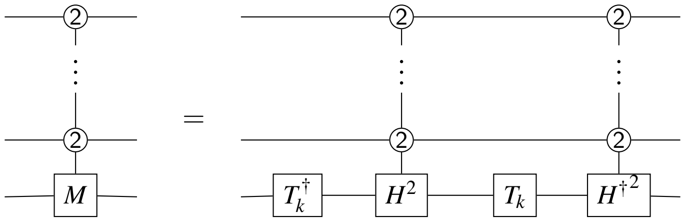
- *$(-1)_{[x]}$ requires 1 borrowed ancilla,*
- *$(\omega_2)_{[x]}$ requires 2 borrowed ancillae,*
- *$X_{[x_1,x_2]}$ requires 0 borrowed ancillae, and*
- *$H_{[x_1,x_2,x_3]}$ requires 0 borrowed ancillae.*

Proof. This follows from Lemmas A.6, A.12, A.13, A.14 and A.15 and Theorem A.2. □

We can generalize the above construction to Clifford-cyclotomic gate sets of higher degree.

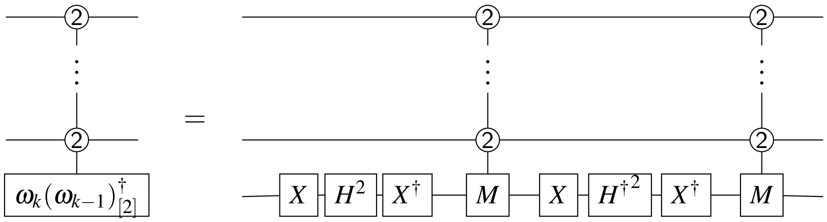
Proposition A.17. *Let $k \geq 1$. The 1-level matrix $(\omega_k)_{[x]}$ can be represented by a circuit over $\mathscr{G}_k$ using k borrowed ancillae.*

Proof. First, we build the multiply-controlled M gate, where $M = \mathrm{diag}(1, \omega_k, \omega_k^\dagger)$.



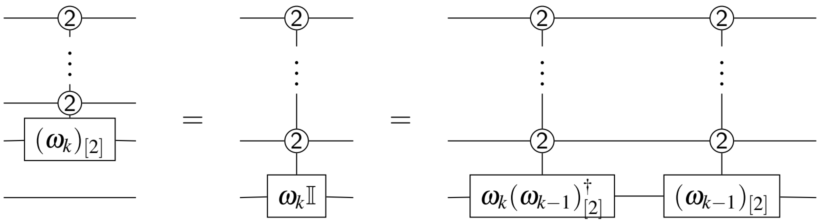
(9)

Then, we can build the multiply-controlled one-qutrit gate $\omega_k(\omega_{k-1})_{[2]}^\dagger = \omega_k \mathrm{diag}(1, 1, \omega_{k-1}^\dagger)$.



(10)

Finally, we can combine this with the multiply-controlled one-qutrit gate $(\omega_{k-1})_{[2]} = \mathrm{diag}(1, 1, \omega_{k-1})$ to get $(\omega_k)_{[2\ldots2]}$.



(11)

Since a single borrowed ancilla suffices to build (ω) and 2 borrowed ancillae suffice to build (ω_2), the above equation shows that k ancillae suffice to build (ω_k). □

Procedurally Optimised ZX-Diagram Cutting for Efficient T-Decomposition in Classical Simulation

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

A quantum circuit may be strongly classically simulated with the aid of ZX-calculus by decomposing its t T-gates into a sum of $2^{\alpha t}$ classically computable stabiliser terms. In this paper, we introduce a general procedure to find an optimal pattern of vertex cuts in a ZX-diagram to maximise its T-count reduction at the cost of the fewest cuts. Rather than reducing a Clifford+T diagram based on a fixed routine of decomposing its T-gates directly (as is the conventional approach), we focus instead on taking advantage of certain patterns and structures common to such circuits to, in effect, design by automatic procedure an arrangement of spider decompositions that is optimised for the particular circuit. In short, this works by assigning weights to vertices based on how many T-like gates they are blocking from fusing/cancelling and then appropriately propagating these weights through any neighbours which are then blocking weighted vertices from fusing, and so on. Ultimately, this then provides a set of weightings on relevant nodes, which can then each be cut, starting from the highest weighted down. While this is a heuristic approach, we show that, for circuits small enough to verify, this method achieves the most optimal set of cuts possible 71% of the time. Furthermore, there is no upper bound for the efficiency achieved by this method, allowing, in principle, an effective decomposition efficiency $\alpha \to 0$ for highly structured circuits. Even applied to random pseudo-structured circuits (produced from CNOTs, phase gates, and Toffolis), we record the number of stabiliser terms required to reduce all T-gates, via our method as compared to that of the more conventional T-decomposition approaches (namely [19], with $\alpha \approx 0.47$), and show consistent improvements of orders of magnitude, with an effective efficiency $0.1 \lesssim \alpha \lesssim 0.2$.

1 Introduction

Present-day quantum hardware is very limited, with few qubits and much noise [25]. Consequently, there are many classical techniques that are often employed to better optimise quantum circuits and/or to verify the behaviour of quantum hardware and software. A particularly useful tool for facilitating this is ZX-calculus [11, 30, 1, 12, 20], which allows quantum circuits to be expressed and simplified graphically with the use of known rewriting rules. This has been utilised for various problems in the field, including notably optimisation [14, 13, 2, 5, 15, 23, 24] and classical simulation [19, 21, 31, 9, 10, 8, 22]. On the latter problem, for instance, wherein one wishes to compute the probabilities of particular measurement outcomes of a quantum circuit, ZX-calculus can be utilised to re-express a large 'Clifford+T' circuit (which is notoriously inefficient to simulate classically) as a sum of 'Clifford' circuits (which *are* efficient to classically simulate) [19].

Re-expressing a Clifford+T circuit as sum of Clifford circuits in this way relies upon known decompositions of sets of costly 'T-gates' into cheap 'Clifford' terms. There are many such decompositions that have been discovered [7, 6, 19, 21], of varying efficiencies, and the conventional strategies [19], perhaps unsurprisingly, tend to opt for the most efficient (i.e. those which translate a set of T-gates into

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 63–78, doi:10.4204/EPTCS.406.3

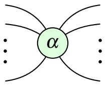
the fewest number of Clifford terms). However, the apparent efficiencies of these decompositions can be misleading. In fact, it has been noted that applying apparently *less* efficient decompositions in certain circumstances can result in *more* efficient results overall (that is, fewer Clifford terms in exchange for removing all T-gates of a circuit) [9]. Knowing when this can be applicable on local scales is often fairly straightforward. However, what is seldom considered is how appropriate applications of these less efficient decompositions can be found when looking on broader scales.

Specifically, in this paper, we demonstrate how drastic improvements to the overall efficiency can be attained by applying such decompositions to inconspicuous gates in a circuit with no obvious or immediate local benefit for doing so. We then present a means by which one can procedurally analyse a given circuit to determine where these optimal gates for decomposition are. We do this based on a heuristic of weighing vertices due to their immediate local benefit of applying such a decomposition, and then propagating these weights through their neighbours as appropriate. Lastly, we showcase the effectiveness of our method by comparing its overall efficiency at fully decomposing various Clifford+T circuits, of various numbers of T-gates, against the more traditional methods [19].

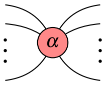
2 Background

2.1 ZX-calculus

A very useful notation with which to express quantum circuits is that provided by *ZX-calculus* [11, 30, 1], wherein all operations are expressed as *spiders* (phase rotations) about either the Z-axis or X-axis of the Bloch sphere, connected via *edges* (or *wires*). Circuits expressed in this notation are known as *ZX-diagrams* and, by convention, Z-spiders are green and X-spiders red - in either case with the angle of rotation written within (or left blank if zero), as follows:

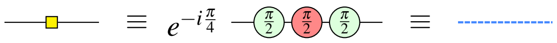


$$:= \quad |0...0\rangle\langle 0...0| + e^{i\alpha}|1...1\rangle\langle 1...1|$$



$$:= \quad |+...+\rangle\langle +...+| + e^{i\alpha}|-...-\rangle\langle -...-|$$

In addition to these two types of spiders, for convenience the Hadamard gate is often also included explicitly, as a yellow box, though this too may be decomposed (via the Euler decomposition) into spiders as such:



As shown here, an edge containing a Hadamard (referred to, unsurprisingly, as a *Hadamard edge*) may alternatively be expressed as a dashed blue line.

From these basic components, any quantum circuit may be expressed as a ZX-diagram. Indeed, completeness for the Clifford gateset is achieved provided phases of $\frac{n\pi}{2}$ are allowed (where $n \in \mathbb{Z}$). (Note also that, as phases correspond to rotations, they are all of modulo 2π.) To extend the completeness to the Clifford+T gateset (including T-gates and Toffoli gates), the resolution must be expanded to support phases of $\frac{n\pi}{4}$.

2.2 ZX-diagram rewriting

The major benefit of ZX-calculus is that it contains a set of well-defined *rewriting rules* which outline how certain structures within ZX-diagrams may be equivalently written in simpler forms - allowing a circuit to be simplified while maintaining its behaviour. In particular, the fundamental rewrite rules are as outlined in figure 1.

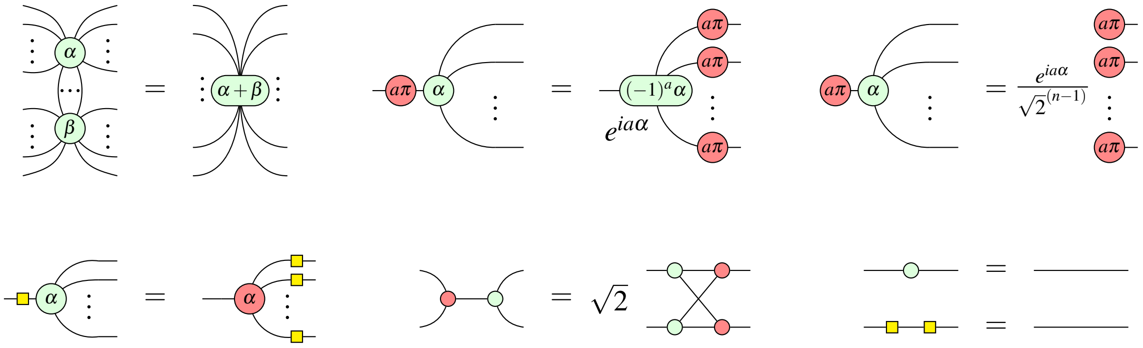


Figure 1: A set of the basic rewriting rules of ZX-calculus [19], where Greek letters denote arbitrary real variables, $[0,2\pi)$, and Latin letters denote arbitrary boolean variables, $\{0,1\}$. Note that the rules still apply if all the spider colours are inverted. These rules are known, in column-major order, as (a) spider fusion, (b) colour change, (c) π-commutation, (d) bialgebra, (e) state copy (where n is the number of output edges), (f) identity removal, (g) Hadamard cancellation.

From these basic rules, one may derive a number of more complex rules. In particular, *local complementation* and *pivoting*, outlined in figure 2, prove to be extremely useful in reducing ZX-diagrams. A more thorough collection of derived rules may be found in [30].

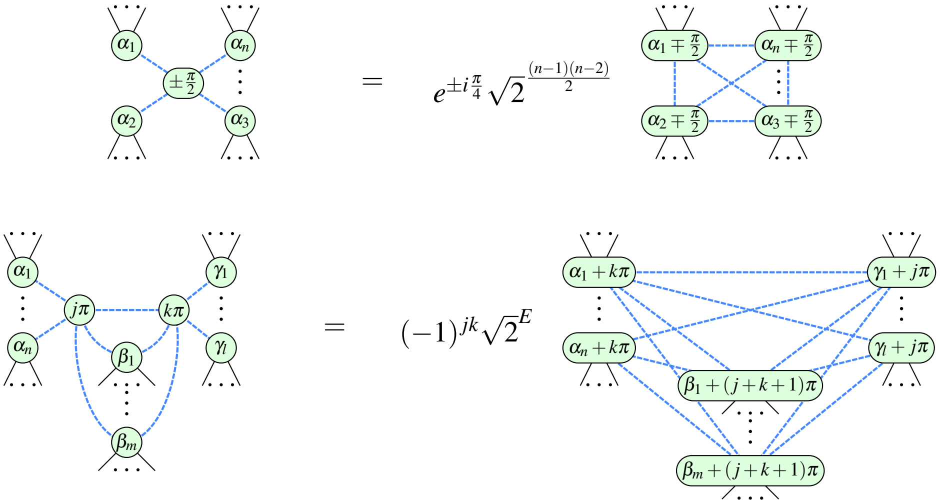
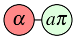


Figure 2: Two important derived ZX-calculus rewrite rules [19], namely (top) local complementation and (bottom) pivoting. Here, $E = (n-1)m + (l-1)m + (n-1)(l-1)$.

A particular subset of ZX-diagrams are those with no open input or output wires. These are known as *scalar* diagrams, and the rules above are sufficient to reduce such diagrams, of the Clifford gateset, to complex scalars, making use of the following scalar relations:

$= 1 + e^{i\alpha}$  $= \sqrt{2}e^{ia\alpha}$

2.3 Classical simulation

To verify the behaviour of complex quantum algorithms, especially when present-day quantum hardware is insufficient to run them, one may turn to classical simulation. There are two classes of this - namely *strong* simulation and *weak* simulation. For the purposes in this paper, we will focus on the former, wherein the aim is to determine the probability of a specific measurement outcome. In short, this may be done with ZX-calculus by 'plugging' the inputs and outputs with spiders corresponding to the desired states and measurements to produce a *scalar* ZX-diagram, which may then be fully reduced to a simple complex scalar value. This result will represent its amplitude, A, and thus relates to its probability, $|A|^2$.

Notably, Cliford diagrams (those restricted to phases of $\frac{n\pi}{2}$) are very efficient to simulate classically. This manifests itself in ZX-calculus in that such diagrams, given no open inputs or outputs, may be fully reduced to a scalar via the rewriting rules highlighted above. Clifford+T diagrams, on the other hand, are notoriously inefficient to simulate classically. While the rewriting rules may remove a number of T-gates from a ZX-diagram, it is generally unable to remove them all, meaning such diagrams are unable to fully reduce to a scalar. To deduce the scalar amplitude in such cases then, one must make use of decompositions to translate the Clifford+T diagram into a sum of efficiently-reducible Clifford diagrams. The inefficiency lies in the fact that the number of such summand Clifford terms one attains in place of a Clifford+T diagram scales exponentially with the number of T-gates. This is typically quantified with a parameter α which represents the efficiency of the precise decomposition(s) used, given it translates a Clifford+T diagram of t T-gates into a sum of $2^{\alpha t}$ Clifford terms. So, smaller values of α describe more efficient decompositions.

For instance, the decomposition presented by Bravyi, Smith, and Smolin [7], and expressed in ZX-calculus terms by Kissinger and van de Wetering [19], allows sets of 6 T-gates to be replaced with a sum of 7 Clifford terms, according to figure 3. As such, this "BSS" decomposition scales as $7^{t/6} \approx 2^{0.468t}$, hence $\alpha \approx 0.468$. The current state of the art T-decompositions, meanwhile, achieve $\alpha \approx 0.396$ [26].

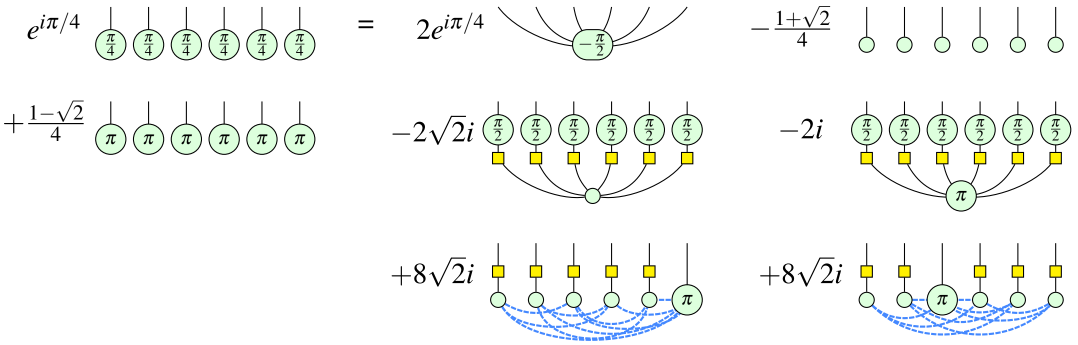


Figure 3: The BSS decomposition [7], relating a set of 6 T-gates to a sum of 7 Clifford terms, expressed in ZX-diagrams as per [19].

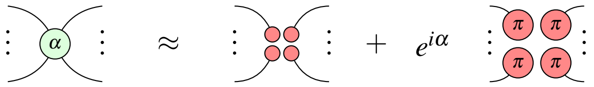
As outlined in [19], after every decomposition, some further reduction in T-count may be facilitated by Clifford simplification via the rewriting rules, such that the $2^{\alpha t}$ actually represents an upper-bound estimate of the number of terms ultimately produced. Moreover, in their work they essentially select groups of 6 T-gates for decomposition arbitrarily, without regard for whether certain groupings may be more likely to allow further simplification after decomposition. Lastly, while their method relies primarily on the BSS decomposition, when the number of T-gates remaining in any graph falls below 6, they fall back on other known decompositions.

3 Method

3.1 Graph cutting

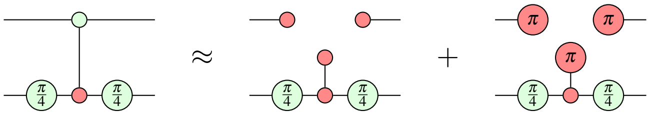
While Clifford+T diagrams are typically reduced via such T-gate decompositions as that of Bravyi, Smith, and Smolin, highlighted above, paying close attention to the structures inherent in a given ZX-diagram can reveal that blindly decomposing T-gates in an essentially random order is not necessarily optimal. In fact, decomposing conveniently positioned *Clifford* spiders - even with an apparently less efficient decomposition - can actually produce more efficient results.

From the definitions of Z- and X- spiders in section 2.1, one can infer the basic relation:

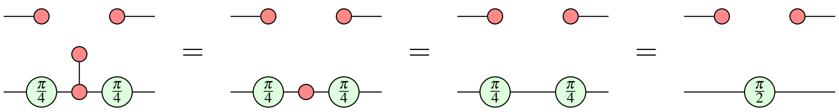


Note that the equality here is up to some global scalar factor (neglected for brevity). This acts as a very simple decomposition for any arbitrary spider. Herein, the act of applying this particular decomposition will be referred to as '*cutting*' the graph, as structurally it behaves like physically slicing a vertex from its edges. Superficially, this may not seem particularly useful as, applied to T-gates, this doubles the number of terms for each one (hence has a very poor efficiency of $\alpha = 1$). However, by cutting appropriate spiders (not necessarily even T-like spiders), this can allow some T-gates to cancel, along both paths of the decomposition.

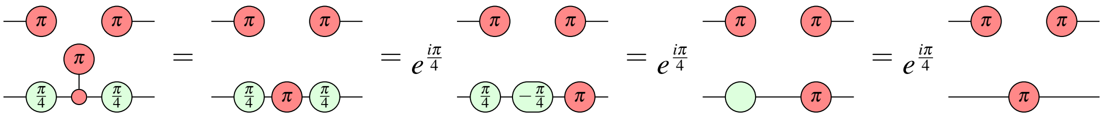
Specifically, consider a subgraph consisting of two T-like spiders sandwiched between a CNOT. Cutting one end of the CNOT (the side opposite the T-gates) will produce two terms, like so:



An observant reader may notice that both of these terms, after a little simplification via the rewriting rules, may fuse their T-gates into a Clifford. The left-hand term, for instance, simplifies as follows:



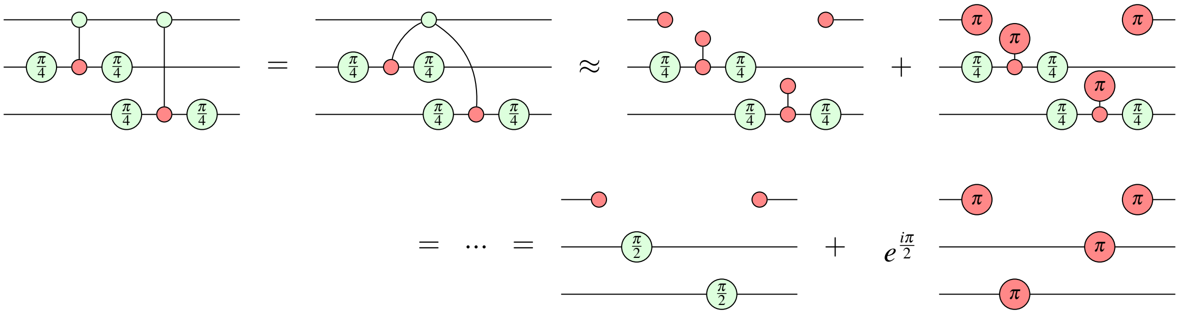
The right-hand term simplifies similarly, except utilising π-commutation in place of identity removal:



Consequently, via a cut of a Clifford spider the T-count has been reduced by 2. This gives a decomposition efficiency of $\alpha = 0.5$. Moreover, if the green end of the CNOT had itself taken a T-like phase, then that too would have been removed (specifically, converted into a scalar factor directly by the decomposition). In that case, one would have observed a better efficiency of $\alpha = 1/3$.

3.2 CNOT-grouping

One may further recognise that it is possible for any number of these T-CNOT-T arrangements to be aligned such that their respective CNOTs may fuse. In such cases, each pair of T-like spiders may fuse to a Clifford as above, though still just requiring one vertex cut overall. As a simple example, the following shows how two sets might be grouped and reduced in this way:

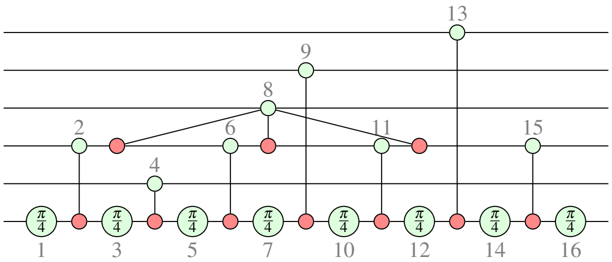


Here, the T-count has been reduced by 4, still at the expense of just one vertex cut (hence 2 Clifford terms). This corresponds to $\alpha = 0.25$. It should be clear from this point that this reasoning could be extended ad infinitum, with arbitrarily many T-CNOT-T subgraphs being fusible and hence arbitrarily many T-gates being reducible to Clifford at the expense of just one cut, giving $\alpha \to 0$.

A similar concept to this was recognised in [9] (albeit notated in a different fashion), wherein, for a single decomposition utilising π-commutations, T-count reductions of up to 286 were found on SAT counting [4, 3] ZX-diagrams (giving $\alpha \approx 0.0035$). However, this work utilised a very simple heuristic amounting to prioritising decompositions of T-gates with the maximum number of immediately relevant connections. Perhaps surprisingly, however, this naïve heuristic seldom produces the most efficient solutions and indeed can often be extremely suboptimal. Moreover, their approach to recognising instances where these cuts are applicable is essentially limited to those where the decomposition is to be applied to a T-like spider, rather than any arbitrarily-phased spider as per the means outlined above. And lastly, by acting on ZX-diagrams *after* they have undergone full Clifford simplification, their approach risks losing much of the graph's structure and thus missing relevant patterns.

3.3 Cutting in tiered structures

Just as a CNOT may be the only thing standing in the way of two or more T-like spiders from fusing, so too may a CNOT be the only obstacle preventing some T-CNOT-T sandwiches from grouping. Consider, as a prime example, the structured circuit that follows, where the Z-spiders here have been labelled for easy reference:



Immediately, one might recognise 7 T-CNOT-T sandwiches, centred on vertices #2, #4, #6, #9, #11, #13, and #15, respectively. Notably, however, there is some clashing here, in that - for example - the sandwich centred on #2 and that centred on #4 are mutually exclusive, as they a share one of their T-gates. Indeed, there are a number of possible ways of reducing this circuit via the means outlined so far in this paper.

For example, cutting vertices #4, #9, and #13 would allow 6 T-gates to reduce to Clifford. Alternatively, cutting #2, #6, #11, and #15 would reduce all 8 T-gates, at the cost of 4 cuts (hence $\alpha = 0.5$). But, the best solution here would be to firstly cut vertex #8 - even though this doesn't immediately allow any T-gates to reduce - as this then allows vertices #2, #6, #11, and #15 to fuse into one. Cutting this newly fused vertex then would allow all 8 T-gates to reduce. So, this solution would reduce the T-count entirely (all 8 T-gates) at the cost of just 2 cuts (hence $\alpha = 0.25$). And of course, this reasoning could be extended for higher tiers, where the optimal initial cuts are two, three, or more, steps away from any T-gates.

Similarly, there may be instances where *multiple* CNOTs are directly blocking some set of T-gates from fusing. In such cases, the likelihood that the cutting all of those CNOTs would be worthwhile to reduce the T-gates they block will be determined by the ratio of the CNOTs to T-gates involved, as well as whether some of those CNOTs are considered worthwhile cuts in their own right, with regard to any other T-gates that they alone may be blocking. We can call such groups of spiders 'spousal', with respect to the children spiders they are collectively blocking from reducing.

Evidently, therefore, selecting which vertices to cut, and indeed in which order, is a very intricate task (though the former is more important as suboptimality in the latter can be corrected for via a slight modification and parametric analysis, as detailed in appendix A). Naïvely tackling this problem via an exhaustive, brute force, approach would require checking the reduction achieved by every possible combination of vertex cuts. This is obviously infeasible for large-scale graphs, as the time complexity scales exponentially with the number of vertices. Consequently, a heuristic approach is desired. On this note, as we have shown, simply prioritising vertices which are directly blocking the most number of T-like pairs from fusing is not generally optimal. Rather, it is preferential to look at the whole picture and determine the optimal cuts on higher tiers.

3.4 Optimised cutting procedure

The solution we propose is a procedure based on assigning weights to vertices, determined by how many T-like gates they are preventing from fusing to Clifford, and then propagating these weights through any neighbours which are then preventing weighted vertices from fusing, and so on up the tiers. Particular care is given to balance the weightings appropriately, especially in places where multiple 'spousal' cuts are required to facilitate a fusion of their children.

Note that we may label the weight of a vertex (i.e. spider) v, for a particular tier t, as w_v^t, such that, for instance, w_{12}^2 refers to the weight of vertex #12 with respect to tier #2. Given this, the procedure steps are as follows:

1. Partially simplify the circuit such that any instances of spider fusion are applied (so no like-coloured spiders remain directly connected via a solid edge) and any π-phase spiders are pushed to one side or into CNOTs via the π-commutation and fusion rules. Then, let $t = 0$ and assign an initial weight of 0 to every spider (i.e. let $w_v^0 = 0 \;\forall v$).

2. For any pair of T-like spiders that are separated by k (that is, one or more) CNOTs, add $2/k$ to the weights of the opposite ends of each of those CNOTs. (For any given CNOT, take care not to count a particular T-spider more than once.) Now, for instance, any CNOT that is preventing 2 T-spiders from fusing to Clifford will have a corresponding weight of 2, and any CNOT preventing,

collectively, 4 T-spiders from reducing to Cliffords will have a corresponding weighting of 4, etc. Similarly, if, for instance, 3 CNOTs are collectively blocking a single pair of T-spiders from fusing, then each will have a weighting of $2/3$.

3. Increment $t \leftarrow t+1$. Then, similar to step 2, for any *weighted* vertex v of the previous tier (i.e. any v for which $w_v^{t-1} \geq 0$) that is separated from fusing with another weighted vertex of *any* lower tier (i.e. any v for which $w_v^u \geq 0$ for any $u < t$) by k (that is, one or more) CNOTs, add $\gamma(w_v^{t-1})/k$ to the weight of the opposite ends of each of those CNOTs. Here, the γ function normalises a given weighting to the range $[0,1]$, such that (crudely speaking) 0 roughly means "very unlikely to be a worthwhile cut" and 1 "very likely to be a worthwhile cut": $\gamma(w) := \min\left(\frac{w}{2}, 1\right)$.

4. Repeat step 3 until no new changes are made (that is, until no weightings, w_v^t, are found for any v, given the current t). At this point, one will be left with weightings for every vertex, for every tier ($w_v^t \ \forall v,t$). From this, one can trivially extract the relevant data, namely, for every vertex v, its *maximum* weight w_v^t (for any t) and, respectively, the maximum t for which the vertex has a weight (i.e. largest t for which $w_v^t \geq 0$). One may label the maximum weight of a vertex, W_v, and its maximum tier for which it has a non-zero weight, T_v.

5. Among the vertices with the largest recorded T, namely T_{max}, select the one with the greatest max weight W_v (i.e. select vertex V s.t. $W_V \geq W_v \ \forall v$ s.t. $T_v = T_V = T_{max}$). (Note that for this step only, we may temporarily add 1 to the weight of any vertex of a T-like phase.) If this weight is below 2 (hence $\gamma(W_V) \leq 1$, implying the cut would not likely be worthwhile), then search instead among the vertices of the lower tier (i.e. for which $T_v = T_{max-1}$), until an appropriate vertex is found for which $W_V \geq 2$. This is the vertex which the heuristic has concluded is likely an optimal choice to cut. As such, cut this vertex (i.e. decompose it into two branches as per section 3.1).

6. Partially simplify each branch, without compromising the graph structure. Specifically, push any new π-phase spiders to one side, and/or into CNOTs, via repeated applications of the π-commutation rule (and fusion), and thereafter apply the fusion rule until no like-coloured spiders remain connected via a solid edge. Moreover, when fusing weighted spiders, update their combined weight accordingly as the sum of their respective max weights (i.e. in fusing vertex B into vertex A, the former is removed along with its recorded weightings and tier data and the latter is updated as $W_A \leftarrow W_A + W_B$). Similarly, the max tier of the newly fused vertex takes that of the larger of the two fused vertices (i.e. $T_A \leftarrow \max(T_A, T_B)$). Moreover, recalculate the weightings on any vertices whose children have been altered by this partial simplification.

7. Repeat steps 5 and 6 until no further cuts are made (or the number of T-spiders ≤ 2). If any T-like spiders remain, then these may be decomposed via a typical T-decomposition (e.g. the BSS decomposition outlined in section 2.3).

Applying this procedure to a Clifford+T ZX-diagram will result in a set of Clifford ZX-diagrams whose sum equals the original. For scalar diagrams (those with no open input or output wires), each summand will simply be a scalar term, and thus the overall sum will be likewise [19].

Python code, based on the PyZX package [16, 17], that implements this procedure may be found at `https://github.com/mjsutcliffe99/ProcOptCut` [28], and a step-by-step illustrative example of this procedure in action is shown therein.

4 Results

4.1 Complexity and efficiency

The exact runtime complexity of the procedure is difficult to discern as it depends on the density of the graph (i.e. typical number of neighbours the vertices have) and its number of 'tiers', T_{max}, (which is heuristically determined). Very reasonably, one could assume $T_{max} \ll V$ (for a non-trivial number of vertices, V), and so a crude upper-bound (taking, very unrealistically, maximal density) may be given by $O(V^2)$. Even this gross over-estimate of the runtime complexity shows that, compared to the number of stabiliser terms produced ($2^{\alpha t}$ for t T-gates and some $\alpha < 1$), the procedure's runtime is negligible, so that any improvements it offers to the number of the stabiliser terms can be taken as is.

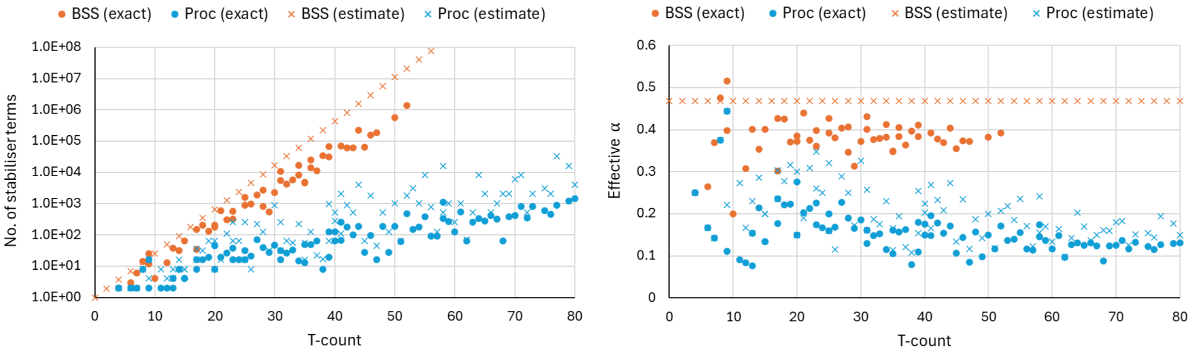
To test the effectiveness of the proposed method, we explored how often - for circuits small enough to verify - it was able to find the *most* optimal set of vertex cuts, rather than simply *an* optimal set. Specifically, we considered many random, non-trivially structured (see appendix B) ZX-diagrams of 16 internal Z-spiders or fewer and measured the number of stabiliser terms required to remove all T-spiders via the method outlined in section 3.4. In each case, we also tested every possible combination of vertex cuts on the internal Z-spiders (applying the reasoning of appendix A to correct for any suboptimal cut ordering). Thus, we could determine with certainty the most optimal set of vertex cuts (on Z-spiders) and correspondingly the number of stabiliser terms produced (or, by extension, the effective α) and compare this, in each case, to the result achieved by our method. In this way, we observed that our method found the most optimal set of cuts possible (on Z-spiders) 71% of the time. Even so, in every case in which the method failed to find the most optimal solution, it still invariably found a very good solution with an α usually only marginally above that of the best. It is worth noting, however, that we cannot be certain that this would translate to a similar success rate for much larger circuits (i.e. those too big to verify by brute force), where greater levels of structure (and indeed chaos) are possible. As such, to extend our analysis we compared our method also to that of the more typical approach of T-decompositions a la [19], of which we outline our results ahead.

4.2 Experimental measurements for random circuits

Applying the procedure outlined in section 3.4 to a Clifford+T ZX-diagram will produce an exact result (e.g. for computing the probability amplitude of a Clifford+T diagram for classical simulation). However, in benchmarking the efficiency of the method, one is more interested in the *number* of stabiliser terms the original diagram is decomposed to, rather than the precise value of the sum of these terms. With this in mind, one can run a '*blind*' version of the procedure to compute an (upper-bound) estimate of the number of resulting terms in linear time. As well as not providing the final numerical result, this blind version of the procedure also misses out on inter-step ZX-calculus simplifications, and hence typically overestimates the number of resulting terms (and thus is an upper-bound estimate). The appropriate modifications for this version of the procedure are outlined and justified in appendix C.

Similarly, one can likewise calculate an upper-bound estimate of the number of terms produced after fully decomposing a Clifford+T diagram via the BSS-based approach of [19]. This is much more simple to do as the BSS decomposition efficiency is known in advance, namely $\alpha \approx 0.468$. Thus, an estimate of the number of resulting terms this method achieves for a given diagram of T-count t is given by $2^{0.468t}$, which can be computed trivially in constant time. As before, this does not account for the inter-step ZX-calculus simplifications and hence is an approximate upper-bound estimate.

For various randomly generated ZX-diagrams (constructed from CNOTs, $\frac{n\pi}{4}$ phase gates, and Toffolis as per appendix B), we measured the number of stabiliser terms required to remove all T-spiders via the approach of Kissinger and van de Wetering [19] versus the method we propose in section 3.4. For computationally feasible T-counts (those computable within a few minutes), we can compute these exactly with both methods, and for larger (computationally infeasible) T-counts, we instead rely on the upper-bound estimations outlined above. For each of these measurements, we can also infer the overall effective decomposition efficiency, α, by simply taking $\alpha = \frac{1}{t}\log_2 n$, where t is the initial T-count of the circuit (after full Clifford simplification) and n is the number of stabiliser terms to which it is reduced. The results are illustrated in figure 4. (Note that the initial T-counts are all taken as that achieved after full Clifford simplification.)



(a) Number of stabiliser terms versus T-count

(b) Effective efficiency α versus T-count

Figure 4: (a) The number of stabiliser terms, n, produced after decomposing all t T-gates, and correspondingly (b) the effective overall decomposition efficiency α (given by $\alpha = \frac{1}{t}\log_2 n$) for numerous randomly generated (see appendix B) pseudo-structured Clifford+T circuits of various T-counts. In each case, we measure experimentally the exact results, as well as approximate upper-bound estimations, achieved by both the conventional ("BSS") method of [19] and the procedural approach ("proc") presented in this paper.

We observe that our procedural method, applied to circuits of this type, is very effective at minimising the number of resulting stabiliser terms, compared to relying predominantly on naïve applications of the BSS decomposition. The exact effectiveness of our method can vary quite substantially from circuit to circuit, depending on how much structure they embed. The direct BSS approach, meanwhile, tends to be a lot more consistent, though can also vary somewhat depending on how much inter-step simplification it is able to undertake. Regardless, the procedural method very consistently shows magnitudes of improvement on these circuits, allowing above even double the T-counts to be computed within the same time frame.

This is reflected too in the effective α measurements, where, for circuits of this type, our method offers typically $0.1 \lesssim \alpha \lesssim 0.2$ for such pseudo-structured circuits. The results are less remarkable on trivially small T-counts as such small circuits have little room for much structure of which to take advantage. Evidently, therefore, this method is, as intended, very well optimised for decomposing Clifford+T circuits that exhibit some structure, enabling such circuits of much larger T-counts to be computed within reasonable time frames.

For full context, it is worth remembering two points when considering these results. Firstly, the random circuits generated for these experiments were those with some inherent localised structural elements. And secondly, while the runtime of the relevant steps of the procedure, after each decomposition, is very quick, it does nevertheless contribute time to each term that is not present in the BSS method, so that the difference in numbers of terms produced by the two methods does not translate 1-to-1 to a difference in runtime. In other words, computing n terms via the procedural method may be a little slower than computing n terms via the BSS method (although this difference will not be more than some small factor, so that the improvement offered by the reduced number of terms still vastly outweighs this offset).

All of the above experiments may be repeated from the corresponding Jupyter notebook, hosted on Github [28].

5 Conclusion

We demonstrated how a procedure could be designed to analyse the structure of any given Clifford+T quantum circuit in order to determine an optimised set of vertex cuts to efficiently decompose it to a sum of (classically computable) Clifford terms. This is contrary to the more conventional approach [19], which applies decompositions essentially arbitrarily, without such regard to the specific structures involved.

Specifically, we show that our method is very effective at finding optimal sets of vertex cuts, with a 71% success rate at finding the *most* optimal set when applied to random small semi-structured circuits. We further show that our method is very efficient at decomposing even larger such circuits, consistently outperforming the more conventional approach [19] by orders of magnitude, with regard to the number of resulting terms (and by extension the runtime). What this means in practise is that our method could allow for classical simulation, within a reasonable time frame, of Clifford+T circuits of more than double the T-count as could be achieved with the conventional methods. This is hugely relevant to verifying the behaviour of quantum software and hardware for present-day NISQ (noisy intermediate-scale quantum) devices [25].

While already very effective, there are many ways in which the method outlined in this paper could be improved - many stemming from the rigid scope with which it applies and propagates weights. After all, T-gate fusion facilitated by cutting a CNOT is just one way in which the T-count of a circuit may be reduced. It would be worth considering also the potential of cuts to remove T-gates by pushing them into the scalar factor (e.g. via the state copy rule) or to partition small segments from the graph. The method ought also to consider simplification and weight propagation laterally, rather than solely through the lens of ‘tiers’. Indeed, the ‘partial simplification’ strategy we utilise (to simplify while maintaining the structure) does not take into account some vital steps in the normal ‘full reduce’ [16] function (namely pivoting - i.e. on CNOTs - and local complementation). We suspect this is largely the reason we didn’t observe an even higher success rate in verifying how often the method was able to find the best solutions on small circuits. Moreover, a more robust analysis could determine appropriate weightings *without* the need for the circuit to be expressed in a very rigid graph-like form, such that further simplification between steps could be enabled and, for instance, Toffolis could be expressed as phase gadgets (so that we are not restricted due to the arbitrary choice of which way around to decompose each Toffoli’s control qubits). And lastly, of course, one could consider cuts on X-spiders as well as just Z-spiders (this might be particularly relevant if there are many Hadamards involved, resulting in many X-spiders of T-like phase).

There are also a number of ways in which this concept, more broadly, could be improved, such as developing newer and better heuristics - perhaps even different heuristics for different types of circuit (e.g. dense circuits, or those with many Toffolis, etc.). Nevertheless, we demonstrate very clearly how analysing the circuit structure and applying decompositions discriminately can offer vastly more efficient results than simply decomposing the T-spiders directly with a decomposition that has a better *immediate* efficiency.

References

[1] Miriam Backens, Simon Perdrix & Quanlong Wang (2017): *A Simplified Stabilizer ZX-calculus*. *Electronic Proceedings in Theoretical Computer Science* 236, p. 1–20, doi:10.4204/eptcs.236.1.

[2] 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.

[3] Niel de Beaudrap, Aleks Kissinger & Konstantinos Meichanetzidis (2021): *Tensor Network Rewriting Strategies for Satisfiability and Counting*. *Electronic Proceedings in Theoretical Computer Science* 340, p. 46–59, doi:10.4204/eptcs.340.3.

[4] Lucas Berent, Lukas Burgholzer & Robert Wille (2022): *Towards a SAT Encoding for Quantum Circuits: A Journey From Classical Circuits to Clifford Circuits and Beyond*. doi:10.48550/arXiv.2203.00698.

[5] 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.

[6] 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.

[7] 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.

[8] Tristan Cam & Simon Martiel (2023): *Speeding up quantum circuits simulation using ZX-Calculus*. *arXiv preprint arXiv:2305.02669*.

[9] Julien Codsi (2022): *Cutting-Edge Graphical Stabiliser Decompositions for Classical Simulation of Quantum Circuits*. Master's thesis, University of Oxford. Available at `https://www.cs.ox.ac.uk/people/aleks.kissinger/theses/codsi-thesis.pdf`.

[10] Julien Codsi & John van de Wetering (2022): *Classically Simulating Quantum Supremacy IQP Circuits trough a Random Graph Approach*. *arXiv preprint arXiv:2212.08609*.

[11] 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.

[12] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes*. Cambridge University Press, doi:10.1017/9781316219317.

[13] 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.

[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.

[15] 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.

[16] Aleks Kissinger & John van de Wetering: *PyZX*. Available at `https://github.com/Quantomatic/pyzx`.

[17] 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.

[18] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Physical Review A* 102(2), doi:10.1103/physreva.102.022406.

[19] 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.

[20] Aleks Kissinger & John van de Wetering (2023): *Picturing Quantum Software [Preprint]*.

[21] 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.

[22] Mark Koch, Richie Yeung & Quanlong Wang (2023): *Speedy Contraction of ZX Diagrams with Triangles via Stabiliser Decompositions*. *arXiv preprint arXiv:2307.01803*.

[23] 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.

[24] Maximilian Nägele & Florian Marquardt (2023): *Optimizing ZX-Diagrams with Deep Reinforcement Learning*. *arXiv preprint arXiv:2311.18588*.

[25] John Preskill (2018): *Quantum Computing in the NISQ era and beyond*. *Quantum* 2, p. 79, doi:10.22331/q-2018-08-06-79.

[26] 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.

[27] Matthew Sutcliffe: *ParamZX*. Available at `https://github.com/mjsutcliffe99/ParamZX`.

[28] Matthew Sutcliffe: *ProcOptCut*. Available at `https://github.com/mjsutcliffe99/ProcOptCut`.

[29] Matthew Sutcliffe & Aleks Kissinger (2024): *Fast classical simulation of quantum circuits via parametric rewriting in the ZX-calculus [Preprint]*.

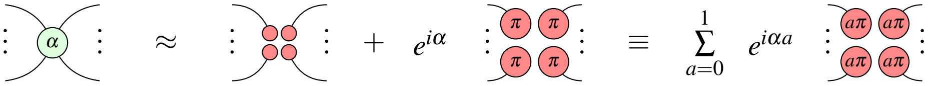
[30] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. doi:10.48550/arXiv.2012.13966.

[31] 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.

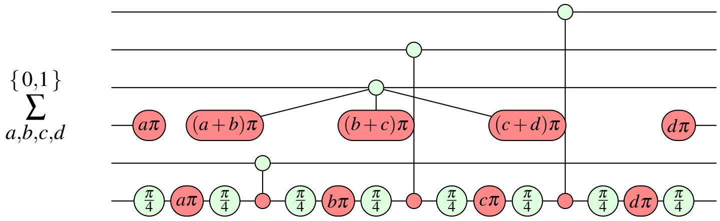
A Cut order correction

It was shown in section 3.3 that determining the *order* in which to cut the vertices is apparently at least as important as determining *which* vertices to cut. In the example showcased there, it seemed necessary for the optimal solution to cut vertex #8 first, such that vertices #2, #6, #11, and #15 could then be fused and cut as one. It would appear that recognising the same vertices to cut but applying a different cut ordering (specifically cutting vertex #8 *last*) would achieve the same ends at the cost of 5 cuts rather than 2 (and hence a much less efficient 32 stabiliser terms rather than 4). However, with slight alteration (using the modified PyZX package of [27]) to denote the cuts parametrically, followed by some simple parametric analysis, a suboptimal cut ordering can effectively be corrected to its more optimal arrangement at a negligible cost to the runtime.

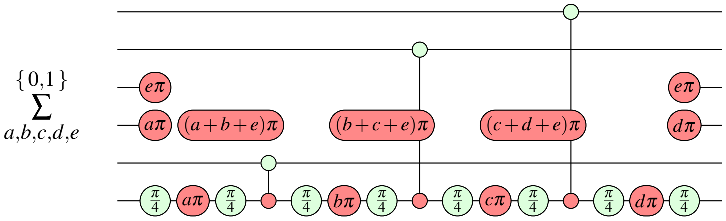
Firstly, one can parameterise the cutting decomposition of section 3.1 like so:



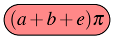
Now consider again the circuit shown in section 3.3, and imagine cutting first vertices #2, #6, #11, and #15. Writing this in the parameterised form - needing only one parameterised graph (rather than 16 evaluated graphs) - leads to the following, after only some trivial spider fusion:



Cutting lastly vertex #8 results in the following:



Here we have a graph in 5 parameters (a,b,c,d,e), containing 3 free nodes (legless spiders). As the parameters are boolean, it necessarily follows that each of these free nodes can be replaced with a scalar factor of 0 or 2, such as follows:



$$= \begin{cases} 2 & \text{if } a \oplus b \oplus e = 0 \\ 0 & \text{if } a \oplus b \oplus e = 1 \end{cases}$$

Moreover, for any combination of parameter values that results in a scalar factor of 0 in one or more of these free nodes, one can ignore the entire corresponding graph (as the whole graph then becomes 0). In other words, one is only interested in the sets of parameter values which result in non-zero scalar factors for all free nodes. So, given the equation above, the leftmost free node must equal 2 and thus $a \oplus b \oplus e = 0$. Rearranging this for, say, a gives: $a = b \oplus e$. And now, every instance of a throughout the parameterised graph can be substituted out for $b \oplus e$, thus reducing the number of parameters from 5 to 4. Repeating this reasoning for the remaining two free nodes finds that $b = c \oplus e$ and $c = d \oplus e$, resulting in all parameters being reduced to some combination of d and e.

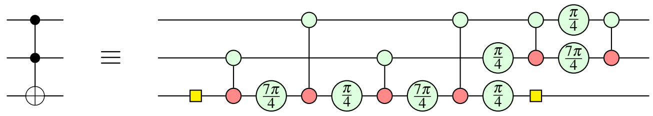
Consequently, while only needing to reason on one graph, the number of parameters has been reduced from the 5 attained via a suboptimal cut ordering to the most optimal 2. Indeed, this graph is now equivalent to what would have been attained if the more optimal cut ordering had been adopted (namely cutting vertex #8 first and then fusing the remaining 4 before cutting them as one). Having effectively corrected suboptimal cut ordering, one can then expand the parameterised graph out into its (in this case 4) distinct evaluated graphs and proceed with simplification as in section 3.

Note, however, that the procedure outlined in section 3.4 prioritises higher tiers so that the cut ordering is already optimised and thus this parametric reasoning should not generally be necessary. Nevertheless, it might prove beneficial to the procedure in extreme cases, and indeed is applicable in other cutting techniques (such as the brute-force verifications of section 4.1).

B Generating random pseudo-structured circuits

The procedure presented in this paper, by design, works best on circuits that are highly structured. This raised an interesting problem when benchmarking, as testing on wholly random circuits would not properly showcase its effectiveness and, conversely, demonstrating only ideal example cases would not yield particularly informative results as such circuits could be made arbitrarily ideal, sending $\alpha \to 0$. Consequently, for the benchmarking experiments, we generated random circuits in such a way as to include some localised structural elements, so as to avoid trivial (unstructured) cases, while also not simply designing "best case" circuits on which to experiment.

Specifically, for the main benchmarking tests (figure 4), we generated circuits from random combinations of randomly placed T-CNOT-T sandwiches, Toffoli gates, CNOTs, and phase gates. We varied the number of such instances of these components in order to vary the T-counts of the circuits. Meanwhile, the circuits for the initial, small-scale verification experiments (section 4.1) were generated likewise, minus the Toffoli gates (as even one or two of these produce circuits too large to verify). Note that in randomly placing multi-qubit gates, each of its qubits may be placed randomly among the circuit's qubits, and note also that we express Toffoli gates in ZX-diagram form as follows:



This approach ensures we create appropriately random circuits of non-trivial local structure, akin to that of section 3.3 or the composed-Toffolis circuits seen in [18].

C Estimating efficiency

Following the procedure of section 3.4 will result in a list of scalar terms whose sum gives the overall scalar value corresponding to the original Clifford+T circuit. However, for the purposes of benchmarking the effectiveness of the procedure, it is not important what that scalar value is, but rather how many scalar terms were produced (and hence the total runtime). With this in mind, one can determine an approximate upper-bound estimate of the number of terms that would be produced, in linear time, without having to compute all such terms.

This can be done by parameterising the branches of each cut into a single parametric graph (rather than two sets of distinct enumerated graphs), akin to [29] (and utilising the corresponding modified PyZX package of [27]). As such, every cut can be recorded parametrically on a single graph, without the need to append a list of distinct graphs, similar to the figures shown in appendix A. By continuing the procedure on this parametric graph, one will ultimately arrive at a final Clifford graph of n parameters (given n cuts), which represents 2^n distinct graphs whose sum is the solution. Thus, the number of terms will be known to be 2^n. This is, however, an upper-bound estimate as the parameterisation of the cuts can prevent much inter-step simplification that could be facilitated with exact numerical phases.

Scalable Spider Nests (...Or How to Graphically Grok Transversal Non-Clifford Gates)

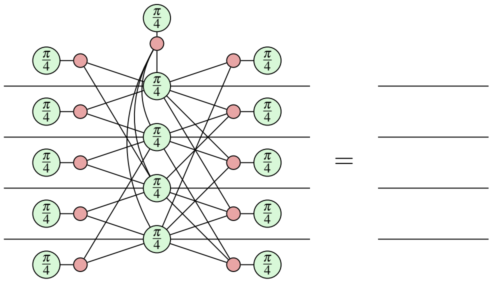
Aleks Kissinger
University of Oxford
aleks.kissinger@ox.ac.uk

John van de Wetering
University of Amsterdam
john@vdwetering.name

This is the second in a series of “graphical grokking” papers in which we study how stabiliser codes can be understood using the ZX-calculus. In this paper we show that certain complex rules involving ZX-diagrams, called spider nest identities, can be captured succinctly using the *scalable* ZX-calculus, and all such identities can be proved inductively from a single new rule using the Clifford ZX-calculus. This can be combined with the ZX picture of CSS codes, developed in the first “grokking” paper, to give a simple characterisation of the set of all transversal diagonal gates at the third level of the Clifford hierarchy implementable in an arbitrary CSS code.

The ZX-calculus [12, 13] is a useful tool for expressing and reasoning about quantum computations. It represents computations, such as quantum circuits or measurement patterns, as certain labelled open graphs called *ZX-diagrams*, subject to a collection of rewrite rules that can be used to transform and simplify diagrams. While the literature tends to talk about “the” ZX-calculus, there are actually several calculi of increasing power. One of the simpler versions is what we will call here the *Clifford ZX-calculus*, which is complete for Clifford ZX-diagrams, i.e. diagrams whose phase parameters are restricted to integer multiples of $\pi/2$ [3]. These give a natural generalisation of Clifford circuits, and the Clifford ZX-calculus admits many efficient algorithms, e.g. for reducing Clifford diagrams to normal form, equality checking, and computing arbitrary measurement amplitudes. In some sense, it is the graphical counterpart to stabiliser theory (see e.g. [3, 6]), so it gives a natural setting for studying quantum error correction [18].

However, the Clifford ZX-calculus, unlike some of its more powerful counterparts [23, 17] is known to be incomplete for ZX-diagrams whose phases are not all multiples of $\pi/2$. An interesting family of equations that are not provable in the Clifford ZX-calculus are the *spider-nest identities*, where elaborate configurations of *phase gadgets* can collectively cancel out with each other [4]. For example, connecting a $\pi/4$ phase gadget to all $2^4-1=15$ non-empty subsets of 4 qubits is equal to the identity:



(1)

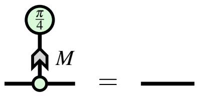
While this might seem like a very specific and complicated rule, such identities have already been used to great effect for T-gate optimisation, originally in the language of phase polynomials [16, 2] and later explicitly as ZX-diagram rules [4].

The starting point for this paper is the observation that all spider-nest identities can be succinctly characterised using certain boolean matrices called *triorthogonal* matrices, i.e. those where the Hamming

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 79–95, doi:10.4204/EPTCS.406.4

weight of the product of any $\ell \leq 3$ columns is zero modulo 2^ℓ. Triorthogonal matrices have been used extensively in the study of transversal gates for quantum error-correcting codes and magic state distillation [8, 7, 22]. This connection is not really new, and could probably best be described as folklore, as it really just explicitly connects the dots between two important results in the literature: the equivalence between T-count optimisation and Reed-Muller decoding [2] and the equivalence between triorthogonal matrices and certain Reed-Muller codewords [22].

We show that the equivalence between spider nest identities and triorthogonal matrices can be made explicit and fully graphical, with the help of the *scalable ZX-calculus* [11]. This extension to ZX notation enables one to work with entire registers of qubits at once to represent arbitrary connectivity between components using boolean biadjacency matrices. Notably, all spider nest equations take a simple form, for some triorthogonal matrix M:



(2)

We show that we only have to assume one further rule, a variant of Eq. (1), beyond the standard Clifford ZX-calculus in order to prove all identities of the form of Eq. (2) (cf. Theorem 3.4). This in turn gives us a complete calculus for CNOT+T circuits (cf. Corollary 3.6).

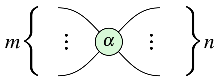
The graphical form for spider-nest identities (2) is particularly handy when used in the context of error correcting codes. As shown in the first "graphical grokking" paper [18], the encoding map of a CSS code can be represented as a ZX-diagram in a certain normal form. By "pushing" maps through the encoder, one can compute the effect of a physical map on the logical qubits or vice-versa. Of particular importance for fault-tolerant computation are those logical maps that can be implemented by a *transversal* operation on the physical qubits, which in the setting we will consider are operations implemented by a tensor product of single-qubit unitaries. Codes with rich sets of transversal logical gates are interesting both for supporting computation on their own encoded qubits and as part of *magic state distillation* protocols, used to boost (non-universal) fault-tolerant computation in other codes to universality [20].

Many characterisations of transversal gates exist in the literature for specific classes of gates and/or codes [8, 7, 10, 25, 27]. The one we give here is essentially equivalent to the one given in [28] concerning diagonal gates in the Clifford hierarchy, although we will focus on just the third level of the Clifford hierarchy for the sake of simplicity. Namely, for a fixed CSS code, we give a complete classification for the set of transversal gates whose logical action is in $\mathscr{D}_3$, the diagonal unitaries on the third level of the Clifford hierarchy (Theorem 4.1). While we focus on $\mathscr{D}_3$, the method we show translates straightforwardly to $\mathscr{D}_\ell$, the diagonal unitaries of the ℓth level of the Clifford hierarchy, and phase gadgets with a $\pi/2^{\ell-1}$ phase, as we will remark in the conclusion. A notable feature of our characterisation is not so much the result itself, but the proof technique, which demonstrates the interplay between the CSS code and the triorthogonal structure that needs to be present in it. Both of these can be treated uniformly using scalable notation, and the graphical rules allow one to easily see (and hopefully grok) how the stabiliser and non-stabiliser aspects of the computation interact.

1 Preliminaries

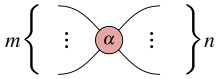
The basic building blocks of ZX-diagrams are spiders, which come in two varieties, *Z spiders* and *X spiders*, defined respectively relative to the eigenbases of the Pauli Z and Pauli X operators.

$$Z^n_m[\alpha] \quad := $$



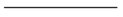
$$= \quad |0\rangle^{\otimes n}\langle 0|^{\otimes m} + e^{i\alpha}|1\rangle^{\otimes n}\langle 1|^{\otimes m}$$

$$X^n_m[\alpha] \quad := $$

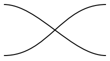


$$= \quad |+\rangle^{\otimes n}\langle +|^{\otimes m} + e^{i\alpha}|-\rangle^{\otimes n}\langle -|^{\otimes m} \qquad (3)$$

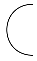
where $\langle\psi|^{\otimes m}$ and $|\psi\rangle^{\otimes n}$ are the m- and n-fold tensor products of bras and kets, respectively, and we take the convention that $(...)^{\otimes 0} = 1$. The parameter α is called the *phase* of a spider. If we omit the phase, it is assumed to be 0. In addition to spiders, we allow identity wires, swaps, cups, and caps in ZX-diagrams, which are defined as follows:



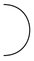
$$:= \sum_i |i\rangle\langle i|$$



$$:= \sum_{ij} |ij\rangle\langle ji|$$



$$:= \sum_i |ii\rangle$$

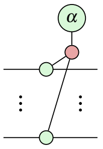


$$:= \sum_i \langle ii|$$

If all of the angles in a ZX-diagram are integer multiples of $\pi/2$, it is called a *Clifford ZX-diagram*. If not, it is called a *non-Clifford ZX-diagram*. There is a direct translation from Clifford+phase circuits to ZX-diagrams, where the resulting diagram is Clifford if and only if the circuit contains no non-Clifford phase gates.

ZX-diagrams have the useful property that they are invariant under arbitrary deformations and swapping input/output wires of spiders. This property is sometimes referred to as *only connectivity matters*. In addition to this "meta-rule", the *Clifford ZX-calculus* consists of the 7 rules shown in Figure 1. Notably, this set of rules is *complete* for Clifford ZX-diagrams. That is, if two Clifford ZX-diagrams describe the same linear map, one can transform one into the other using the Clifford ZX-calculus. For the Clifford ZX-calculus, this transformation is furthermore efficient. Note that we presented here the rules only up to non-zero scalar factor (denoted by $\propto$). Scalars will not play an important role in this paper.

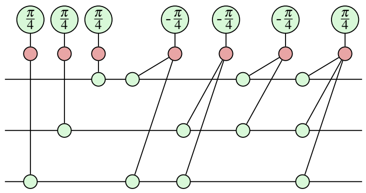
In addition to encodings of basic gates, a useful unitary ZX-diagram is a *phase gadget*, which applies a relative phase of α to all of the computational basis states whose bitstring has parity 1 [19]:



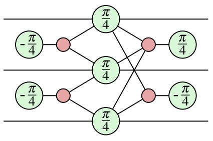
$$:: \quad |x_1 \dots x_n\rangle \mapsto e^{i\alpha \cdot x_1 \oplus \dots \oplus x_n}|x_1 \dots x_n\rangle$$

Phase gadgets with arbitrary angles applied to arbitrary subsets of qubits form a spanning set for all diagonal unitaries. By restricting phases to integer multiples of $\pi/2^{r-1}$, we obtain a spanning set for all diagonal unitaries on the r-th level of the Clifford hierarchy [15]. For example, we can represent CCZ, on the 3rd level of the Clifford hierarchy as a collection of $\pi/4$ phase gadgets:

$$\mathrm{CCZ} \; = $$



$$=$$



(4)

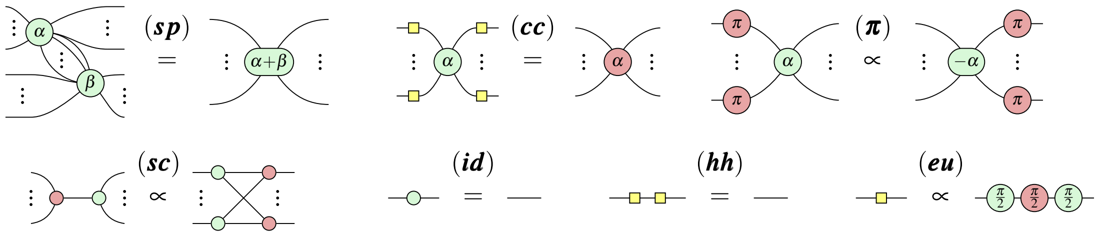
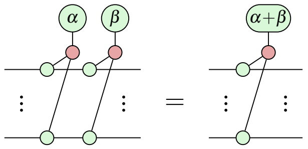


Figure 1: The Clifford ZX-calculus: spider fusion (***sp***), colour change (***cc***), π-copy (***π***), strong complementarity (***sc***), identity (***id***), H-cancellation (***hh***), and Euler decomposition of H (***eu***). Thanks to (***cc***), all rules hold with their colours reversed.

Note that we have here fused all the Z-spiders on the qubit wires together to write this circuit with multiple phase gadgets more compactly. Each individual gadget corresponds to a term in the associated phase polynomial $\phi : \mathbb{B}^n \to \mathbb{R}$, which represents the relative phases of basis elements as $\mathbb{R}$-linear combinations of parity functions. For example, CCZ can be represented as: $\mathrm{CCZ}|x_1x_2x_3\rangle = e^{i\cdot\phi}|x_1x_2x_3\rangle$ where $\phi(x_1,x_2,x_3) = \frac{\pi}{4}x_1 + \frac{\pi}{4}x_2 + \frac{\pi}{4}x_3 - \frac{\pi}{4}x_1 \oplus x_2 - \frac{\pi}{4}x_2 \oplus x_3 - \frac{\pi}{4}x_1 \oplus x_3 + \frac{\pi}{4}x_1 \oplus x_2 \oplus x_3$.

It is also worth noting that phase gadgets applied to the same set of qubits "fuse" in the sense that their angles add together, as a consequence of the rules in Figure 1:



(5)

This is called the *gadget fusion* rule [19].

1.1 Spider nests and triorthogonal matrices

When non-Clifford angles are assumed to be arbitrary free parameters, we are unlikely to find non-trivial equations between collections of phase gadgets beyond those already provable using the Clifford ZX-calculus (cf. [29]). However, if we assume that the non-Clifford angles take specific values, especially values of the form $\pi/2^r$ for some integer $r > 1$, more equations hold than just those provable by the Clifford ZX-calculus [2]. An important class of such rules are the *spider nest identities* [4]. These are certain configurations of non-Clifford phase gadgets whose overall action is the identity, up to a global phase.

To understand the collection of all such rules, we should get some structural understanding of what is actually happening. Consider first a single phase gadget and how it acts on the computational basis: $|x_1 \dots x_n\rangle \mapsto e^{i\alpha \cdot x_1 \oplus \dots \oplus x_n}|x_1 \dots x_n\rangle$. This action is totally determined by a phase polynomial $\phi : \mathbb{F}_2^k \to \mathbb{R}$ here given by $\phi(\vec{x}) = \alpha \bigoplus_j x_j$. There are two useful bases for expressing phase polynomials: the XOR basis and the monomial basis. The XOR basis represents ϕ as a real linear combination of functions of the form $f(\vec{x}) = \bigoplus_{i \in S} x_i$, where $S \subseteq \{1, \dots, n\}$. The monomial, or "AND" basis consists of functions of the form $g(\vec{x}) = \Pi_{i \in S} x_i$ for some subset $S \subseteq \{1, \dots, n\}$.

We can transform from the basis of XOR functions into the basis of AND functions by using the fact that $x \oplus y = x + y - 2x \cdot y$. From this, we can derive the n variable version:

$$x_1 \oplus \dots \oplus x_n = \sum_{S \subseteq [n]} (-2)^{|S|-1} \prod_{i \in S} x_i. \tag{6}$$

Transforming functions between these two bases is known as the *Boolean Fourier transform*.

Using Eq. (6) we can rewrite an $\mathbb{R}$-linear combination of XORs into a linear combination of monomials. Note that the prefactor $(-2)^{|S|-1}$ of each term grows as the degree of the monomial increases. In particular, if the coefficient of $x_1 \oplus \ldots \oplus x_n$ is some integer multiple of $\pi/4$, applying the inverse Fourier transform will result in linear terms whose coefficients are integer multiples of $\pi/4$, quadratic terms with multiples of $\pi/2$, cubic terms with multiples of π, and all terms of degree 4 or more being multiples of 2π, which will vanish:

$$e^{i\frac{\pi}{4}x_1\oplus\cdots\oplus x_n} = \exp\left(i\left(\frac{\pi}{4}\sum_j x_j - \frac{\pi}{2}\sum_{i<j} x_i x_j + \pi \sum_{i<j<k} x_i x_j x_k - 2\pi\cdots\right)\right) \tag{7}$$

Hence, each phase gadget corresponds to a collection of T gates (the linear terms), CS gates (quadratic terms) and CCZ terms (cubic terms). A collection of phase gadgets then corresponds to adding together their respective T, CS and CCZ gates. We then see that a collection of phase gadgets implements the identity precisely when all these T, CS and CCZ gates cancel each other out. This happens when each single variable x_i occurs 0 mod 8 times ($T^8 = I$), each pair of variables occurs 0 mod 4 times ($\mathrm{CS}^4 = I$), and each triple occurs 0 mod 2 times ($\mathrm{CCZ}^2 = I$).

We can formalise the cancellation property as follows. Represent a collection of $\pi/4$ phase gadgets as a set of bit strings $\vec{y}^1, \ldots, \vec{y}^m$, where $y_i^l = 1$ means the l-th phase gadget is connected to the i-th wire. Then for the compositions of all M phase gadgets to form an identity they need to satisfy:

$$\forall i : \sum_l y_i^l = 0 \pmod 8 \qquad \forall i < j : \sum_l y_i^l y_j^l = 0 \pmod 4 \qquad \forall i < j < k : \sum_l y_i^l y_j^l y_k^l = 0 \pmod 2$$

Writing these $\vec{y}^l$ as the rows of a binary matrix, these conditions specify precisely what it means for the matrix to be *triorthogonal*, namely that each column, product of pairs of columns and product of triples of columns needs to have a Hamming weight that is a multiple of respectively 8, 4 and 2 [22].

Example 1.1. The collection of phase gadgets in the LHS of equation (1) corresponds to the following triorthogonal matrix, whose rows are all the non-zero length 4 bit strings:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}^T \tag{8}$$

If we instead require the weaker condition that each of these properties holds just modulo 2, then we get the notion of a *semi-triorthogonal* matrix. In that case, we can abbreviate the 3 conditions into one, where we no longer require i, j, k to be distinct:

$$\forall i, j, k : \sum_l y_i^l y_j^l y_k^l = 0 \pmod 2 \tag{9}$$

These describe collections of phase gadgets that are equal to a Clifford, instead of exactly equal to the identity. This is because instead of $\sum_l y_i^l = 0 \pmod 8$, so that each qubit i has a multiple of 8 T gates that cancel out, we just have $\sum_l y_i^l = 0 \pmod 2$, so that we can pair up all the T gates on the qubit i, to combine them as $T^2 = S$, which is Clifford. Similarly, the equation $\sum_l y_i^l y_j^l = 0 \pmod 2$ means that we can pair up all the CS gates on qubits i and j to produce $\mathrm{CS}^2 = \mathrm{CZ}$, which is also Clifford. We get the same for the CCZ gates, which pair up into an identity: $\mathrm{CCZ}^2 = I$.

We summarise the above discussion in the following theorem.

Theorem 1.2. Let M be a boolean matrix with n columns and k rows and define the unitary U_M as the circuit consisting of k $\frac{\pi}{4}$ phase gadgets, where the connectivity of the jth gadget is described by the jth row of M. Then U_M is Clifford if and only if M is semi-triorthogonal and U_M is the identity if and only if M is triorthogonal.

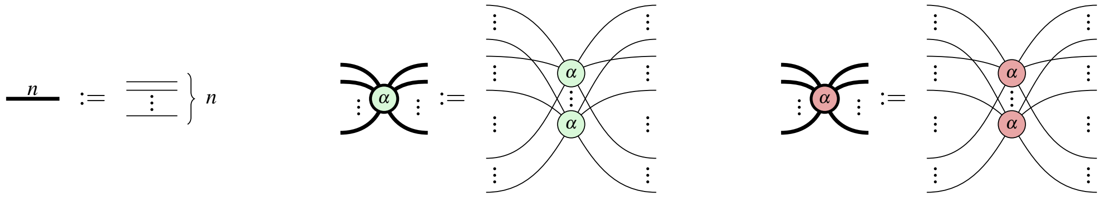
Remark 1.3. Several inequivalent definitions of the term “triorthogonal” appear in the literature. It is commonly used to describe the weaker condition (9), or an even weaker condition that only requires products of pairs and triples of distinct rows to have even Hamming weight. The stronger condition we call *triorthogonal* is also sometimes called 3-even [27]. Our terminology matches e.g. [20], with the slight difference that we impose conditions on the columns of a matrix rather than the rows, as it will make some calculations simpler.

There is clearly a close relationship between the graphical concept of spider nest identities and the (non-graphical) concepts of triorthogonal matrices and low-degree polynomials. To make this formal, it will be useful to have a bridge between the graphical notation and matrices, which thankfully is provided by the scalable ZX-calculus.

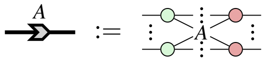
1.2 The scalable ZX-calculus

While plain ZX-diagrams are convenient for doing many concrete calculations, it will be convenient when discussing quantum error correcting codes and transversal gates to adopt the *scalable ZX notation* [11]. This notation enables us to compactly represent operations on registers of many qubits, while still maintaining much of the flavour of calculations with standard ZX-diagrams.

We represent a register of qubits as a single thick wire and the product of n (unconnected) copies of a Z or X spider as a bold spider:

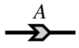


In [11], the authors allowed bold spiders to be labelled by lists of phases $\vec{\alpha} \in \mathbb{R}^n$, enabling each copy to have a different phase. For our purposes, we won’t need this extra generality, so a bold spider labelled by $\alpha \in \mathbb{R}$ corresponds to n spiders all with phase α. The authors of [11] also introduced explicit maps called *dividers* and *gathers* for splitting a register of $m+n$ qubits into two registers of m and n qubits and vice-versa. For our purposes, we will leave these maps implicit. The most important new generator is the “arrow”, which allows us to represent arbitrary connectivity from m Z-spiders to n X-spiders using an $n \times m$ biadjacency matrix $A \in \mathbb{F}_2^{n\times m}$:



(10)

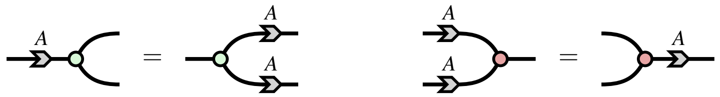
Taking the convention that A_i^j represents the entry in the i-th column and j-th row of the matrix A, we have in Eq. (10) that $A_i^j = 1$ if and only if the i-th Z-spider on the left is connected to the j-th X-spider on the right. Concretely, Eq. (10) corresponds, up to scalar factors, to a linear map that acts as A on computational basis vectors:



$:: \ |\vec{b}\rangle \mapsto |A\vec{b}\rangle$

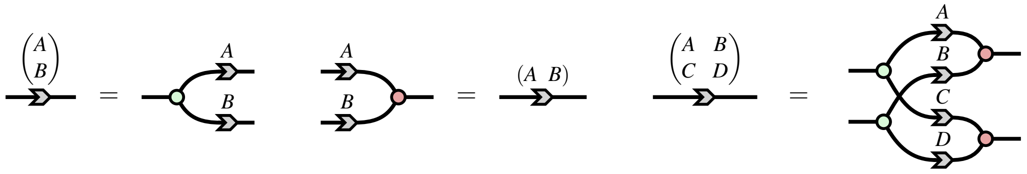
Note that we treat the bitstring $\vec{b}$ as a column vector for the purposes of matrix multiplication.

Spiders and arrows satisfy several rules that will prove useful. First, we have two "copy" laws relating arrows to Z/X spiders:



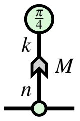
(11)

Second, we can express block diagonal matrices in terms of spiders:


$\begin{pmatrix} A \\ B \end{pmatrix}$
=
A
B
A
B
=
$(A\ B)$
$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$
=
A
B
C
D
(12)

2 Inductive construction of spider nest identities

We can now use the scalable notation to relate spider nest identities to certain families of triorthogonal matrices. The first thing to note is that for any boolean matrix M, we can write the associated n-qubit diagonal unitary U_M from Theorem 1.2, composed of a $\pi/4$ phase gadget for each of the k rows of M, as follows:

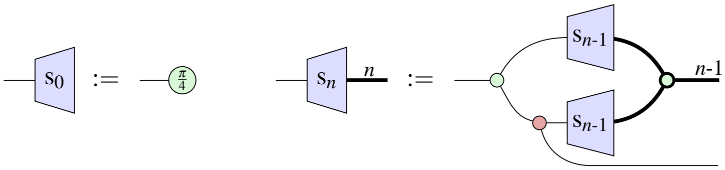
$D_M =$


(13)

M has n columns corresponding to the n qubits of D_M and k rows, corresponding to k phase gadgets. The i-th row of M then says which qubits are connected to the i-th phase gadget. Hence, a matrix M is triorthogonal if and only if $D_M = I$. We write here D_M instead of U_M to refer to the specific diagram in Eq. (13).

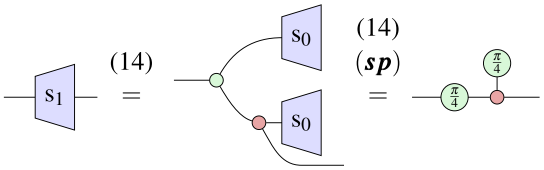
Notably, this gives us an infinite family of graphical equations, of the form $D_M = I$ for all triorthogonal matrices M. In fact, this is precisely the set of all spider nest identities, which we justified by the concrete calculations involving the inverse Fourier transform in Section 1.1. We know by completeness of the Clifford+T ZX-calculus [17] that all of these equations are provable by an extended version of the ZX-calculus. However, from those rules, it is very difficult to see how one could directly reduce a diagram D_M to I for some fixed M, and whether that reduction could be done efficiently (i.e. without expanding to a large normal form before reducing back down). Hence, it is interesting to ask just how much we need to add to the simple Clifford rules in Figure 1 in order to prove the entire family of spider nest identities directly. Toward that goal, we will now inductively construct a family of maps that will enable us to generate all the $\pi/4$ spider nest identities.

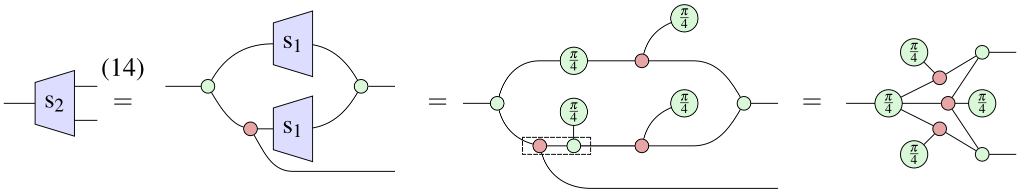
Definition 2.1. The *spider-nest maps* $s_n : 1 \to n$ are constructed inductively as follows:



(14)

Intuitively, this inductive definition results in a phase gadget connecting the single input wire to every subset of the output wires. For example:





where the last step follows from applying the strong complementarity rule (***sc***) to the marked spider pair, and then applying spider fusion (***sp***) as much as possible.

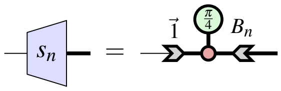
We now formalise this intuitive explanation of s_n using scalable notation. Let B_n be the $n \times 2^n$ matrix whose 2^n rows consist of all n-bitstrings. That is, the matrix defined inductively as follows:

$$B_0 = () \qquad\qquad B_n = \begin{pmatrix} B_{n-1} & \vec{0} \\ B_{n-1} & \vec{1} \end{pmatrix}$$

where $\vec{0}$ and $\vec{1}$ are respectively the column vectors of all 0's and all 1's. For example, we have:

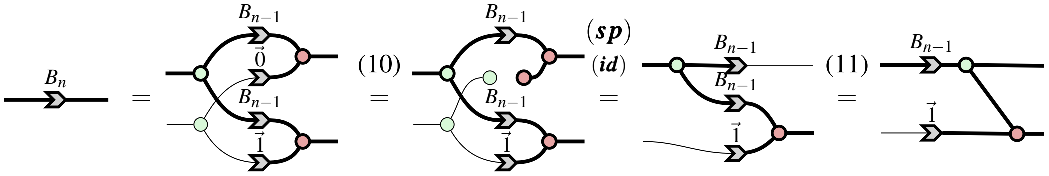
$$B_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad\qquad B_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \tag{15}$$

Theorem 2.2. For all n, we have:



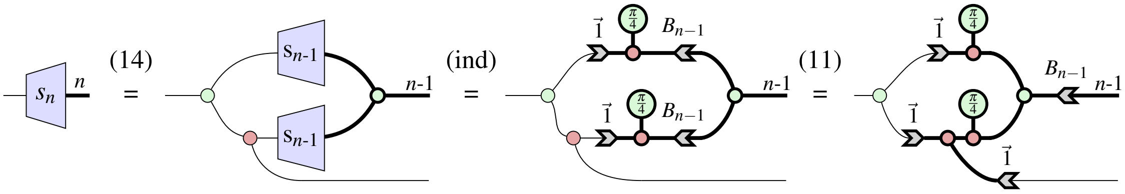
(16)

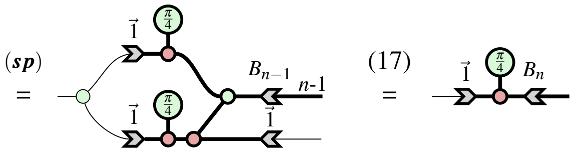
Proof. First, note that:



(17)

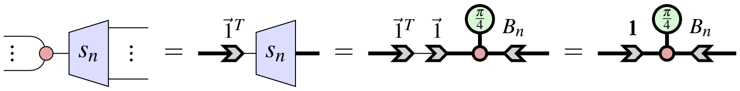
Using this equation and the scalable rules, we can prove (16) from (14) by induction on n:





Here in the last step we gathered together the wires connecting to the X-spiders. This works because the matrix arrows with $\vec{1}$ are just a single Z-spider fully-connected to identity X-spiders on the right. These fuse with the surrounding Z- and X-spiders to produce the right result. □

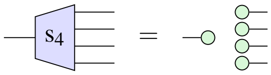
If we connect this s_n generator to an X-spider on the left, we obtain the following:



where **1** is the matrix where every entry is 1. We will see in the next section how this map lets us generate all of the spider nest identities.

3 Proving all spider nest identities

In this section, we will show that, by adding just one rule to the Clifford ZX-calculus, we can prove all spider nest equations. We call this one extra equation the *S4-rule*:



(18)

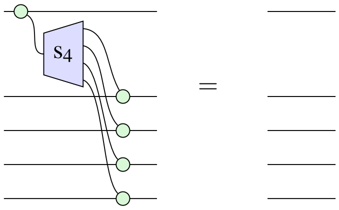
We will show this is sound, but we will first need some more definitions. In particular, we will start by passing to an alternative characterisation for (semi-)triorthogonal matrices, in terms of polynomials of low degree. This result seems to be well-known, and appears in various guises in the literature, e.g. when applying Reed-Muller codes to T-count minimisation or magic state distillation. The version we give here is a variation on one given by Nezami and Haah [22].

Definition 3.1. For a boolean matrix M with n columns, its *indicator polynomial* $P_M \in \mathbb{F}_2[x_1,\dots,x_n]$ sends a bitstring $(b_1,\dots,b_n)$ to 1 if and only if that bitstring appears as a row in M an odd number of times.

Theorem 3.2. A matrix M with n columns is semi-triorthogonal if and only if its indicator polynomial P_M is of degree at most $n-4$.

The proof is straightforward, but relies on some basic facts about Reed-Muller codes. We give these and the proof of Theorem 3.2 in Appendix A.

We can now show that S4 is indeed sound. First, note that it is equivalent up to the standard Clifford rules to the following rule:

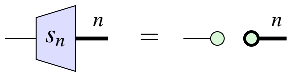


(19)

Here the left-hand side consists of $2^4 = 16$ phase gadgets, which are all connected to the first qubit, and have all possible connections to the bottom four qubits. Its connectivity matrix is hence $M = (\vec{1}\ \ B_4)$. A bitstring $x_1x_2x_3x_4x_5$ hence appears as a row in M iff $x_1 = 1$. Its indicator polynomial is then $P(x_1,x_2,x_3,x_4,x_5) = x_1$. This is a degree 1 polynomial on 5 variables, and hence satisfies the condition of Theorem 3.2, so that M is semi-triorthogonal. We can also manually verify that M is in fact triorthogonal, so that Eq. (19) is indeed correct.

Returning to the S4 rule, we see that, thanks to the inductive definition of s_n, not only does s_4 separate, but so do all s_n for $n \geq 4$.

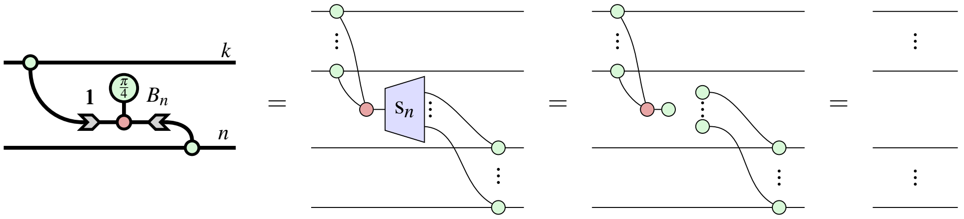
Lemma 3.3. For $n \geq 4$, the Clifford ZX-calculus augmented with the S4 rule implies:



(20)

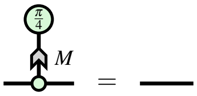
Proof. By induction on n. The base case $n = 4$ is (18). For $n > 4$, unfold s_n using (14) and apply the induction hypothesis. □

We see then that if we include s_n for $n \geq 4$ in a circuit, that we obtain:



(21)

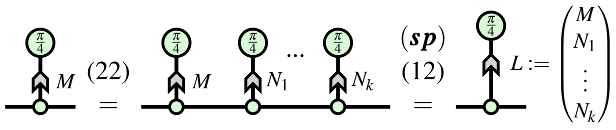
The lefthand-side above consists of the set of all phase gadgets that are connected to the first k wires and all combinations of the last n wires. Writing these phase gadgets as the rows of a matrix, this is $M = (\mathbf{1} \;\; B_n)$. The indicator polynomial of this matrix P satisfies the condition that $P(x_1, \ldots, x_{n+k}) = 1$ if and only if $x_j = 1$ for $j \in \{1, \ldots, k\}$. Hence, it is the monomial $P = x_1 \ldots x_k$. By permuting wires, we can obtain any monomial on $m = k + n$ variables, and as long as the degree $k \leq m - 4$, then equation (21) is satisfied. Hence, in particular, if M is a boolean matrix with m columns and no duplicate rows whose indicator polynomial is of degree at most $m - 4$, then the Clifford rules together with the S4 rule proves:



(22)

Theorem 3.4. The Clifford ZX-calculus, plus the S4 rule (18) are *diagonally complete* for Clifford+T ZX-diagrams. That is, for any boolean matrices M, N, if $[\![D_M]\!] = [\![D_N]\!]$, the D_M can be transformed into D_N using only the rules in Figure 1 and S4.

Proof. Proving $D_M = D_N$ is equivalent to proving $D_M D_N^\dagger = I$. First note that $D_N^\dagger$ is equal to D_N up to a Clifford diagram, as $\frac{\pi}{4}$ phase gadgets and $-\frac{\pi}{4}$ phase gadgets differ by a Clifford. The circuit for $D_M D_N^\dagger$ consists of a combination of phase gadgets, which can be combined into a single matrix describing the connectivity of the gadgets. Duplicate rows can always be cancelled by applying the gadget fusion rule (5). Hence, to prove completeness it suffices to prove that when $[\![D_M]\!] = [\![I]\!]$ we can rewrite D_M into I. Concretely, $D_M = I$ is true if and only if M is triorthogonal. Let P_M be its indicator polynomial, which will have degree at most $n - 4$ for n the number of qubits (Theorem 3.2). Let $N_1, \ldots, N_k$ be matrices whose indicator monomials are $P_1, \ldots, P_k$ for $P_M = \sum_j P_j$. Since these all have degree $\leq n - 4$, we can show using Eq. (22) that:



Then, the indicator polynomial of L is $P_M + \sum_j P_j = 0$. Hence, every row in L appears an even number of times. Using gadget-fusion, we can therefore reduce all angles to integer multiples of $\pi/2$. We can then apply Clifford-completeness to reduce to the identity. □

Remark 3.5. Note that while S4 makes the ZX-calculus diagonally complete for Clifford+T ZX-diagrams, this rule set is not complete for all Clifford+T diagrams. To see this, one can check that S4 is sound for the $[\![-]\!]^\sharp$ interpretation given in [24] and hence cannot derive e.g. the supplementarity law for non-Clifford angles.[1]

One way to think of the Clifford+S4 ZX-calculus is as the ZX analogue of the equational presentation for CNOT-Dihedral circuits of Amy *et al* [1]. Indeed we can get the following as a corollary of Theorem 3.4.

Corollary 3.6. The Clifford rules plus S4 are complete for CNOT+T circuits.

Proof. For CNOT+T circuits U,V where $[\![U]\!] = [\![V]\!]$, it suffices to show that $U^\dagger V$ can be rewritten to the identity. Using just Clifford rules, it is possible to rewrite any CNOT+T circuit into a layer of phase gadgets D_M followed by a CNOT circuit C. Since $[\![U^\dagger V]\!] = I$, it must be the case the $[\![D_M]\!] = [\![C]\!] = I$. Hence, we can use the completeness of phase-free ZX to rewrite C into I and Theorem 3.4 to rewrite D_M into I. □

A Clifford+T circuit can be written as a composition of CNOT, T and Hadamard gates. Hence, the results above apply to those Clifford+T circuits without Hadamards. For any circuit with Hadamards, we can "split it up" at those points and reason about the CNOT+T circuits separately. This technique is used for instance when doing T-count optimisation [4, 16, 26], where the circuit is either split up on the Hadamards, or the Hadamards are made into CZ gates by the use of an ancilla and measurement [16]. Note that there do seem to be Clifford+T circuit rewrite rules that specifically involve Hadamard gates [14] that seem likely to not be provable just using Clifford rules and spider-nest identities.

Remark 3.7. Theorem 3.4 and Corollary 3.6 should be compared to the results in [5, 21]. There they show that assuming the 15 T gate spider-nest Eq. (1) they can prove the family of spider nests consisting of a single high-degree phase gadget combined with many phase gadgets of degree 3 and lower. However, they only use these rules for circuit optimisation and do not discuss completeness.

4 Characterisation of transversal $\mathscr{D}_3$ gates for CSS codes

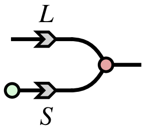
We can use our representation of spider-nest identities in the scalable ZX-calculus to prove a characterisation of CSS codes with transversal diagonal logical operations in the third level of the Clifford hierarchy.

A *CSS code* is a stabiliser code that has a generating set of stabilisers consisting of pure X stabilisers and pure Z stabilisers (i.e. that are respectively tensor products of I and X, or I and Z). Its logical X (resp. Z) operators also purely consist of X (resp. Z) operators. The Z stabilisers and logical operators are in fact completely determined by the X stabilisers and logical operators (up to choice of generator), and hence we only have to specify the "X part" of the CSS code.

We can hence fully specify a $[\![n,k,d]\!]$ CSS code—i.e. one that has n physical qubits, k logical qubits, and is distance d— by fixing $k+r \leq n$ linearly independent vectors in $\mathbb{F}_2^n$ corresponding to k logical X operators and r X stabilisers. Here a 1 at position i in the vector denotes an X on the ith qubit [18]. Letting L and S be the matrices that have these vectors as their columns (L standing for "logical" and S for "stabiliser"), the vectors corresponding to the remaining $n-k-r$ Z checks can be recovered by fixing a basis for $\text{span}(\text{cols}(L) \cup \text{cols}(S))^\perp$. Following [18], we can write the encoder for a CSS code as

[1] The authors wish to thank Richie Yeung for pointing this out.

a phase-free ZX-diagram. If we choose to write it in Z-X normal form, the encoder consists of a row of Z-spiders at the input and a row of X-spiders at the output. For each logical operator, an input wire connects to a Z-spider, which then connects to X-spiders on the output corresponding to the support of the operator. For each X stabiliser, an additional Z-spider with no inputs connects to output X-spiders, according to the support of the operator. While this is a bit unwieldy to say in words, the encoder can be written in terms of the matrices L and S straightforwardly as follows:



Indeed this corresponds to the linear map that sends basis vectors $|\vec{b}\rangle \in (\mathbb{C}^2)^{\otimes k}$ of the logical space to their associated codewords $\sum_{\vec{c}} |L\vec{b} + S\vec{c}\rangle \in (\mathbb{C}^2)^{\otimes n}$.

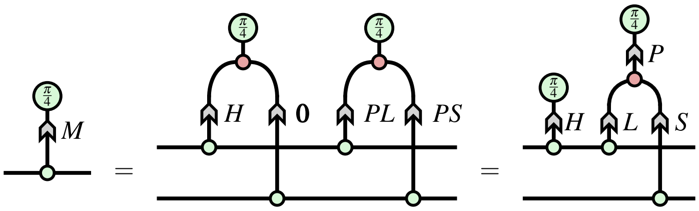
Note that diagonal unitaries of the form D_P correspond to transversal applications of powers of T gates if and only if the rows of P each have Hamming weight 1 (as T gates are phase gadgets connected to just a single qubit). With this, we are now ready to state our characterisation result.

Theorem 4.1. A CSS code with X-logical operators and X-stabilisers L and S admits a transversal implementation of a gate $D_H^\dagger \in \mathscr{D}_3$ if and only if there exists a matrix P whose rows have Hamming weight 1 such that the matrix

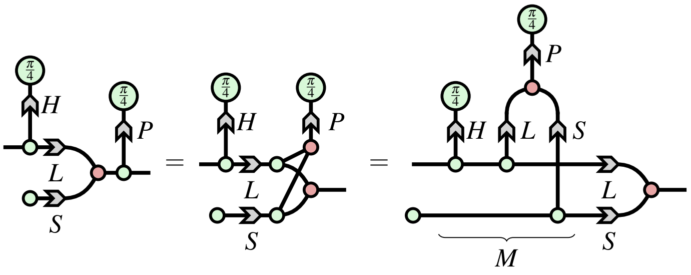
$$M = \left(\begin{array}{c|c} H & 0 \\ \hline PL & PS \end{array}\right)$$

is triorthogonal.

Proof. First, we will apply the scalable ZX rules to graphically decompose the block form of M:



For E the encoder associated with the CSS code (L,S), the code implements $D_H^\dagger$ if and only if for some P we have $D_P E D_H = E$. We begin by "pushing" D_P through the encoder:



If M is triorthogonal, the part marked M in the diagram above will vanish and we are left with the encoder E. If it is not triorthogonal, then $D_M \neq I$. Since D_M is diagonal, it follows that $D_M|+\ldots+\rangle \neq |+\ldots+\rangle$, so $D_P E D_H|+\ldots+\rangle \neq E|+\ldots+\rangle$. Hence, $D_P E D_H \neq E$. □

Remark 4.2. For simplicity, in Theorem 4.1 we took the meaning of "transversal" to be "consisting of single-qubit operations". However, there is a more general notion of transversality that classifies operations e.g. between two or more copies of a code. For instance, if a CSS code is the tensor product of two identical CSS codes, then implementing pairwise CNOT gates between the corresponding qubits of the two codes will implement a logical transversal CNOT. This kind of operation is still called transversal because each CNOT gate only involves a single qubit in each copy of the code, i.e. in each *code block*. If our code actually consists of b distinct code blocks, we can accomodate this more general notion of transversality by replacing the condition that the rows of P have Hamming weight 1 by a condition that P can be factored across code blocks as $P = \begin{pmatrix} P_1 & P_2 & \cdots & P_b \end{pmatrix}$ where each of the P_i has rows of hamming weight at most 1.

Note if we take $P = I$ (corresponding to a single T gate on each physical qubit), then from any triorthogonal matrix of the form:

$$M = \left(\begin{array}{c|c} H & 0 \\ \hline L & S \end{array}\right)$$

we can read off a CSS code (L,S) with a transversal implementation of $D_H^\dagger$. Many notable CSS codes with transversal gates can be seen as instances of this construction. For the examples below we write $RM(k,n)$ for the *Reed-Muller* code space of polynomials on n variables of degree at most k.

Example 4.3. The degree-1 monomial $x_1 \in \mathrm{RM}(1,5)$ gives the following triorthogonal matrix:

$$\left(\begin{array}{c|ccccccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)^T$$

The matrices (L,S) on the right define a $[\![15,1,3]\!]$ quantum Reed-Muller code. Reading H from the top-left, we see that it has a transversal implementation of $T^\dagger$. This property is used as the basis of the original 15-to-1 magic state distillation protocol given by Bravyi and Kitaev [8].

Example 4.4. The constant polynomial $1 \in \mathrm{RM}(0,4)$ gives us a triorthogonal matrix whose columns are all the 4-bitstrings. If we partition the matrix as follows:

$$\left(\begin{array}{cccccccc|cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)^T$$

then (L,S) defines the X-logical operators and X-stabiliers of an $[\![8,3,2]\!]$ colour code, dubbed the "smallest interesting colour code" [9]. In the upper left, we see the phase gadgets of a CCZ gate as in equation (4), hence this code admits a transversal $\mathrm{CCZ}^\dagger = \mathrm{CCZ}$.

While Theorem 4.1 gives a complete characterisation for the transversal gates in a CSS code, it is not obvious from the statement whether it can be used to efficiently find such gates. However, following a technique similar to [28], this is in fact possible. In fact, there are three related problems: (1) for fixed logical H find transversal gates P, (2) for fixed P find H, and (3) compute a generating set of all logical gates and their associated transversal implementations. All three of these problems can be posed as a system of linear equations over $\mathbb{Z}_8$, which as noted in [28], can be solved in polynomial time using the Howell normal form of a matrix over a ring. We sketch this procedure in Appendix B.

5 Conclusion

We have shown that spider-nest identities can be captured directly in terms of their associated triorthogonal matrices using the scalable ZX-calculus. Combining this fact with the graphical encoders for CSS codes introduced in [18] gives us a succinct and easy to digest characterisation for the set of transversal logical gates in $\mathscr{D}_3$ supported by a CSS code. The results we have shown here can be straightforwardly extended up the diagonal Clifford hierarchy, from $\mathscr{D}_3$ to $\mathscr{D}_\ell$. To do this, we can generalise the inductive family s_n in Definition 2.1 to $s_n^{(\ell)}$, whose base case $s_0^{(\ell)}$ has a $\pi/2^{\ell-1}$ phase. Then the analogue to the S4 equation is that $s_{\ell+1}^{(\ell)}$ separates. Interestingly, if we assume this equation for any fixed ℓ, it automatically follows for any $\ell' \leq \ell$. The rest of the proof goes through by noting that Theorem 3.2 generalises from triorthogonal to ℓ-orthogonal matrices and degree $n-\ell-1$ polynomials. Hence, we can prove all spider nest identities up the ℓ-th level of the Clifford hierarchy just by adding one rule. However, it seems that proving spider nests at *all* levels of the Clifford hierarchy still requires infinitely many rules. Note that this argument also holds for $\mathscr{D}_2$, so that we also characterise all the transversal diagonal Clifford unitaries. The proofs then don't require any additional spider-nest identity since the calculus is already complete for Cliffords.

Natural next steps are looking at non-diagonal transversal gates or non-CSS stabiliser codes. The story for diagonal gates in general stabiliser codes is relatively straightforward, given any stabiliser encoder can be decomposed into a CSS part and a diagonal part (cf. [28], Appendix C). There seems to be a nice graphical story there as well, relating to normal forms of Clifford ZX-diagrams, but we leave this, along with further explorations of the graphical structure of non-CSS codes, as future work. It also remains an open question whether Clifford+S4 (or Clifford+Sℓ) rules can be extended naturally to a complete set of equations for the appropriate class of ZX-diagrams. While other complete axiomatisations exist, constructing the rules this way could provide new insights into working effectively with quantum computations at higher levels of the Clifford hierarchy.

References

[1] Matthew Amy, Jianxin Chen & Neil J. Ross (2018): *A Finite Presentation of CNOT-Dihedral Operators*. 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. 84–97, doi:10.4204/EPTCS.266.5.

[2] Matthew Amy & Michele Mosca (2019): *T-count optimization and Reed-Muller codes*. *Transactions on Information Theory*, doi:10.1109/TIT.2019.2906374. Available at `https://ieeexplore.ieee.org/document/8672175`.

[3] 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.

[4] 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.

[5] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Techniques to Reduce $\pi/4$-Parity-Phase Circuits, Motivated by the ZX Calculus*. 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. 131–149, doi:10.4204/EPTCS.318.9.

[6] Coen Borghans (2019): *ZX-calculus and quantum stabilizer theory*. Master's thesis, Radboud University.

[7] Sergey Bravyi & Jeongwan Haah (2012): *Magic-state distillation with low overhead*. *Physical Review A* 86(5), p. 052329, doi:10.1103/PhysRevA.86.052329.

[8] Sergey Bravyi & Alexei Kitaev (2005): *Universal quantum computation with ideal Clifford gates and noisy ancillas*. *Physical Review A* 71(2), p. 022316, doi:10.1103/PhysRevA.71.022316.

[9] Earl Campbell: *The smallest interesting colour code*. `https://earltcampbell.com/2016/09/26/the-smallest-interesting-colour-code/`.

[10] Earl T Campbell & Mark Howard (2017): *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost*. *Physical Review A* 95(2), p. 022316, doi:10.1103/PhysRevA.95.022316.

[11] 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. Available at `http://drops.dagstuhl.de/opus/volltexte/2019/10999`.

[12] Bob Coecke & Ross Duncan (2008): *Interacting quantum observables*. In: *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, doi:10.1007/978-3-540-70583-3_25.

[13] 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.

[14] Bob Coecke & Quanlong Wang (2018): *ZX-rules for 2-qubit Clifford+ T quantum circuits*. In: *International Conference on Reversible Computation*, Springer, pp. 144–161, doi:10.1007/978-3-319-99498-7_10.

[15] Shawn X. Cui, Daniel Gottesman & Anirudh Krishna (2017): *Diagonal gates in the Clifford hierarchy*. *Physical Review A* 95(1), p. 012329, doi:10.1103/PhysRevA.95.012329.

[16] Luke E Heyfron & Earl T Campbell (2018): *An efficient quantum compiler that reduces T count*. *Quantum Science and Technology* 4(015004), doi:10.1088/2058-9565/aad604.

[17] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2020): *Completeness of the ZX-Calculus*. *Logical Methods in Computer Science*, doi:10.23638/LMCS-16(2:11)2020.

[18] Aleks Kissinger (2022): *Phase-free ZX diagrams are CSS codes (... or how to graphically grok the surface code)*. *arXiv preprint arXiv:2204.14038*, doi:10.48550/arXiv.2204.14038.

[19] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Physical Review A* 102, p. 022406, doi:10.1103/PhysRevA.102.022406.

[20] Daniel Litinski (2019): *A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery*. *Quantum* 3, p. 128, doi:10.22331/q-2019-03-05-128.

[21] Anthony Munson, Bob Coecke & Quanlong Wang (2021): *AND-gates in ZX-calculus: Spider Nest Identities and QBC-completeness*. In Benoît Valiron, Shane Mansfield, Pablo Arrighi & Prakash Panangaden, editors: *Proceedings 17th International Conference on Quantum Physics and Logic, Paris, France, June 2 - 6, 2020*, *Electronic Proceedings in Theoretical Computer Science* 340, Open Publishing Association, pp. 230–255, doi:10.4204/EPTCS.340.12.

[22] Sepehr Nezami & Jeongwan Haah (2022): *Classification of small triorthogonal codes*. *Physical Review A* 106(1), p. 012437, doi:10.1103/PhysRevA.106.012437.

[23] Kang Feng Ng & Quanlong Wang (2017): *A universal completion of the ZX-calculus*. Preprint, doi:10.48550/arXiv.1706.09877. arXiv:1706.09877.

[24] 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.

[25] Narayanan Rengaswamy, Robert Calderbank, Michael Newman & Henry D Pfister (2020): *On optimality of CSS codes for transversal T*. *IEEE Journal on Selected Areas in Information Theory* 1(2), pp. 499–514, doi:10.1109/JSAIT.2020.3012914.

[26] Francisco JR Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco JH Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering et al. (2024): *Quantum Circuit Optimization with AlphaTensor*. *arXiv preprint arXiv:2402.14396*, doi:10.48550/arXiv.2402.14396.

[27] Christophe Vuillot & Nikolas P Breuckmann (2022): *Quantum pin codes*. *IEEE Transactions on Information Theory* 68(9), pp. 5955–5974, doi:10.1109/TIT.2022.3170846.

[28] Mark A Webster, Armanda O Quintavalle & Stephen D Bartlett (2023): *Transversal diagonal logical operators for stabiliser codes*. *New Journal of Physics* 25(10), p. 103018, doi:10.1088/1367-2630/acfc5f.

[29] John van de Wetering, Richie Yeung, Tuomas Laakkonen & Aleks Kissinger (2024): *Optimal compilation of parametrised quantum circuits*. *arXiv preprint arXiv:2401.12877*, doi:10.48550/arXiv.2401.12877.

A Indicator polynomials for triorthogonal matrices

Since it is possible to represent arbitrary functions as polynomials, we can think of a polynomial P in n variables as a vector $[P]$ in $\mathbb{F}_2^{2^n}$ where $[P]_{\vec{b}} = P(\vec{b})$. By restricting to polynomials of a fixed degree r, we obtain certain subspaces called Reed-Muller codes.

Definition A.1. The *Reed-Muller code* $\mathrm{RM}(r,m)$ is the linear subspace of $\mathbb{F}_2^{2^m}$ consisting of vectors of the form $[P]$ for some polynomial $P \in \mathbb{F}_2[x_1,\dots,x_m]$ of degree $\leq r$.

A classic property of Reed-Muller codes is their orthocomplements are also Reed-Muller codes. This fact will help establish a correspondence with triorthogonal matrices.

Theorem A.2. For any $r < m$, $\mathrm{RM}(r,m)^\perp = \mathrm{RM}(m-r-1,m)$.

Proof. First, we note that if a polynomial P of m variables has degree $< m$, then $\sum_{\vec{b}} P(\vec{b}) = 0 \pmod 2$. This is easy to check for monomials, as any monomial of degree $< m$ must omit some variable x_j, hence

$$\sum_{\vec{b}} P(\vec{b}) = \sum_{\vec{b},b_j=0} P(\vec{b}) + \sum_{\vec{b},b_j=1} P(\vec{b}) = 0 \text{ (mod 2)}$$

The result holds for all polynomials by $\mathbb{F}_2$-linearity of the map $[P] \mapsto \sum_{\vec{b}} P(\vec{b})$ (mod 2). Now, for any polynomial P of degree at most r and Q of degree at most $m-r-1$, PQ has degree at most $m-1$. Hence $[P]\cdot[Q] = \sum_{\vec{b}} PQ(\vec{b}) = 0$ (mod 2). This implies $\mathrm{RM}(m-r-1,m) \subseteq \mathrm{RM}(r,m)^\perp$. Since $\mathrm{RM}(r,m)$ has the monomials as its basis, $\dim(\mathrm{RM}(r,m)) = \sum_{d=0}^{r} \binom{m}{d}$. By manipulating binomial coefficients, we can see that:

$$\begin{gathered}\dim(\mathrm{RM}(r,m)) + \dim(\mathrm{RM}(m-r-1,m)) \\ = \sum_{d=0}^{r}\binom{m}{d} + \sum_{d=0}^{m-r-1}\binom{m}{d} = \sum_{d=0}^{r}\binom{m}{d} + \sum_{d=r+1}^{m}\binom{m}{d} = 2^m = \dim(\mathbb{F}_2^{2^m})\end{gathered}$$

so $\mathrm{RM}(m-r-1,m) = \mathrm{RM}(r,m)^\perp$. □

This enables us to show that semi-triorthogonal matrices are closely related to Reed-Muller codes. We give a proof here similar to the one given for unital triorthogonal spaces by Nezami and Haah [22].

Theorem A.3. A matrix M with n columns is semi-triorthogonal if and only if its indicator polynomial P_M is of degree at most $n-4$.

Proof. Let M' be a matrix obtained from M by removing all repeated pairs of rows. M is semi-triorthogonal if and only if M' is, and both matrices have the same indicator polynomial. Hence, we can assume without loss of generality that M has no repeated rows.

Now, let M have indicator polynomial $P = P_M$. Then $Q = x_j P$ is a polynomial with the property that $Q(\vec{b}) = 1$ if and only if $b_j = 1$ and $P(\vec{b}) = 1$, hence $\sum_{\vec{b}} Q(\vec{b})$ is equal to the Hamming weight of the j-th column of M. This also works for products of columns: for $R = x_i x_j x_k P$, $\sum_{\vec{b}} R(\vec{b})$ is equal to the Hamming weight of the element-wise product of the i, j and k-th rows. Noting that $\sum_{\vec{b}} R(\vec{b}) \pmod 2 = [x_i x_j x_k] \cdot [P]$, where $\cdot$ is the dot-product of vectors in $\mathbb{F}_2^{2^n}$, we see that $[P]$ must be orthogonal to all degree-3 monomials $[x_i x_j x_k]$. Since the latter span $\mathrm{RM}(3,n)$, $P \in RM(3,n)^\perp$, so by Theorem A.2 $P \in \mathrm{RM}(n-4,n)$. □

B Computing transversal logical operations efficiently

Following a technique similar to [28], we will show that for fixed (L,S) defining a CSS code, we can efficiently calculate matrices H and P making:

$$M = \left(\begin{array}{c|c} H & 0 \\ \hline PL & PS \end{array}\right)$$

triorthogonal.

To see this, first note that we can span the space of all logical operators $D_H^\dagger$ using phase gadgets of degree at most 3, as larger phase gadgets can always be decomposed using spider nest identities [1]. Hence, we can replace H with QK, where Q is a matrix whose rows all have Hamming weight 1, and K has $O(k^3)$ rows consisting of all bitstrings of Hamming weight ≤ 3. Thus M becomes:

$$M = \left(\begin{array}{c|c} QK & 0 \\ \hline PL & PS \end{array}\right)$$

Since the order of rows in M is not relevant, the only function of P and Q is to fix the multiplicity of rows in the matrix:

$$N = \left(\begin{array}{c|c} K & 0 \\ \hline L & S \end{array}\right)$$

Suppose N is an $m \times n$ matrix. Form a new matrix $\widehat{N}$ over $\mathbb{Z}_8$ whose columns are labelled by the rows of N and whose rows are labelled by all sets of at most 3 columns of N. Then let:

$$\widehat{N}_{S,i} = \begin{cases} 2^{|S|-1} & \text{if } \forall j \in S. N_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases}$$

In other words, the i-th column of $\mathscr{M}$ corresponds to the inverse Fourier transform of the j-th row of N. Our goal is to find multiplicities for the rows of N such that the sum of the inverse Fourier transform of each phase gadget adds to 0 (mod 8) on all sets of 1, 2, or 3 qubits. In other words, we should find a *multiplicity vector* $\vec{m} \in \mathbb{Z}_8^m$ such that $\widehat{N}\vec{m} = \mathbf{0}$.

Given a multiplicity vector $\vec{m}$, we can make N into a triorthogonal matrix M by repacing each row i with $\vec{m}_i$ copies of that row. The set of all multiplicity vectors making N into a triorthogonal matrix is therefore the kernel of $\widehat{N}$.

As noted in [28], we can find a generating set for the kernel of a $\mathbb{Z}_8$ matrix in polynomial time using its Howell normal form. Similarly, we can fix multiplicities of the upper or lower half of the matrix and obtain an associated inhomogeneous system of equations, which can also be solved in polynomial time.

Multi-controlled Phase Gate Synthesis with ZX-calculus applied to Neutral Atom Hardware

Korbinian Staudacher[1]* Ludwig Schmid[2] Johannes Zeiher[3,4,5]
Robert Wille[2,6] Dieter Kranzlmüller[1,7]

*Korbinian.Staudacher@nm.ifi.lmu.de

[1]MNM-Team, Ludwig-Maximilians-Universität München, 80538 Munich, Germany
[2]Chair for Design Automation, Technical University of Munich, 80333 Munich, Germany
[3]Fakultät für Physik, Ludwig-Maximilians-Universität München, 80799 Munich, Germany
[4]Max-Planck-Institut für Quantenoptik, 85748 Garching, Germany
[5]Munich Center for Quantum Science and Technology (MCQST), 80799 Munich, Germany
[6]Software Competence Center Hagenberg GmbH (SCCH), 4232 Hagenberg im Mühlkreis, Austria
[7]Leibniz Supercomputing Centre (LRZ), 85748 Garching, Germany

Quantum circuit synthesis describes the process of converting arbitrary unitary operations into a gate sequence of a fixed universal gate set, usually defined by the operations native to a given hardware platform. Most current synthesis algorithms are designed to synthesize towards a set of single-qubit rotations and an additional entangling two-qubit gate, such as CX, CZ, or the Mølmer–Sørensen gate. However, with the emergence of neutral atom-based hardware and their native support for gates with more than two qubits, synthesis approaches tailored to these new gate sets become necessary. In this work, we present an approach to synthesize (multi-) controlled phase gates using ZX-calculus. By representing quantum circuits as graph-like ZX-diagrams, one can utilize the distinct graph structure of diagonal gates to identify multi-controlled phase gates inherently present in some quantum circuits even if none were explicitly defined in the original circuit. We evaluate the approach on a wide range of benchmark circuits and compare them to the standard Qiskit synthesis regarding its circuit execution time for neutral atom-based hardware with native support of multi-controlled gates. Our results show possible advantages for current state-of-the-art hardware and represent the first exact synthesis algorithm supporting arbitrary-sized multi-controlled phase gates.

1 Introduction

Compiling and optimizing quantum algorithms towards hardware-specific constraints is indispensable to efficiently use currently available noisy quantum hardware with limited gate fidelities and coherence times. An important step of the compilation process is quantum circuit synthesis, converting arbitrary unitary operations to gate sequences natively supported by the hardware. State-of-the-art synthesis algorithms, such as [25,47], are often focused on a superconducting hardware setting and synthesize towards singular two-qubit gates, e.g., CX, and single-qubit gates. Such synthesis algorithms are less preferable for other hardware architectures, for instance, when gates acting on three or more qubits can be executed natively without decomposition, resulting in a reduced execution cost.

In this work, we propose an approach to synthesize quantum circuits towards single qubit gates and arbitrary-sized multi-controlled phase gates $C_nP(\varphi)$. To this end, we make use of the representation of a quantum circuit as a graph-like ZX-diagram where we can use powerful rewrite rules of the ZX-calculus to simplify diagrammatic structures [15]. This approach has shown to be a useful tool for tasks like hardware-agnostic circuit optimization or equivalence checking [23,39,49]. We show that $C_nP(\varphi)$ have a distinct representation in graph-like ZX-diagrams as a combination of so-called phase gadgets, which occur naturally in the diagrams when using a simplification strategy proposed in [23]. By modifying an

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 96–116, doi:10.4204/EPTCS.406.5

existing extraction algorithm from [4] to translate graph-like diagrams back to quantum circuits, we can specifically optimize towards extracting phase gadget combinations corresponding to $C_nP(\varphi)$ gates.
The benefit and potential of the resulting approach are shown by synthesizing gate functionality for the recently emerging neutral atoms platforms [17, 20, 32, 41–44]. Besides dynamic connectivity with atom rearrangements [7, 8, 46] and favorable properties regarding scalability and large-scale control [6, 8, 19, 35, 38], this technology offers native support for multi-controlled gates such as C_nP, and CZ_n [12, 14, 16, 21, 22, 34]. We integrate our extraction scheme into a full gate synthesis and optimization process and compare total execution times on hardware against Qiskit synthesis routines, considering current state-of-the-art parameters. Our results show promising advantages in the form of reduced execution times on different benchmark circuits.
The paper is structured as follows: In the first part, we give a basic introduction to ZX-calculus, including graph-like ZX-diagram simplification, and show how multi-controlled phase gates can be identified and extracted from the diagrams. In the second part, we focus on the application of the proposed approach to neutral-atom-specific gate synthesis and discuss its effect on the execution time.

2 Related work

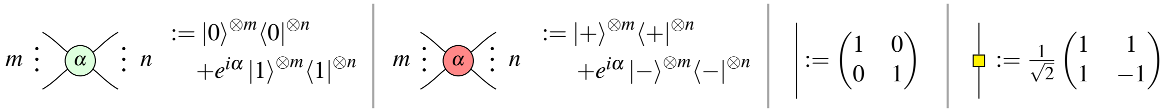
So far, algorithms supporting the synthesis of gates acting on more than two qubits are mostly centered around the generation of Toffoli gates. Ref. [18] introduces a synthesis algorithm for classical logic reversible functions using multi-control Toffoli gates and there exist algorithms for synthesizing towards universal Toffoli gate sets [3], even with optimal numbers of Toffoli gates [33]. The synthesis of multi-controlled phase gates is less studied. Ref. [57] proposes an optimal synthesis algorithm, but restricted to diagonal unitaries as an input. For universal circuits, a recent framework for neutral atom systems [37] is able to synthesize circuits with *CCZ* gates. However, the synthesis process is based on non-exact numerical optimization procedures and does not consider more than three-qubit gates or arbitrary rotations.

3 Preliminaries

In this section we introduce the ZX-calculus fundamentals and describe how graph-like diagrams can be simplified and extracted to quantum circuits, which represents the basis of our synthesis approach. We only give a brief overview of ZX-calculus, for a more detailed introduction we refer to [13, 15, 56].

3.1 ZX-calculus

ZX-calculus is a diagrammatic language for reasoning about linear maps in quantum computing where nodes (spiders) and edges (wires) form an undirected graph called ZX-diagram. There are two types of spiders: The green Z-spiders and the red X-spiders. Spiders can be parametrized with an angle $\alpha \in [0, 2\pi)$ and correspond to two-dimensional matrices in Hilbert space:



$$:= |0\rangle^{\otimes m}\langle 0|^{\otimes n} + e^{i\alpha}|1\rangle^{\otimes m}\langle 1|^{\otimes n} \qquad := |+\rangle^{\otimes m}\langle +|^{\otimes n} + e^{i\alpha}|-\rangle^{\otimes m}\langle -|^{\otimes n} \qquad := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad := \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Spiders can have any number of ingoing and outgoing wires and we can compose two diagrams either horizontally by joining the outputs of one diagram with the inputs of the other (denoted by $\circ$), or vertically by placing them side by side (denoted by $\otimes$). This corresponds to the known dot and tensor product in Hilbert space. For convenience, we distinguish between two types of wires: *Normal wires*, representing

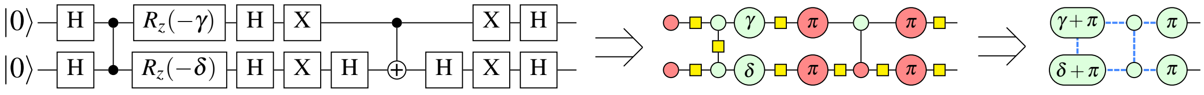
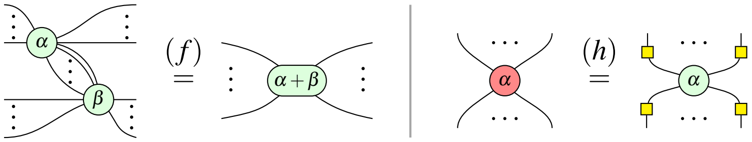


Figure 1: Translation of a two-qubit Grover search into a graph like ZX-diagram. Gates are replaced by their ZX-calculus counterpart and the diagram is made graph-like by repeated application of (f) and (h).

the identity, and *Hadamard wires*, representing the Hadamard matrix. Wires entering the diagram from the left are called input wires, with the adjacent spiders defined as inputs I, and wires exiting to the right are called output wires, with adjacent spiders defined as outputs O. We refer to the set of spiders $v \in I \cup O$ as *boundary spiders* and the complementary set of spiders $v \in V \setminus (I \cup O)$ as the *interior spiders*. The complements of the inputs and the outputs are defined as $\overline{I} = V \setminus I$ and $\overline{O} = V \setminus O$ respectively. We can write any quantum circuit as ZX-diagram by replacing gates with equivalent diagrams and use rules from ZX-calculus to modify them without changing the linear map. For instance, the following rules hold:



The fusion rule (f) allows to merge spiders of the same color together if they are connected by at least one normal wire and (h) allows to change the colors of spiders by flipping normal and Hadamard wires. All rules hold in both directions and are also valid with interchanged colors, so we can also split up spiders with (f). There exists a complete graphical rule set for transforming ZX-diagrams [53].

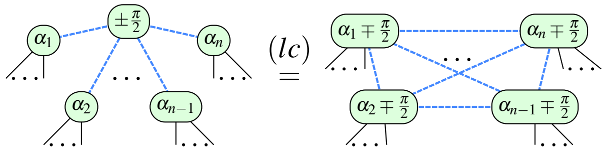
3.1.1 Graph-like diagrams

In this work, we consider the class of *graph-like* ZX-diagrams as introduced in [15], which allow us to represent any quantum computation as a graph of parametrized green Z-spiders and Hadamard wires. In those diagrams we represent Hadamard wires between spiders as dashed blue line instead of the yellow box for easier visualization. One can transform any ZX-diagram into an equivalent graph-like ZX-diagram by repeatedly applying standard ZX-rules [15] (c.f. Figure 1). This formalism provides a link between quantum computing and graph theory since the entire computation is captured by the *underlying graph* spanned by Hadamard wires, combined with phases of Z-spiders. Moreover, graph-like diagrams can be directly interpreted as measurement patterns in the model of measurement-based quantum computing (MBQC) [4, 10].

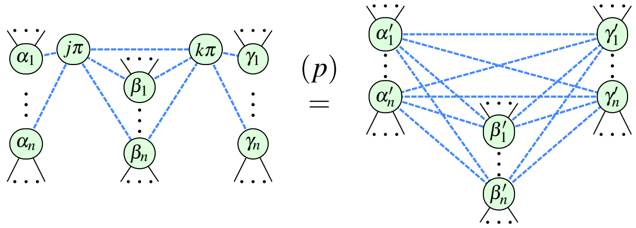
3.1.2 Diagram simplification

We can rewrite graph-like diagrams into equivalent simplified versions (i.e., decreasing the number of spiders or wires), by using graph-theoretic rewrite rules as shown in [15]:

Local complementation Given an undirected graph G, local complementation on a vertex v (written $G \star v$) consists of flipping the edges between the neighbors of v. That is, after local complementation, every pair of neighbors of v is connected iff it was not connected before. In graph-like ZX-diagrams, we can use a rewrite rule based on local complementation (lc) to eliminate spiders with a phase of $\pm\frac{\pi}{2}$:


α_1
$\pm\frac{\pi}{2}$
α_n
α_2
α_{n-1}
(lc)
$=$
$\alpha_1 \mp \frac{\pi}{2}$
$\alpha_n \mp \frac{\pi}{2}$
$\alpha_2 \mp \frac{\pi}{2}$
$\alpha_{n-1} \mp \frac{\pi}{2}$

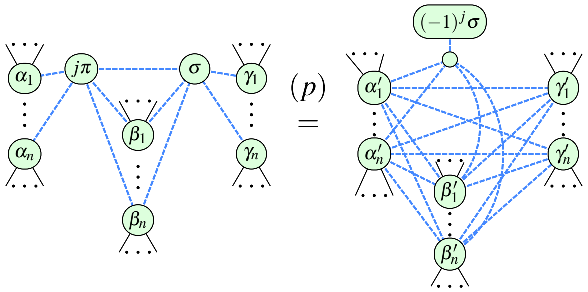
Pivoting A Pivot $G \wedge uv$ consists of three local complementations $(G \star u \star v \star u)$ applied on a pair of neighboring vertices u, v. We can use a similar rewrite rule (p) in graph-like ZX-diagrams to eliminate pairs of spiders with phase 0 or π:



$$\alpha'_i = \alpha_i + k\pi$$
$$\beta'_i = \beta_i + (j+k+1)\pi$$
$$\gamma'_i = \gamma_i + j\pi$$

By repeatedly applying those rules one can eliminate all interior spiders with phase $\pm\frac{\pi}{2}$ and every pair of interior spiders with phase 0 or π [15].

Phase gadgets One can further simplify ZX-diagrams with a slightly modified version of the pivot rule if we allow one spider of a pair to have a non-Clifford phase σ [23]. The non-Clifford spider does not get removed but is transformed into a so called *phase gadget*:



$$\alpha'_i = \alpha_i + k\pi$$
$$\beta'_i = \beta_i + (j+k+1)\pi$$
$$\gamma'_i = \gamma_i + j\pi$$

In graph-like ZX-diagrams a phase gadget consists of a "top" spider exclusively connected to a phaseless "root" spider connected to other spiders. Simplifying graph-like diagrams with all three rules, we obtain a diagram where spiders either have a non-Clifford phase, are part of a phase gadget or a boundary.

3.1.3 Gflow in graph-like diagrams

Gflow is a graph-theoretic property for measurement patterns defined on labeled open graphs (G, I, O, λ), where $G = (V, E)$ is an undirected graph with vertices V and edges E, $I \subseteq V$, $O \subseteq V$ are the set of inputs resp. outputs, and λ is a labeling function assigning each vertex a measurement plane of the Bloch sphere in $\{XY, XZ, YZ\}$ [11]. A labeled open graph has gflow if there exists a map $g : \overline{O} \to \mathscr{P}(\overline{I})$ and a partial order $\prec$ over V, s.t. for all $v \in \overline{O}$:

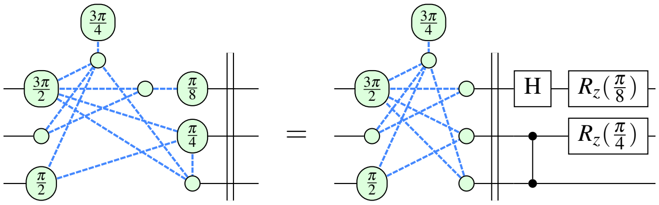
- If $w \in g(v)$ and $v \neq w$, then $v \prec w$.
- If $w \in Odd(g(v))$ and $v \neq w$, then $v \prec w$.
- If $\lambda(v) = XY$, then $v \notin g(v)$ and $v \in Odd(g(v))$.
- If $\lambda(v) = XZ$, then $v \in g(v)$ and $v \in Odd(g(v))$.

- If $\lambda(v) = YZ$, then $v \in g(v)$ and $v \notin Odd(g(v))$.

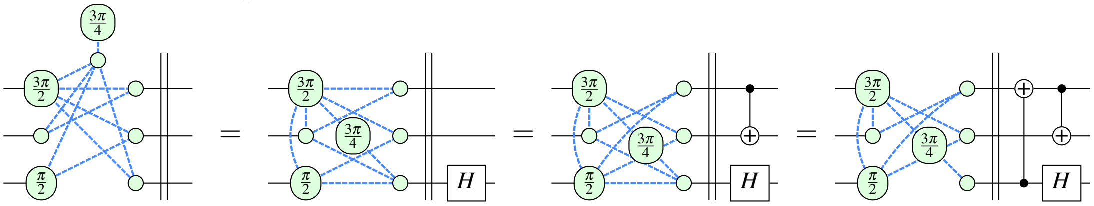
In graph-like diagrams, we interpret the underlying graph as a labeled open graph with phase gadgets corresponding to YZ measurements[1] and other spiders corresponding to XY measurements [4]. Since the initial graph-like diagrams obtained from quantum circuits (as in Figure 1) have gflow [15] and all above rules preserve gflow [4], the simplified diagrams have gflow as well.

3.1.4 Circuit extraction

Extracting quantum circuits back from graph-like ZX-diagrams where the circuit has only as many qubits as there are outputs/inputs, is so far only possible in polynomial time if the underlying graph has some kind of flow [4, 48]. Here, we give a brief overview of the extraction algorithm for graph-like ZX-diagrams with gflow as described in [4]. The algorithm extracts a quantum circuit from a ZX-diagram by taking suitable parts of the diagram and creating their equivalent representation as a quantum gate within the circuit at the corresponding position. These parts are then removed from the diagram, extracting one gate at a time, until only the inputs and outputs of the diagram remain. During the process, a set of green Z-spiders called the *frontier* separates the extracted part of the diagram from the unextracted part. Phases of frontier spiders can be directly extracted as R_z gates, and Hadamard wires between frontier spiders as CZ gates. Furthermore, Hadamard wires where a frontier spider w is exclusively connected to a non-frontier spider v can be extracted as Hadamard gates with v replacing w in the frontier:



If every spider in the frontier has at least two non-frontier neighbors we can add wires of a frontier spider to the wires of another one by placing a CX gate on the extracted circuit. If all neighbors are measured in the XY plane, gflow ensures that there exists a combination of additions so that there remains a frontier spider with only a single neighbor. We can obtain such a combination by applying Gaussian elimination on the biadjacency matrix between the frontier vertices and their neighbors. Otherwise, if there are YZ-measured neighbors, we can transform them into XY measurements by applying a pivot on the neighbor and a connected frontier spider:



By repeating these procedures, we can transform the entire diagram into a quantum circuit.

4 Extracting controlled phase gates from graph-like ZX-diagrams

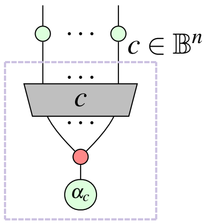
The process of transforming quantum circuits to graph-like ZX-diagrams, simplifying them, and re-extracting circuits can already be seen as an implicit synthesis algorithm to the gate set $\{R_z, H, CZ, CX\}$.

[1] The root spider is labeled as YZ measurement, while the top spider corresponds to a measurement effect which is omitted from the underlying graph.

Given the requirements of neutral atom platforms, it may be desirable to incorporate two- and multi-controlled phase gates of arbitrary rotations into this gate set. We first show how such gates are represented in graph-like ZX-diagrams, then how we incorporate this finding into the extraction algorithm.

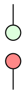
4.1 Graph-like representation of controlled phase gates

The (multi-) controlled phase gate is a diagonal gate, meaning all non-zero entries of its corresponding matrix in the Z-basis are on its diagonal. Such gates can be represented as a semi-Boolean function $f:\{0,1\}^n \to \mathbb{C}$ which assigns a complex number to each basis state. Ref. [26] shows, that any semi-Boolean function $f(b) = a_b$ with $a_b \in \mathbb{C}$ and $b \in \mathbb{B}^n$ can be expressed in ZX-calculus as follows:



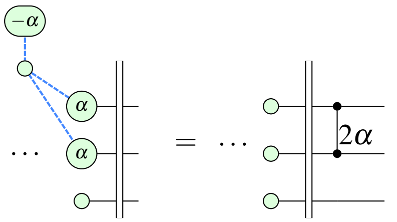
with

$c_i = \;|\;$ if $c_i = 1$, and

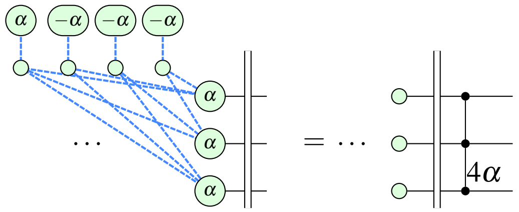
$c_i =$  if $c_i = 0$

$$\alpha_c = \tfrac{-1}{2^{n-1}} \textstyle\sum_{b\in\mathbb{B}^n} f(b)\chi(b,c) \tag{1}$$

The part in the dashed box is repeated for every Boolean vector c in $\mathbb{B}^n$ and the grey box decomposes into n subdiagrams either connecting the corresponding lower and upper wire with a normal wire if the i-th element of the Boolean vector is 1, or disconnecting them if it is 0. Further, the phase α_c can be obtained by the formula on the right, where $\chi(b,c) = (-1)^{b\cdot c}$ corresponds to a parity function with $\cdot$ being the inner product: If b and c overlap in an odd number of elements it returns -1, else 1. This rule yields a combination of phase gadgets, and when applying the color change rule on the middle red spider, we obtain the same graph structures as introduced in the previous section. To model an n-controlled phase gate $\mathrm{C}_n\mathrm{P}(\varphi)$ as a semi-Boolean function, we take α as an all-zero vector of length 2^n except for its last entry being φ. Following Equation (1), one can transform the function to a ZX-diagram which has $2^n - 1$ phase gadgets split up into $\binom{k}{n}$ phase gadgets for $k \in \{1,\dots n\}$. For instance, the two and three-qubit-controlled phase gates have the following representation:



(2)



(3)

For arbitrary-sized multi-controlled phase gates, this generalizes to the following theorem:

Theorem 1 (Multi-controlled phase gates)**.** *Let $\binom{S}{k}$ denote the set of all k-combinations of a set S and $PG(\alpha, N)$ denote a phase gadget with phase α connected to neighbors N which are empty Z-spiders. A n-qubit controlled phase gate $C_nP(\varphi)$ is equivalent to a graph-like ZX-diagram with outputs $O, |O| = n$ having a phase of α and phase gadgets*[2]

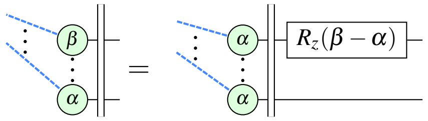
$$\prod_{k=2}^{n} \prod_{s\in\binom{O}{k}} PG((-1)^{k+1}\alpha, s), \qquad \alpha = \frac{\varphi}{2^{n-1}}$$

Proof. A graphical proof is given by the ZX representation of the diagonal gate and the corresponding proof in [26], a combinatorial proof can be found in [2]. We give an alternative combinatorial proof in Appendix B. □

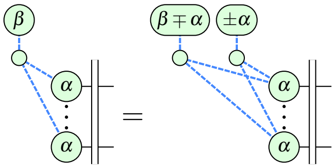
[2] Note, that the product notation here corresponds to the composition $\circ$ of ZX-diagrams.

4.2 Adaption of the extraction algorithm

We adapt the algorithm described in 3.1.4 by including an additional C$_n$P gate extraction step between CZ and R$_z$ extraction. For that, we carry out a pattern match on the phase gadgets which are exclusively connected to the outputs, i.e., the frontier. If we find a graph structure as described in Theorem 1 for n frontier spiders, we take the phase of the gadget which is connected to all n spiders as the desired phase α if n is odd, or $-\alpha$ if n is even, and adjust the phases of all other gadgets using the following two rewrites:

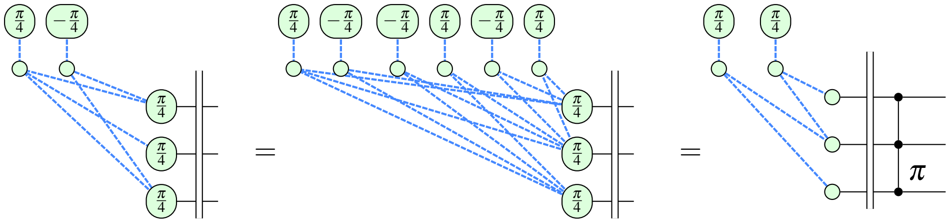


(4)



(5)

With Equation (4), we extract the unwanted part of an output phase as R$_z$ gate, and with Equation (5), we split up phase gadgets into a part with the desired phase and another gadget so that the sum of the phases yields the original one. Both rewrites are sound in ZX-calculus: The first corresponds to an application of the fusion rule as mentioned in Section 3 and the second is a reversed version of the gadget fusion rule as shown in [23, Section D]. With adjusted phases, we extract the entire graph structure by removing it from the diagram and placing a C$_n$P(φ) gate with $\varphi = \alpha \cdot 2^{n-1}$ on the circuit.
We can extend this procedure by also allowing the extraction of graph structures where some phase gadgets required for a C$_n$P extraction are missing in the diagram. Consider the following example, where we are initially missing two 2-ary phase gadgets with $-\frac{\pi}{4}$ to complete a C$_2$P(π) structure:



(6)

By adding pairs of phase gadgets with opposite phases corresponding to the identity, we complete the required graph structure to extract the gate. Some inserted gadgets then remain in the diagram and are extracted later. If we always take the gadget with the most neighbors, complete the diagram to match a C$_n$P gate, and extract it, every phase gadget will get extracted as part of a C$_n$P gate at some point and we can entirely omit YZ spider eliminations via pivoting.

4.2.1 Preservation of gflow

For a complete translation of graph-like diagrams into quantum circuits, it is essential that all operations preserve gflow on the diagram. The rewrites of Equations 3-7 essentially reduce to deletions and insertions of phase gadgets, i.e., YZ measurements, connected to only outputs. Since the original extraction algorithm preserves gflow and it has been shown in [4, Lemma 3.4.] that the deletion of arbitrary YZ measurements preserves gflow, the same remains to be shown for the insertion case:

Lemma 1 (Insertion of YZ measurements on outputs). *Let $(g,\prec)$ be a gflow for (G,I,O,λ) and let $W \subseteq O$. Then (G',I,O,λ'), where $G' = (V',E')$ with $V' = V \cup \{x\}$, $\lambda'(x) = YZ$ and $E' = E \cup \{(x,w)|w \in W\}$ has a gflow.*

Proof. We provide the detailed proof in Appendix A. □

4.2.2 Time complexity

The time complexity of the proposed approach in terms of elementary graph operations depends on whether we allow additional insertions of phase gadgets or not. Let k denote the number of spiders in a diagram and n the number of outputs:

- If we do not allow additional insertions of phase gadgets, we have approximately the same runtime as the original algorithm, namely $O(n^2k^2 + k^3)$ which is summed from the runtime for Gaussian elimination $O(n^2k)$, pivoting YZ measurements $O(k^2)$ and k steps in total [4]. Additionally, for our approach, we have to split at most k phase gadgets at each step, which adds another $O(nk)$ term to the elementary graph operations. Yet, this term gets absorbed by the complexity of the Gaussian elimination.
- If we allow phase gadget insertions to complete the graph structures corresponding to a C_nP gate, the complexity essentially becomes $O(2^{n+1}k)$. This is because, in the worst case, we would complete structures where there is only a single phase gadget connected to all n outputs, and we need to add $2 \cdot 2^n - n - 2$ additional gadgets. We want to emphasize that this worst-case complexity is unlikely to occur in practice. Yet, for larger circuits, it may be useful to limit the size of extractable C_nP gates to a constant.

5 Neutral Atom Circuit Synthesis

In the following, we want to apply the proposed extraction scheme to circuit synthesis for neutral atom (NA)-based hardware due to their native support of C_nP gates. Therefore, we briefly introduce the hardware capabilities [44], embed our proposed scheme into a complete synthesis procedure, and evaluate the effect of the C_nP extraction regarding the circuit execution time in comparison to Qiskits internal synthesis algorithm.

5.1 Neutral Atom Background

For NA-based quantum computers, qubit registers are realized by placing single atoms in optical dipole traps created by laser beams, referred to as optical lattices or optical tweezers. While arbitrary atom arrangements are possible, we assume a rectangular grid as illustrated in Figure 2. The qubit states can be encoded in long-lived internal atomic states such as hyperfine or nuclear spin states. Commonly employed atomic species include alkali or alkaline-earth(-like) atoms such as Rb and Sr, which provide suitable internal states with long coherence times. Gates are realized with specific laser pulses on the atoms using uniform global beams, with the possibility of addressing a whole register or individual qubits [17,27]. Multi-qubit gates are based on the long-range interaction between close-by atoms excited to high-lying Rydberg states [16, 21, 28, 32, 34, 41, 42]. There exist different protocols to realize both single- and multi-qubit gates, and the preferred implementation depends on many parameters such as the chosen atom species, the respective qubit encoding, and the experimental setup. In this work, we focus on individually addressable C_nP(φ) gates between neighboring atoms as a generalization of the often implemented CZ gate, which can be realized by tuning the accumulated phase during the Rydberg interaction to an arbitrary angle φ instead of π [28]. This might even result in improved gate times and fidelity due to the shorter time of the atom spent in the Rydberg state [16]. Regarding single-qubit gates, we assume a scheme between fast, individually addressable AC Stark shift beams, realizing local R_z rotations and slow, globally addressing microwave pulses performing a rotation about an axis in the

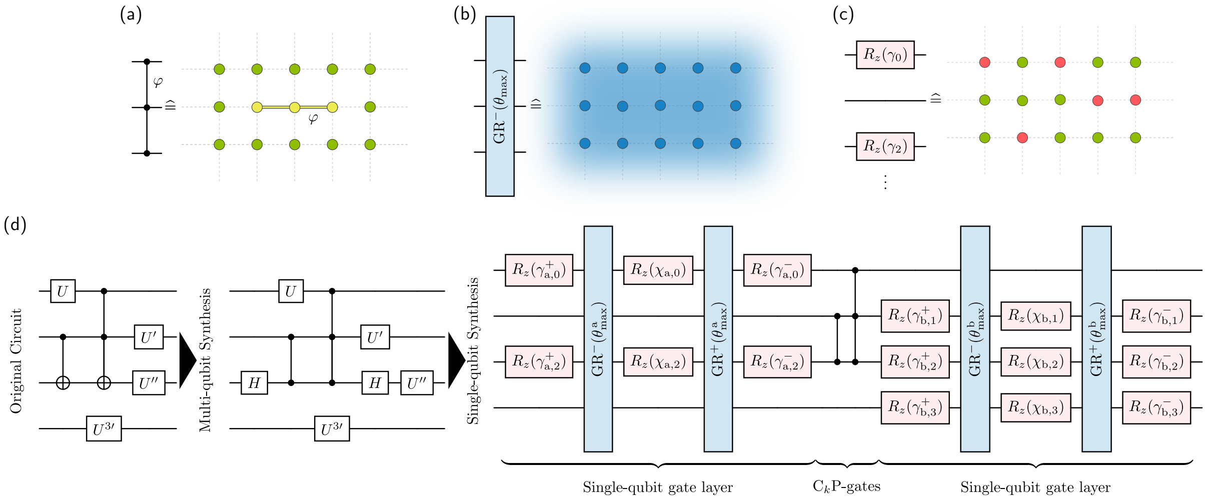


Figure 2: Illustration of the neutral atom gate capabilities and the process of synthesizing and scheduling quantum circuits to the hardware. **(a)** Native (multi-) controlled phase gates ($\mathrm{C}_n\mathrm{P}(\varphi)$), here shown for three qubits. **(b)** Global single-qubit rotations in the XY-plane. **(c)** Individually addressable Z-rotations ($\mathrm{R}_z(\gamma)$). **(d)** Synthesis to alternating single- and multi-qubit layers: First, the synthesis of multi-qubit gates to $\mathrm{C}_n\mathrm{P}$ gates. Second, the synthesis and scheduling of single-qubit gates into global XY-rotations and individually addressable Z-rotations according to the transversal decomposition of [36].

XY-plane for all qubits simultaneously. In particular, we assume the following gate definitions, where single-qubit gates are equivalent to the ones from [36] with GR as global XY-rotations applied to all n qubits, and $\hat{Y}$ being the Pauli-Y matrix:

$$\mathrm{C}_n\mathrm{P}(\varphi) \equiv \mathrm{diag}(1,\ldots,1,e^{i\varphi}), \quad \mathrm{R}_z(\gamma^{\pm}) \equiv \mathrm{diag}(e^{-i\gamma^{\pm}/2}, e^{-i\gamma^{\pm}/2}), \quad \mathrm{GR}(\theta_{\mathrm{max}}) \equiv \exp\left(-i\frac{\theta}{2}\sum_{i=1}^{n}\hat{Y}_i\right) \tag{7}$$

In this setting, sets of arbitrary but simultaneous single-qubit gates on different qubits can be converted into two global illuminations interleaved with single-qubit Z-rotations. On unused qubits, the two complementary global rotations cancel out, effectively applying an identity operation. To convert single-qubit gates to this setting, we consider the transversal decomposition scheme introduced in [36] which is optimal in terms of global pulse time. An illustration of the gate capabilities and how to synthesize the respective single- and multi-qubit gates is shown in Figure 2.

5.2 Related Work

Recently, there has been a fast development of NA-specific compilation methods [5, 9, 30, 36, 37, 45, 51, 52, 54, 55] focusing almost exclusively on the mapping and scheduling tasks within the compilation. The only exceptions are [36], proposing the single-qubit synthesis also used in this work, and the Geyser framework [37] using numerical optimization to introduce additional CCZ gates. Nevertheless, both neglect the capability of NAs to natively execute controlled gates with arbitrary phases and, furthermore, the ability to directly execute multi-controlled gates for more than one control qubit. Although convenient, this neglects potentially shorter or simpler circuits using these specific capabilities of NAs. Therefore, in the following, we evaluate five different synthesis schemes to compare the commonly used naive synthesis algorithms with an NA-specific synthesis employing the ZX-approach of Section 4.2.

5.3 Evaluation Setup

The total gate synthesis process consists of the two steps illustrated in Figure 2: First, the synthesis of the multi-qubit gates into $\mathrm{C}_n\mathrm{P}$ gates and arbitrary single-qubit rotations, and second, the synthesis and scheduling of single-qubit gates into the alternating global vs. local scheme, resulting in the fully synthesized circuit containing only gates from the native gate set of Equation (7). For the first step, we consider the following five schemes:

1. **Qiskit-default:** Circuits are converted into the Qiskit-supported native gate set of $\{\mathrm{U}_3, CZ\}$ using the internal `transpile` function and setting the optimization level to three. This approach uses Qiskit internal schemes to decompose gates with more than two-qubit gates.
2. **No-decomp:** The Qiskit decomposition introduces a large overhead that can be bypassed for NAs. For better comparison, we propose this scheme which synthesizes to $\{\mathrm{U}_3, \mathrm{C}_n\mathrm{Z}\}$ by replacing all (multi-) controlled gates by their $\mathrm{C}_n\mathrm{Z}\}$ equivalent and using the Qiskit-default approach for the single qubit gates.
3. **ZX-default:** The circuit is converted into a graph-like ZX diagram, and the default extraction algorithm of PyZX [24] is used to recreate a circuit.
4. **ZX-no-insert:** Similar to the default but the extraction scheme from Section 4.2 is used to synthesize $\mathrm{C}_n\mathrm{P}$ gates.
5. **ZX-with-insert:** In addition to ZX-no-insert, we allow the insertion of additional phase gadgets, resulting in possibly more and larger phase gate extractions.

Since the ZX extraction sometimes produces redundant gates, we additionally apply a basic gate cancellation algorithm afterwards. In the second step, the transverse decomposition according to [36] is used to synthesize the single qubit gates. In this scheme, $\theta_{\max} \equiv \max_i \theta_i$ is defined as the maximum of the first Euler angle θ_i of any single qubit rotation in this layer. According to the discussion in [36], the total gate execution time scales linearly in this angle with the maximal duration at $\theta_{\max} = \pi$.

As for many single-qubit gates the actual moment of execution is not unique, it can be added to different layers. We thus use an additional greedy optimization step, not performed in [36], to check the possible positions of the single-qubit gates and assign them such as to minimize the overall $\theta_{\max}$. In particular, a gate with Euler angle θ is preferably assigned to a layer with $\theta_{\max} > \theta$, allowing the gate to be executed without increasing the gate time. As evaluation metrics, we compute both simple gate counts and the total circuit execution time T by scheduling the gates according to the illustration in Figure 2. We sum individual gate times, where we assume the gate execution time increases linearly with the rotation angle as follows:

$$T = \sum_{i=0}^{d} \frac{|max_\gamma(R_z(\gamma), i)|}{\pi} 100ns + \sum_{GR(\theta_{\max})} \frac{|\theta_{\max}|}{\pi} 100\mu s + \sum_{C_1P(\varphi)} \frac{|\varphi|}{\pi} 100ns + \sum_{C_nP(\varphi), n>1} \frac{|\varphi|}{\pi} 400ns \quad (8)$$

Here d denotes circuit depth, meaning we assume full parallel execution of the R_z if possible by taking the maximum angle of each layer. Multi-qubit gates are assumed to be executed in a sequential way. The gate times are 0.1 µs for the R_z [46] and the $\mathrm{CP}(\varphi = \pi)$ gate [31]. For all higher-weight controlled phase gates $\mathrm{C}_{\geq 2}\mathrm{P}$ we assume 0.4 µs for $\varphi = \pi$. The dominating factor for circuit execution time are the slow global illuminations GR with 100 µs [46]. Therefore, our main aim to use the scheme of Section 4.2 is to lower the number of global GR gates and, in this way, reduce to overall execution time.

For a comprehensive and rigorous evaluation, we chose circuits from three different benchmark collections, with their descriptions available online: **QASM-Bench(small)** [29] and **MQT-Bench** [40] contain

Table 1: Averaged reduction of execution time T relative to Qiskit-default. Negative percentages indicate an increased execution time.

	Circuits	Qiskit	No-decomp	ZX-default	ZX-no-insert	ZX-with-insert
QASM-Bench [29] [1]	35	0%	8%	2%	14%	26%
MQT-Bench [40] [2]	11	0%	0%	−44%	−16%	26%
Feynman-Bench [1] [3]	26	0%	63%	−23%	−16%	40%

[1] https://github.com/pnnl/QASMBench [2] https://www.cda.cit.tum.de/mqtbench/ [3] https://github.com/meamy/feynman

various low-level benchmark circuits of different sizes and types with common quantum subroutines and algorithms. Additionally, we also consider the **Feynman-Bench** [1] collection. Created for formal methods, it contains different arithmetic circuits usually based on Toffoli gates.

The code used for the evaluations is available with an MIT license at Zenodo [50] allowing reproducibility and possible usage or integration into other compilation projects.

5.4 Results & Discussion

The five compilation schemes are evaluated on gate count and execution time $T[\text{ms}]$ of the synthesized circuits, together with the algorithm runtime $r[\text{s}]$. We first discuss time reduction averaged over all circuits, summarized in Table 1, then, we highlight six examples shown in Table 2, which have been selected to best illustrate different cases within the dataset. The full dataset with all raw data is available at Zenodo [50]. On the QASM-Bench collection (1), the ZX-with-insert approach results in an average 26% reduction of execution time compared to the Qiskit internal synthesis improving 23 of the 35 circuits. Similar numbers result for the MQT-Bench (2) circuits with 26% reduction of execution time, improving 7 out of 11 circuits. For the Feynman benchmarks (3), results are mixed: While our scheme achieves a 40% reduction compared to Qiskit, improving 24 of 26 circuits, the No-decomp scheme has an even higher average reduction of execution time with 63%.

Considering the above part of Table 2 one can see how the synthesis approach described in this work is capable of successfully synthesizing C_nP gates. Since No-decomp just replaces multi-controlled gates by their $\text{C}_n\text{Z}\}$ equivalent, the corresponding column indicates the number of multi-controlled gates present in the original circuit. In comparison, one can then see that while No-insert only resynthesizes a few of the original gates, With-insert synthesizes more multi-qubit gates and is often able to create even higher-dimensioned controlled gates. This higher-controlled gate synthesis appears very dominantly in dense circuits such as qnn_10, corresponding to a quantum neural network circuit, but also in circuits that are natively built on controlled phase gates such as HHL.

Due to the controlled gates, the proposed approach is capable of effectively reducing the number of slow global GR gates in comparison to the regular ZX extraction scheme and Qiskit, which are not capable of synthesizing multi-controlled gates. Generally, a lower number of GR gates also results in a shorter circuit execution time with some exceptions, such as the hhl_n7, where the ZX synthesis has a longer execution time than the No-decomp scheme, although the number of absolute GR pulses is lower. This is likely due to an increased pulse time of the individual GR gates.

The ZX approaches do not perform well on circuits that already contain close to optimal multi-controlled gates, for instance on circuits of the Feynman benchmark. Here, the approaches extract gates in a less efficient way, resulting in a gate and time overhead. In such cases, replacing multi-controlled gates without changing circuit structure as in the No-decomp scheme is the best option. This can also be seen when comparing the number of GR gates for the two adder circuits to the No-decomp scheme, where

Table 2: Evaluation results for six benchmarks, selected to illustrate both good and poor performance. Numbers after the names indicate the corresponding benchmark collection. The first table shows gate counts corresponding to the native gate set of Equation (7). The second table contains the total execution time T[ms] and the synthesis algorithm runtime r[s] on a consumer notebook.

	ZX											Qiskit		Own alternative		
	Default		No-insert			With-insert						Default		No-decomp		
	GR	CP	GR	CP	C_2P	GR	CP	C_2P	C_3P	C_4P	C_5P	GR	CP	GR	CP	C_2P
hhl_n7 (1)	448	296	362	241	-	**306**	207	42	29	6	-	356	196	356	196	-
qft_10 (2)	90	140	36	75	-	**14**	62	14	-	-	-	44	105	44	105	-
qnn_10 (2)	106	334	62	199	-	**28**	159	48	26	8	1	76	188	76	188	-
gf2^7_mult (3)	214	956	134	447	19	**14**	22	114	-	-	-	194	300	18	6	49
rc_adder_6 (3)	76	100	84	95	-	62	91	4	-	-	-	86	93	**28**	27	11
qcla_adder_10 (3)	128	331	138	462	1	24	153	121	-	-	-	74	233	**18**	29	34

	ZX						Qiskit		Own alternative	
	Default		No-insert		With-insert		Default		No-decomp	
	T	r	T	r	T	r	T	r	T	r
hhl_n7 (1)	11.07	0.085	8.71	0.156	7.70	0.239	5.45	0.122	**5.45**	0.277
qft_10 (2)	2.16	0.036	0.91	0.029	**0.35**	0.039	0.89	0.043	0.89	0.103
qnn_10 (2)	3.12	0.031	1.53	0.108	**0.74**	0.214	3.06	0.117	3.06	0.151
gf2^7_mult (3)	5.84	0.218	3.31	1.286	**0.40**	3.421	4.20	0.083	0.47	0.041
rc_adder_6 (3)	1.89	0.030	2.04	0.049	1.56	0.052	1.67	0.033	**0.71**	0.041
qcla_adder_10 (3)	3.26	1.326	3.48	0.448	0.66	2.063	1.63	0.073	**0.47**	0.078

the ZX approaches are not capable of reconstructing a similar efficient circuit structure. Since all ZX strategies yield inefficient circuits, it may be that in such cases the ZX-diagram simplification creates too complex graph structures.

Regarding algorithmic runtime, ZX performs similarly to the Qiskit internal synthesis. Note, however, the significant increase in runtime for qcla_adder and gf2^t_mult for the With-insert synthesis. This is likely the overhead due to the insertion of additional phase gadgets, resulting in worst-case exponential runtime as discussed in Section 4.2.2.

6 Conclusion

In this work we introduced a novel approach to synthesize quantum circuits to the universal gate set $\{H, R_z, C_nP\}$. As a key contribution, our approach is able to efficiently identify structures in graph-like ZX-diagrams that correspond to multi-controlled phase gates and extract them to quantum circuits. This allows the synthesis of such gates even if they were not present in the original circuit. Together with existing simplification strategies for ZX-diagrams, our approach can be used to synthesize arbitrary quantum circuits towards neutral atom architectures. Here, our synthesis often trades slow global pulse rotations for fast multi-controlled qubit gates and we are thus able to reduce execution time significantly for many common circuits. Further, this could also help hardware developers to evaluate whether increasing the number of qubits supported by multi-controlled phase gates is beneficial for certain problems in terms of execution time and fidelity. In cases where the circuit already consists of optimized multi-controlled gates, such as circuits based on arithmetic functions, the synthesis may result in less efficient quantum circuits. This is likely due to overly complex graph structures resulting from ZX-diagram simplification. We leave it as a topic for further research whether in those cases more sophisticated strategies allow exploiting the phase gadget structures for multi-controlled phase gates synthesis without increasing the underlying graph structure complexity. Possible approaches include advanced heuristics applying the proposed scheme only to cases where it is likely to improve the circuit structure. We also want to mention that a similar synthesis approach could be done without ZX-calculus using the Pauli Dependency DAG representation of quantum circuits [48]. It has been shown that the diagram simplification rules from Section 3.1.2 are equivalent to reordering Pauli terms in a Pauli Dependency DAG and by identifying patterns of individual Pauli-Z terms similar to Theorem 1 we can then synthesize C_nP gates. As future work, it would be interesting to see how these two versions compare.

Acknowledgments

This work is partially supported by the German Federal Ministry of Education and Research (BMBF) under the funding program Quantum Technologies - From Basic Research to Market under contract number 13N16070.

The authors acknowledge funding from the Munich Quantum Valley initiative (K3, K5), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

J.Z. acknowledges funding by the Max Planck Society (MPG), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC-2111-39081486, and acknowledges support from the German Federal Ministry of Education and Research (BMBF) through the program "Quantum technologies - from basic research to market" (SNAQC, Grant No. 13N16265).

L.S. and R.W. acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101001318).

The authors would like to thank Nathalia Nottingham for providing an implementation of the transversal layer decomposition and Miriam Backens for helpful discussions.

References

[1] Matthew Amy (2019): *Towards Large-scale Functional Verification of Universal Quantum Circuits*. *Electronic Proceedings in Theoretical Computer Science* 287, pp. 1–21, doi:10.4204/EPTCS.287.1.

[2] Matthew Amy, Parsiad Azimzadeh & Michele Mosca (2018): *On the controlled-NOT complexity of controlled-NOT–phase circuits*. *Quantum Science and Technology* 4(1), p. 015002, doi:10.1088/2058-9565/aad8ca. Available at `https://dx.doi.org/10.1088/2058-9565/aad8ca`.

[3] Matthew Amy, Andrew N. Glaudell, Sarah Meng Li & Neil J. Ross (2023): *Improved Synthesis of Toffoli-Hadamard Circuits*, pp. 169–209. doi:10.1007/978-3-031-38100-3_12. arXiv:2305.11305.

[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. Available at `http://arxiv.org/abs/2003.01664`. ArXiv:2003.01664 [quant-ph].

[5] Jonathan M. Baker, Andrew Litteken, Casey Duckering, Henry Hoffmann, Hannes Bernien & Frederic T. Chong (2021): *Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quantum Architectures*, pp. 818–831. doi:10.1109/ISCA52012.2021.00069.

[6] Daniel Barredo, Sylvain de Léséleuc, Vincent Lienhard, Thierry Lahaye & Antoine Browaeys (2016): *An Atom-by-Atom Assembler of Defect-Free Arbitrary Two-Dimensional Atomic Arrays*. Science 354(6315), pp. 1021–1023, doi:10.1126/science.aah3778.

[7] 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, pp. 1–3, doi:10.1038/s41586-023-06927-3.

[8] Dolev Bluvstein, Harry Levine, Giulia Semeghini, Tout T. Wang, Sepehr Ebadi, Marcin Kalinowski, Alexander Keesling, Nishad Maskara, Hannes Pichler, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2022): *A Quantum Processor Based on Coherent Transport of Entangled Atom Arrays*. Nature 604(7906), pp. 451–456, doi:10.1038/s41586-022-04592-6.

[9] Sebastian Brandhofer, Ilia Polian & Hans Peter Büchler (2021): *Optimal Mapping for Near-Term Quantum Architectures Based on Rydberg Atoms*. In: 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pp. 1–7, doi:10.1109/ICCAD51958.2021.9643490.

[10] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf & Maarten Van den Nest (2009): *Measurement-based quantum computation*. Nature Physics 5(1), pp. 19–26, doi:10.1038/nphys1157.

[11] 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. Available at `https://dx.doi.org/10.1088/1367-2630/9/8/250`.

[12] Alec Cao, William J. Eckner, Theodor Lukin Yelin, Aaron W. Young, Sven Jandura, Lingfeng Yan, Kyungtae Kim, Guido Pupillo, Jun Ye, Nelson Darkwah Oppong & Adam M. Kaufman (2024): *Multi-qubit gates and 'Schrödinger cat' states in an optical clock*. arXiv:2402.16289.

[13] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes*. Cambridge University Press, doi:10.1017/9781316219317.

[14] Clemens Dlaska, Kilian Ender, Glen Bigan Mbeng, Andreas Kruckenhauser, Wolfgang Lechner & Rick van Bijnen (2022): *Quantum Optimization via Four-Body Rydberg Gates*. Physical Review Letters 128(12), p. 120503, doi:10.1103/PhysRevLett.128.120503.

[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. Available at `https://quantum-journal.org/papers/q-2020-06-04-279/`. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[16] Simon J. Evered, Dolev Bluvstein, Marcin Kalinowski, Sepehr Ebadi, Tom Manovitz, Hengyun Zhou, Sophie H. Li, Alexandra A. Geim, Tout T. Wang, Nishad Maskara, Harry Levine, Giulia Semeghini, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2023): *High-Fidelity Parallel Entangling Gates on a Neutral-Atom Quantum Computer*. Nature 622(7982), pp. 268–272, doi:10.1038/s41586-023-06481-y. arXiv:2304.05420.

[17] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl, O. Crawford, N. S. Blunt, B. Rogers, T. Noel & M. Saffman (2022): *Multi-Qubit Entanglement and Algorithms on a Neutral-Atom Quantum Computer*. Nature 604(7906), pp. 457–462, doi:10.1038/s41586-022-04603-6.

[18] Daniel Große, Robert Wille, Gerhard W Dueck & Rolf Drechsler (2009): *Exact multiple-control Toffoli network synthesis with SAT techniques*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28(5), pp. 703–715, doi:10.1109/TCAD.2009.2017215.

[19] Flavien Gyger, Maximilian Ammenwerth, Renhao Tao, Hendrik Timme, Stepan Snigirev, Immanuel Bloch & Johannes Zeiher (2024): *Continuous Operation of Large-Scale Atom Arrays in Optical Lattices*. arXiv:2402.04994.

[20] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond & Christophe Jurczak (2020): *Quantum Computing with Neutral Atoms*. Quantum 4, p. 327, doi:10.22331/q-2020-09-21-327.

[21] L. Isenhower, M. Saffman & K. Mølmer (2011): *Multibit CkNOT Quantum Gates via Rydberg Blockade*. Quantum Information Processing 10(6), p. 755, doi:10.1007/s11128-011-0292-4.

[22] Sven Jandura & Guido Pupillo (2022): *Time-Optimal Two- and Three-Qubit Gates for Rydberg Atoms*. Quantum 6, p. 712, doi:10.22331/q-2022-05-13-712.

[23] Aleks Kissinger & John van de Wetering (2020): *Reducing the Number of Non-Clifford Gates in Quantum Circuits*. Physical Review A 102(2), p. 022406, doi:10.1103/PhysRevA.102.022406.

[24] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning* 318, pp. 229–241. doi:10.4204/EPTCS.318.14.

[25] 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:10.5555/2535649.2535653.

[26] Stach Kuijpers, John van de Wetering & Aleks Kissinger: *Graphical fourier theory and the cost of quantum addition*. Available at `https://doi.org/10.48550/arXiv.1904.07551`.

[27] Harry Levine, Dolev Bluvstein, Alexander Keesling, Tout T. Wang, Sepehr Ebadi, Giulia Semeghini, Ahmed Omran, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2022): *Dispersive Optical Systems for Scalable Raman Driving of Hyperfine Qubits*. Physical Review A 105(3), p. 032618, doi:10.1103/PhysRevA.105.032618.

[28] Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Tout T. Wang, Sepehr Ebadi, Hannes Bernien, Markus Greiner, Vladan Vuletić, Hannes Pichler & Mikhail D. Lukin (2019): *Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms*. Physical Review Letters 123(17), p. 170503, doi:10.1103/PhysRevLett.123.170503.

[29] Ang Li, Samuel Stein, Sriram Krishnamoorthy & James Ang (2022): *QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation*, doi:10.48550/arXiv.2005.13018. arXiv:2005.13018.

[30] Yongshang Li, Yu Zhang, Mingyu Chen, Xiangyang Li & Peng Xu (2023): *Timing-Aware Qubit Mapping and Gate Scheduling Adapted to Neutral Atom Quantum Computing*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1–1, doi:10.1109/TCAD.2023.3261244.

[31] Ivaylo S. Madjarov, Jacob P. Covey, Adam L. Shaw, Joonhee Choi, Anant Kale, Alexandre Cooper, Hannes Pichler, Vladimir Schkolnik, Jason R. Williams & Manuel Endres (2020): *High-Fidelity Entanglement and Detection of Alkaline-Earth Rydberg Atoms*. Nature Physics 16(8), pp. 857–861, doi:10.1038/s41567-020-0903-z.

[32] M Morgado & S Whitlock (2021): *Quantum simulation and computing with Rydberg-interacting qubits*. AVS Quantum Science 3(2), doi:10.1116/5.0036562. arXiv:https://pubs.aip.org/avs/aqs/article-pdf/doi/10.1116/5.0036562/19739152/023501_1_online.pdf.

[33] Priyanka Mukhopadhyay (2024): *Synthesizing Toffoli-optimal quantum circuits for arbitrary multi-qubit unitaries*. arXiv preprint arXiv:2401.08950. Available at `https://doi.org/10.48550/arXiv.2401.08950`.

[34] M. Müller, I. Lesanovsky, H. Weimer, H. P. Büchler & P. Zoller (2009): *Mesoscopic Rydberg Gate Based on Electromagnetically Induced Transparency*. Physical Review Letters 102(17), p. 170502, doi:10.1103/PhysRevLett.102.170502.

[35] M. A. Norcia, H. Kim, W. B. Cairncross, M. Stone, A. Ryou, M. Jaffe, M. O. Brown, K. Barnes, P. Battaglino, A. Brown, K. Cassella, C.-A. Chen, R. Coxe, D. Crow, J. Epstein, C. Griger, E. Halperin, F. Hummel, A. M. W. Jones, J. M. Kindem, J. King, K. Kotru, J. Lauigan, M. Li, M. Lu, E. Megidish, J. Marjanovic, M. McDonald, T. Mittiga, J. A. Muniz, S. Narayanaswami, C. Nishiguchi, T. Paule, K. A. Pawlak, L. S. Peng, K. L. Pudenz, A. Smull, D. Stack, M. Urbanek, R. J. M. van de Veerdonk, Z. Vendeiro, L. Wadleigh, T. Wilkason, T.-Y. Wu, X. Xie, E. Zalys-Geller, X. Zhang & B. J. Bloom (2024): *Iterative Assembly of ^{171}Yb Atom Arrays in Cavity-Enhanced Optical Lattices*, doi:10.48550/arXiv.2401.16177. arXiv:2401.16177.

[36] Natalia Nottingham, Michael A. Perlin, Ryan White, Hannes Bernien, Frederic T. Chong & Jonathan M. Baker (2023): *Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures*, doi:10.48550/arXiv.2307.14996. arXiv:2307.14996.

[37] Tirthak Patel, Daniel Silver & Devesh Tiwari (2022): *Geyser: A Compilation Framework for Quantum Computing with Neutral Atoms*. In: Proceedings of the 49th Annual International Symposium on Computer Architecture, ISCA '22, Association for Computing Machinery, New York, NY, USA, pp. 383–395, doi:10.1145/3470496.3527428.

[38] Lars Pause, Lukas Sturm, Marcel Mittenbühler, Stephan Amann, Tilman Preuschoff, Dominik Schäffner, Malte Schlosser & Gerhard Birkl (2023): *Supercharged Two-Dimensional Tweezer Array with More than 1000 Atomic Qubits*, doi:10.48550/arXiv.2310.09191. arXiv:2310.09191.

[39] Tom Peham, Lukas Burgholzer & Robert Wille (2022): *Equivalence checking of quantum circuits with the ZX-calculus*. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 12(3), pp. 662–675, doi:10.1109/JETCAS.2022.3202204.

[40] Nils Quetschlich, Lukas Burgholzer & Robert Wille (2023): *MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing*. Quantum 7, p. 1062, doi:10.22331/q-2023-07-20-1062.

[41] M Saffman (2016): *Quantum computing with atomic qubits and Rydberg interactions: progress and challenges*. Journal of Physics B: Atomic, Molecular and Optical Physics 49(20), p. 202001, doi:10.1088/0953-4075/49/20/202001. Available at `https://dx.doi.org/10.1088/0953-4075/49/20/202001`.

[42] M. Saffman, T. G. Walker & K. Mølmer (2010): *Quantum Information with Rydberg Atoms*. Reviews of Modern Physics 82(3), pp. 2313–2363, doi:10.1103/RevModPhys.82.2313.

[43] Mark Saffman (2019): *Quantum Computing with Neutral Atoms*. National Science Review 6(1), pp. 24–25, doi:10.1093/nsr/nwy088.

[44] Ludwig Schmid, David F Locher, Manuel Rispler, Sebastian Blatt, Johannes Zeiher, Markus Müller & Robert Wille (2024): *Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts*. Quantum Science and Technology 9(3), p. 033001, doi:10.1088/2058-9565/ad33ac. Available at `https://dx.doi.org/10.1088/2058-9565/ad33ac`.

[45] Ludwig Schmid, Sunghye Park, Seokhyeong Kang & Robert Wille (2023): *Hybrid Circuit Mapping: Leveraging the Full Spectrum of Computational Capabilities of Neutral Atom Quantum Computers*, doi:10.48550/arXiv.2311.14164. arXiv:2311.14164.

[46] Adam L. Shaw, Ran Finkelstein, Richard Bing-Shiun Tsai, Pascal Scholl, Tai Hyun Yoon, Joonhee Choi & Manuel Endres (2024): *Multi-Ensemble Metrology by Programming Local Rotations with Atom Movements*. Nature Physics, pp. 1–7, doi:10.1038/s41567-023-02323-w.

[47] VV Shende, SS Bullock & IL 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.

[48] Will Simmons (2021): *Relating Measurement Patterns to Circuits via Pauli Flow*. Electronic Proceedings in Theoretical Computer Science 343, p. 50–101, doi:10.4204/eptcs.343.4.

[49] Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann & Wolfgang Gehrke (2023): *Reducing 2-QuBit Gate Count for ZX-Calculus based Quantum Circuit Optimization*. Electronic Proceedings in Theoretical Computer Science 394, pp. 29–45, doi:10.4204/EPTCS.394.3.

[50] Korbinian Staudacher, Ludwig Schmid, Johannes Zeiher, Robert Wille & Dieter Kranzlmüller (2024): *Multi-Controlled Phase Gate Synthesis with ZX- Calculus*, doi:10.5281/zenodo.10730427.

[51] Daniel Bochen Tan, Dolev Bluvstein, Mikhail D. Lukin & Jason Cong (2024): *Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors*. Quantum 8, p. 1281, doi:10.22331/q-2024-03-14-1281.

[52] Daniel Bochen Tan, Shuohao Ping & Jason Cong (2024): *Depth-Optimal Addressing of 2D Qubit Array with 1D Controls Based on Exact Binary Matrix Factorization*. arXiv:2401.13807.

[53] Renaud Vilmart (2019): *A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*, pp. 1–10. doi:10.1109/LICS.2019.8785765.

[54] Hanrui Wang, Pengyu Liu, Bochen Tan, Yilian Liu, Jiaqi Gu, David Z. Pan, Jason Cong, Umut Acar & Song Han (2023): *FPQA-C: A Compilation Framework for Field Programmable Qubit Array*, doi:10.48550/arXiv.2311.15123. arXiv:2311.15123.

[55] Hanrui Wang, Bochen Tan, Pengyu Liu, Yilian Liu, Jiaqi Gu, Jason Cong & Song Han (2023): *Q-Pilot: Field Programmable Quantum Array Compilation with Flying Ancillas*, doi:10.48550/arXiv.2311.16190. arXiv:2311.16190.

[56] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. *arXiv preprint arXiv:2012.13966*. Available at `https://doi.org/10.48550/arXiv.2012.13966`.

[57] Shihao Zhang, Junda Wu & Lvzhou Li (2023): *Characterization, synthesis, and optimization of quantum circuits over multiple-control Z-rotation gates: A systematic study*. *Phys. Rev. A* 108, p. 022603, doi:10.1103/PhysRevA.108.022603. Available at `https://link.aps.org/doi/10.1103/PhysRevA.108.022603`.

A Gadget insertion

Inserting YZ measurements on the outputs preserves gflow:

Corollary 1. *Let $(g,\prec)$ be a gflow for (G,I,O,λ) and let $W \subseteq O$. Then (G',I,O,λ'), where $G' = (V',E')$ with $V' = V \cup \{x\}$, $\lambda'(x) = YZ$ and $E' = E \cup \{(x,w)|w \in W\}$ has a gflow $(g',\prec')$ with following properties:*

- $g'(x) = \{x\}$,
- $\forall v \in V$: $g'(v) = g(v)$,
- $\prec'$ *is the transitive closure of* $\prec \cup \{(x,v)|v \in O\} \cup \{(v,x)|v \in V\backslash O\}$.

Proof. The only new correction set in g' is $g'(x) = \{x\}$, for all other vertices, it is the same as in g. Therefore, all conditions except $(g2)$ are trivially satisfied. For $(g2)$, we need to distinguish two cases for the new vertex x and vertices $v \in V\backslash\{x\}$:

- $x \in Odd(g(v))$: By definition, x is the last element in the partial order $\prec'$ of all non-outputs, thus $(g2)$ holds.
- $v \in Odd(g(x))$: $x \prec v$ holds, because $g(x) = \{x\}$ and $Odd(\{x\})$ only contains outputs which we chose to be after x in the partial order.

Note that $x \notin Odd(g(x))$ by definition. □

B Alternative proof of Theorem 1

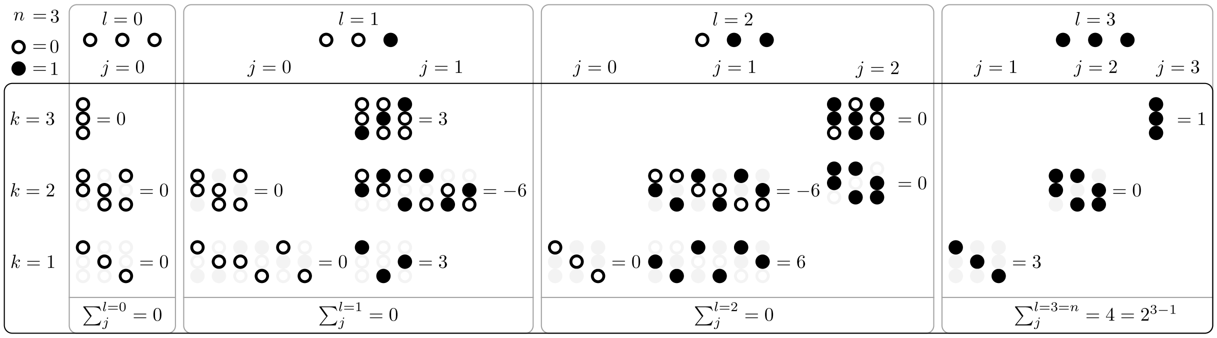
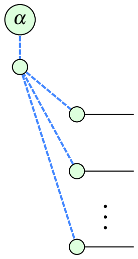


Figure 3: Illustration of the possible combinations and their contribution in Equation (12) for $n = 3$. For each possible value of $l = 0,\dots,3$ the combinations for the possible $k \leq n$ and $j \leq l$ are illustrated as three-circle circles, and their contribution to the sum is computed. The final row shows that the sums of the contributions fulfill the condition of Lemma 2.

This section provides an alternative, combinatorial proof of Theorem 1 instead of using the graphical approach discussed in the main part of the work. The overarching idea is to find a closed formula for the unitary defined by the ZX illustration by summing the corresponding phase contributions and showing that this corresponds to $\mathrm{diag}(1,1,\dots,e^{i\alpha})$ for an arbitrary number of qubits n.

To find a closed formula for Theorem 1 consider the definition of a single phase gadget and its corresponding unitary action on the n-qubit basis states according to [23]:



$$U\left|x_1,\ldots,x_n\right\rangle = e^{i\alpha(x_1\oplus\cdots\oplus x_n)}\left|x_1,\ldots,x_n\right\rangle \quad , \tag{9}$$

where the binary $x_i \in \{0,1\}$, $i = 1,\ldots,n$ label the basis states and $\oplus$ is the binary sum modulo two, i.e. XOR. Note that all phase gadgets of Theorem 1 can be written in such a way, summing only the x_i connected by Hadamard wires to the phase. The single qubit phases give an additional contribution of $e^{i\alpha x_j}\left|x_1,\ldots,x_n\right\rangle$ for each applied qubit j, corresponding to a single-qubit phase gadget.

As the binary sum does not depend on the order of the x_i but only on their value, we introduce the following notation, where we assume that l entries in the sum are non-zero, resulting in a non-zero-sum whenever l is odd:

$$x_1\oplus\ldots\oplus x_n \quad = \quad \frac{(-1)^{l+1}+1}{2} \quad = \quad \mathrm{mod}_2(l) \quad = \quad \begin{cases}1 & ,l \text{ is odd}\\ 0 & ,l \text{ is even}\end{cases} \quad . \tag{10}$$

Considering again Equation (2) one can see that there are two contributions to the total accumulated phase. First, for each qubit, a single-qubit phase α is added. Second, for each possible combination of length k of all the x_i, there is a phase gadget with phase $(-1)^{k+1}\alpha$. For the $\mathrm{C_1P}(2\alpha)$ gate, this reduces to a single $k=2$ phase gadget of phase $-\alpha$. For the $\mathrm{C_2P}(4\alpha)$ gate, on the other hand, there are $\binom{3}{2}=3$ phase gadgets of size $k=2$ and angle $-\alpha$ and a single ($\binom{3}{3}=1$) $k=3$ gadget with angle α. In general, for a n qubit gate, there are $\binom{n}{k}$ combinations for phase gadgets of size $k=1,\ldots,n$. A combination contributes to the total phase if the number l of non-zero entries in the direct sum of Equation (9) is odd, resulting in an additional phase $\pm\alpha$. Otherwise, the combination does not contribute to the phase. To express the number of possible combinations depending on l, the $\binom{n}{k}$ combinations for a length k can also be expressed as choosing j variables from the l one-valued variables and choosing $k-j$ variables from the $n-l$ zero-valued variables and summing over all possible j:

$$\binom{n}{k} = \sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j}. \tag{11}$$

This relation is known as the *Vandermonde identity*. An illustration of the possible combinations depending on k and l is shown in Figure 3 for the simple case $n=3$.

Multiplying the unitaries of all these phase gadgets corresponds to summing the accumulated phases with the appropriate sign, converting the problem of Theorem 1 into a summation of the appropriate phases with a corresponding sign. For the total structure to represent a multi-controlled phase gate, the phases have to vanish for all possible basis states $\left|x_1,\ldots,x_n\right\rangle$ except for $\left|1,\ldots 1\right\rangle$ where they have to sum to $2^{n-1}\alpha$.

Based on these considerations, an equivalent statement of Theorem 1 can be formulated, dropping the illustrations of the ZX-calculus and formulating the multi-controlled phase gate extraction as a purely combinatorial problem, focusing on the accumulated phase. Theorem 1 then directly follows from this Lemma based on the considerations above and using $e^{i\alpha\cdot 0}=1$.

Lemma 2 (Multi-controlled phase gate)**.** *For n binary variables $x_1 \dots x_n$ of which l is non-zero, summing the modulo two sum over all possible combinations of length k with sign $(-1)^{k+1}$, it holds:*

$$\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{\min(k,l)}\binom{l}{j}\binom{n-l}{k-j}\mathrm{mod}_2(j) = \begin{cases} 2^{n-1} & , \textit{if n=l} \\ 0 & , \textit{else} \end{cases} \quad . \tag{12}$$

Where the $\min(k,l)$ *results from the fact that the number of ones in the current combination j cannot be larger than the length k of the combination, nor the total number of ones l available.*

Proof. The proof is two-fold. First, the $n = l$ case is shown explicitly, while the case $n \neq l$ is shown by induction in both variables n and l. Also, note that

$$\sum_{j=0}^{\min(k,l)}\binom{l}{j}\binom{n-l}{k-j} = \sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j} = \sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j} \tag{13}$$

as for $k > l$ the first binomial coefficient vanishes in all additional cases, and for $l > k$ the second, as $k - j < 0$ in these cases. This also becomes clear from the illustration in Figure 3 where the vanishing combinations are either non-existent or only contain zero entries. These identities are used multiple times in the following proof.

Case $n = l$**:**

If $n = l$ all variables are one and, therefore, $\min(k,l) = k$. Furthermore, the second binomial coefficient is non-zero only in the $j = k$ case, where it equals 1. This results in

$$\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}\frac{(-1)^{k+1}+1}{2} = \frac{1}{2}\left[\sum_{k=1}^{n}\binom{n}{k}(-1)^{k+1} + \sum_{k=1}^{n}\binom{n}{k}\right] = \frac{1}{2}\left[1 + 2^n - 1\right] = 2^{n-1},$$

using the regular and the alternating binomial sum, directly showing the first part of Lemma 2.

Case $n \neq l$**:**

Proving Equation (12) for arbitrary n and $l < n$ is done by induction. Therefore, showing the term to be zero for $l = 0$ and arbitrary n as the base case and then performing the induction step both in n and in l.
Base case $l = 0, n$*:* In this case the second sum reduces to the $j = 0$ case, trivially giving zero, independetly for all n. In other words, as all variables are zero, the sum in Equation (9) always gives zero.
Induction step $n \to n+1$*:* Inserting this step into Equation (9) and using the recurrence relation of the binomial coefficient $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$ and the abbreviation $\xi := \mathrm{mod}_2(j)$ one gets

$$\sum_{k=1}^{n+1}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j-1}\xi + (-1)^n + \underbrace{\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\xi}_{=0 \text{ (Base case)}}$$

$$\cancel{+\,(-1)^n\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{n+1-j}\xi}$$

$$= \sum_{k=0}^{n}(-1)^k\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\frac{(-1)^j+1}{2} = \sum_{k=0}^{n}(-1)^{k+1}\sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j}\frac{(-1)^{j+1}+1-2}{2}$$

$$
\begin{aligned}
&= \underbrace{(-1)^1 \binom{l}{0}\binom{0-l}{0-0}\frac{-2}{2}}_{k=0 \text{ case}} + \underbrace{\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\xi}_{=0 \text{ (Base case)}} + \sum_{k=0}^{n}(-1)^{k+1}\underbrace{\sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j}}_{\text{Vandermonde}} \\
&= 1+0-1=0 \quad ,
\end{aligned}
$$

writing the $n+1$ term separately to recover the base case. The term in the second line vanishes due to the lower part of the binomial coefficient always being larger than the top part. Going to the third line, an index shift in k is performed, and then using Equation (13) with an additional reinserted (-1) factor to recover the original form of ξ. Separating the $k=0$ case and the additional introduced –2 term, the base case can be inserted again, resulting in zero after using again the Vandermonde identity and the alternating binomial sum. In a similar fashion, also the induction step in l can be shown.

Induction step $l \to l+1$: With the base case for arbitrary n and the corresponding induction step in n, one can, in the following, assume the base case to be true for arbitrary n, in particular for $n' = n-1$. Performing the induction step in $l \to l+1$ and again using the recurrence relation, one gets

$$
\begin{aligned}
&\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l+1}\binom{l+1}{j}\binom{n-(l+1)}{k-j}\xi \\
&\quad = \sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l+1}\binom{l}{j-1}\binom{n'-l}{k-j}\xi + \underbrace{\sum_{k=1}^{n'}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{k-j}\xi}_{=0 \text{ (Base case for } n')} \\
&\qquad + \underbrace{\cancel{\sum_{k=1}^{n}(-1)^{k+1}\binom{l}{l+1}\binom{n'-l}{k-l-1}}}_{j=l+1 \text{ case}} + \underbrace{\cancel{(-1)^{n+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{n-j}\xi}}_{k=n \text{ case}} \\
&\quad = \sum_{k=1}^{n}(-1)^{k+1}\sum_{j=-1}^{l}\binom{l}{j}\binom{n'-l}{k-j-1}\frac{(-1)^j+1}{2} \\
&\quad = \sum_{k=0}^{n'}(-1)^{k}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{k-j}\frac{(-1)^{j+1}+1}{2} \\
&\quad = 0 \quad ,
\end{aligned}
$$

with the base case for n' used in the second line and the additional terms vanishing because either the first or the second binomial coefficient is zero. In the following two lines, first, an index shift in j is performed, and then, secondly, in k. The result is the same formula as in the previous calculation, just for n' and therefore, also vanishes.

This concludes the induction step in l, concluding also the $n \neq l$ case of Equation (12) and therefore proving Lemma 2.

□

Pauli Flow on Open Graphs with Unknown Measurement Labels

Piotr Mitosek
School of Computer Science, University of Birmingham
pbm148@student.bham.ac.uk

One-way quantum computation, or measurement-based quantum computation, is a universal model of quantum computation alternative to the circuit model. The computation progresses by measurements of a pre-prepared resource state together with corrections of undesired outcomes via applications of Pauli gates to yet unmeasured qubits. The fundamental question of this model is determining whether computation can be implemented deterministically. Pauli flow is one of the most general structures guaranteeing determinism. It is also essential for polynomial time ancilla-free circuit extraction. It is known how to efficiently determine the existence of Pauli flow given an open graph together with a measurement labelling (a choice of measurements to be performed).

In this work, we focus on the problem of deciding the existence of Pauli flow for a given open graph when the measurement labelling is unknown. We show that this problem is in RP by providing a random polynomial time algorithm. To do it, we extend previous algebraic interpretations of Pauli flow, by showing that, in the case of X and Z measurements only, flow existence corresponds to the right-invertibility of a matrix derived from the adjacency matrix. We also use this interpretation to show that the number of output qubits can always be reduced to match the number of input qubits while preserving the existence of flow.

1 Introduction

One-way quantum computation, or measurement-based quantum computation (MBQC), is a quantum computation model alternative to the circuit model, yet at least as powerful [30, 31, 32]. The computation is performed entirely by a series of measurements on a pre-prepared resource state. MBQC is expected to become critical for quantum communication applications. For instance, MBQC is used in blind quantum computation [4], a quantum protocol allowing a client to outsource part of the computation to a server without compromising the client's privacy. However, the quantum measurements are random by their nature, which means that in each step two different outcomes may be obtained, where only one is the desired outcome. Therefore, it is the fundamental problem of MBQC to determine whether the desired computation can be performed deterministically.

A structure essential for deterministic MBQC is flow defined on a labelled open graph – a graph state with designated input and output qubits together with a choice of measurements for non-outputs. The computation is performed by a series of measurements of non-outputs in one of six settings: planar XY, XZ, and YZ measurements and Pauli X, Y, and Z measurements. After each measurement, a correction might be applied which is intended to fix a potential undesired outcome, bringing the open graph into a state equivalent to one that would be reached if a desired outcome was observed. The corrections are performed by applying Pauli gates to yet unmeasured qubits. One of the notions of determinism for such computation corresponds to the existence of flow structure in the labelled open graph. There are many different types of flow, ranging from the causal flow [8], through generalized flow [5], all the way to the Pauli flow [5] and its extended variant [27]. Every causal flow is also a generalized flow. Every

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 117–136, doi:10.4204/EPTCS.406.6

generalized flow is also a Pauli flow. However, the reverses are not true, thus making Pauli flow the most universal.

The flow is also crucial when working with ZX calculus [7, 37], a graphical calculus for quantum computation, where the notions of flow correspond to the most general subclasses of diagrams for which a polynomial time ancilla-free circuit extraction algorithm exists [2, 34]. While diagrams from graphical calculi are easier to work with than circuits, they are not a notion of computation understood by quantum computers. For that, a corresponding circuit must be found, however, circuit extraction in general is #P-hard [11]. In [13], circuits are optimized by translating them to ZX, simplifying diagrams, and translating them back to circuits. Other versions of this approach have appeared since for example in [35]. There, diagram transformations necessarily must preserve flow, which was a subject of for instance [23, 24].

While the existence of different flow variants can be efficiently determined for a given labelled open graph [10, 9, 26, 2, 34], many questions remain open. For instance, what are other properties of labelled open graphs with flow? Previous works looking at the necessary properties the graph must satisfy to contain flow include [25, 22, 34]. We contribute to the general picture of what flow is and which graphs exhibit it.

Results overview In this work, we show that given an (unlabelled) open graph it is possible to efficiently determine whether there exists a choice of measurements resulting in flow. In particular, we show the following problem is in RP, meaning that there exists a random polynomial-time algorithm solving it.

> **FlowSearch**
> **Input:** An open graph (G,I,O).
> **Output:** $True$ if there exists a measurement labelling λ such that the labelled open graph (G,I,O,λ) exhibits Pauli flow, and $False$ otherwise.

To show that, we adapt the algebraic interpretation of flow from [25]. In that paper, the authors relate the existence of generalised flow for XY planar measurements only to the existence of the right inverse of a particular matrix satisfying additional requirements. We relate Pauli flow in the case of only X and Z measurement to just right invertibility. Then, we use the algebraic interpretation to reduce **FlowSearch** to **MaxRank**, a problem about maximizing the rank of a matrix whose entries can contain variables. Our approach is limited to finding labelling consisting of Pauli X and Z measurements. The corresponding measurement scheme is Clifford. Thus, the current algorithm may not be ideal for searching more complex labels which make the corresponding labelled open graph have Pauli flow. However, as we will prove, the graph having Pauli flow for some measurement labels also has flow for labels consisting of only Pauli X and Z measurements. In other words, if a graph does not have flow for some label consisting of just X and Z measurements then it cannot have flow for any label. Therefore, our approach is sufficient to solve **FlowSearch**.

Further, we use the algebraic interpretation of flow to explore the conditions on the sets of inputs and outputs necessary to have flow. In particular, we prove that if a labelled open graph has Pauli flow, then some outputs can be removed and the resulting open graph will still have Pauli flow for some measurement labelling.

Structure In section 2, we present the background. Section 3 is the main part of the paper. There, we formally describe our results and prove them. Finally, in section 4, we discuss the conclusions and possible further work. We include some technical proofs and additional examples in the appendices.

2 Background

In this section, we first look at the general concept of measurement-based quantum computation, where we define Pauli flow. Next, we explain computational complexity definitions critical in our proofs.

2.1 Measurement-Based Quantum Computation

As outlined earlier, we focus on the version of MBQC where measurements are performed on open graphs and the allowed measurements are single-qubit: Pauli measurements or planar measurements from XY, YZ, or XZ planes. The description of the computation is given in the form of measurement patterns. The labelled open graphs can be viewed as runnable measurement patterns without corrections [2].

Definition 2.1 (Open Graph)**.** An **open graph** is a triple (G,I,O) consisting of:

- an undirected graph $G=(V,E)$,
- a set of **inputs** $I \subseteq V$,
- a set of **outputs** $O \subseteq V$.

We define the sets of **non-inputs** $\overline{I} := V \setminus I$, **non-outputs** $\overline{O} := V \setminus O$ and **internal vertices** $B := V \setminus (I \cup O) = \overline{I} \cap \overline{O}$.

An **odd neighbourhood** of a set of vertices A is denoted $Odd(A)$ and consists of all vertices neighbouring an odd number of elements of A: $Odd(A) := \{v \in V \mid \#\{a \in A \mid va \in E\} \text{ is odd}\}$.

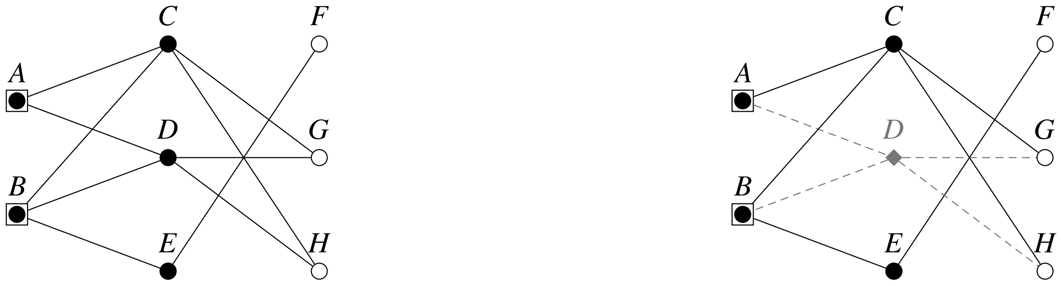
Definition 2.2. A **measurement labelling** for an open graph (G,I,O) is any function λ sending non-outputs $\overline{O}$ to labels $\{X,Y,Z,XZ,XY,YZ\}$, satisfying $\lambda(v) \in \{X,XY,Y\}$ for all $v \in I \setminus O$. A **labelled open graph** is a quadruple (G,I,O,λ) where (G,I,O) is an open graph and λ is a measurement labelling.

An open graph is prepared by entangling input qubits, corresponding to the vertices in I, with qubits prepared in $|+\rangle$ corresponding to the vertices in $\overline{I}$, by applying CZ gate between pair of qubits corresponding to each edge in the graph. Since the CZ gates commute, the order of CZ applications does not matter. When the set of inputs is empty, the notion of an open graph collapses to the graph state. Next, qubits corresponding to the elements of $\overline{O}$ are measured according to the measurement labelling. See figure 1 for an example.

The computation given in the form of the measurement pattern depends on the outcomes of the measurements. After each measurement, an undesired outcome may occur and it must be corrected. The corrections are performed by applying Pauli X and Z gates to the yet unmeasured qubits. The result of a chosen measurement outcome, together with corrections, is called a branch. There are many different notions of determinism [5, 25]. The one we focus on is the strong, uniform, and stepwise determinism – all branches of the computation are equal up to a global phase, for any choice of measurement angles, and the intermediate patterns after performing a subset of measurements also have these properties [5, 25, 2]. The measurement pattern is strongly, uniformly, and stepwise deterministic when the corresponding labelled open graph has Pauli flow [5]. We define the notion of Pauli flow (based on [34]).

Definition 2.3 (Pauli flow)**.** A **Pauli flow** for a labelled open graph (G,I,O,λ) is a pair $(c,\prec)$ where c is a function $\overline{O} \to \mathcal{P}(\overline{I})$ and $\prec$ is a strict partial order on $\overline{O}$, such that for all $u \in \overline{O}$:

(P1) $\forall v \in c(u).u \neq v \wedge \lambda(v) \notin \{X,Y\} \Rightarrow u \prec v$
(P2) $\forall v \in Odd(c(u)).u \neq v \wedge \lambda(v) \notin \{Y,Z\} \Rightarrow u \prec v$
(P3) $\forall v \in \overline{O}.\neg(u \prec v) \wedge u \neq v \wedge \lambda(v) = Y \Rightarrow (v \in c(u) \Leftrightarrow v \in Odd(c(u)))$
(P4) $\lambda(u) = XY \Rightarrow u \notin c(u) \wedge u \in Odd(c(u))$
(P5) $\lambda(u) = XZ \Rightarrow u \in c(u) \wedge u \in Odd(c(u))$



(a) A labelled open graph with two inputs and three outputs. All non-outputs are X labelled.

(b) The labelled open graph from 1a, but with vertex D labelled Z.

Figure 1: Examples of labelled open graphs. Outputs are denoted by empty circles, X labelled non-outputs by filled circles and inputs by a square. The Z labelled vertices are denoted with grey diamonds and edges including them are dashed and grey.

(P6) $\lambda(u) = YZ \Rightarrow u \in c(u) \wedge u \notin Odd(c(u))$
(P7) $\lambda(u) = X \Rightarrow u \in Odd(c(u))$
(P8) $\lambda(u) = Z \Rightarrow u \in c(u)$
(P9) $\lambda(u) = Y \Rightarrow (u \in c(u) \oplus u \in Odd(c(u)))$, where $\oplus$ stands for XOR.

We call c the **correction function** and sets $c(v)$ for $v \in \overline{O}$ are called the **correction sets**.

Example 2.4. Consider open graphs from figure 1. The graph from 1a does not have Pauli flow. On the other hand, the graph from 1b has Pauli flow. For instance, taking $D \prec A$ and $D \prec B$ with the following correction function result in the flow:

$$\begin{aligned}
&c(A) = \{C,E\}, && Odd(c(A)) = \{A,F,G,H\}, && c(B) = \{E\}, && Odd(c(B)) = \{B,F\},\\
&c(C) = \{G\}, && Odd(c(C)) = \{C,D\}, && c(D) = \{D\}, && Odd(c(D)) = \{A,B,G,H\},\\
&c(E) = \{F\}, && Odd(c(E)) = \{E\}.
\end{aligned}$$

In MBQC, the meaning of the Pauli flow is as follows. The prepared open graph state is measured according to the partial order. When a measurement of a vertex u results in an undesired outcome, then the measurement error can be fixed by applying the X gate to all vertices in $c(u)$. The flow guarantees that each correction is physically possible and that every error can be corrected independently of the outcomes of the previous measurements. In other words, the MBQC becomes deterministic. For a detailed explanation, see [34].

2.2 Computational Complexity

We assume familiarity with standard complexity terminology. We are mainly concerned about the **random polynomial time** class RP, where randomness is permitted with one-sided bounded error:

Definition 2.5. The class of RP consists of problems A solvable by a non-deterministic Turing Machine M that given input a proceeds as follows:

- if a is a “NO” instance, then M always rejects a,
- if a is a “YES” instance, then M accepts a on at least half of its computation paths.

For a problem in RP there exists a polytime algorithm with random number generator access that for "NO" instances always returns "NO" and for "YES" instances it returns "YES" with probability at least $\frac{1}{2}$ and otherwise it returns "NO". This last case is the error of the computation and happens with the probability smaller than $\frac{1}{2}$. By running the algorithm multiple times it is possible to reduce the error probability – 50 runs results in error probability below $\frac{1}{2^{50}}$ which is sufficient for all practical applications.

The following is the essential problem in our work. The definition is adapted from [6] (in contrast, we do not require the input matrix to be square).

MaxRank
Fixed: A commutative ring R, and subsets $E, S \subseteq R$ of entries and solutions.
Input: Natural numbers m, n, t, r and $m \times n$ matrix M with entries from $E \cup \{x_1, \dots, x_t\}$.
Output: $True$ if $\operatorname{rank} M(a_1, \dots, a_t) \geq r$ for some $a_1 \dots a_t \in S^t$, and $False$ otherwise.

$M(a_1, \dots, a_t)$ stands for the matrix with substituted variables $x_1 \mapsto a_1, \dots, x_t \mapsto a_t$. We skip the specification of m, n, t, as these numbers are explicit from the input matrix. Thus, we will write instances of **MaxRank** as pairs (M, r) of the matrix and the desired minimal rank under some valuation.

In [6], it is shown that when R is a finite field and each variable occurs at most once, the problem is in RP. We extend their approach to show that the problem is in RP also when each variable appears in at most one row or one column. For that, we need the notion of multi-affine polynomials (adapted from [6]). In general, the **MaxRank** problem with R being a finite field is NP-complete [6].

Definition 2.6. A multivariable polynomial is **multi-affine** when each variable has a degree at most one.

By extending the proof [6, Theorem 28], we get the following version. Here, the entries of the matrix are allowed to be given by multi-affine expressions over $E \cup \{x_1, \dots, x_t\}$, not just elements of the set.

Theorem 2.7. The following version of **MaxRank** is in RP:

- $R = E = S = \mathbb{F}_s$ is a finite field,
- each variable appears in at most one row or at most one column[1],
- the entries are given by (polynomially long) multi-affine expressions over $E \cup \{x_1, \dots, x_t\}$.

We prove the above theorem in the appendix A, where we also give an example and talk about other known results for **MaxRank** and similar problems.

3 Finding labelling resulting in Pauli flow

We start by formally defining theorems capturing our results. We show that given an open graph (G, I, O), there exists a random polytime algorithm deciding the existence of measurement labelling λ such that the labelled open graph (G, I, O, λ) has Pauli flow, i.e we show that **FlowSearch** defined in the introduction is in RP.

Theorem 3.1. FlowSearch is in RP.

Example 3.2. Consider the open graph from figure 1, ignoring the labelling. When considered as an instance of **FlowSearch**, the answer is $True$, because the labelling from figure 1b results in Pauli flow.

To show the above theorem, we expand on the known [25] correspondence between flow and certain algebraic properties of various matrices. We also show that given an open graph with more outputs than inputs, it is always possible to reduce the number of outputs to match the number of inputs, while preserving the flow.

[1] That is if a variable appears in entries at positions $(a_1, b_1), (a_2, b_2), (a_3, b_3), \dots$, then either $a_1 = a_2 = a_3 = \dots$ or $b_1 = b_2 = b_3 = \dots$.

Theorem 3.3. Suppose (G,I,O,λ) has Pauli flow and $|O| > |I|$. Then there exists a subset $O' \subseteq O$ such that $|O'| = |I|$ and a labelling λ' such that (G,I,O',λ') has Pauli flow.

Labelled open graphs with Pauli flow and equal numbers of inputs and outputs have multiple elegant properties. Firstly, they correspond to a circuit without ancillas and hence a unitary. Further, the graph and flow can be "reversed" ([25, Theorem 3.4] for generalized flow theorem). Next, there is a unique correction function in the focussed Pauli flow. In the case of X and Z labels only, the existence of such a correction function corresponds to the invertibility of what we call the flow matrix, rather than just right-invertibility. Finding the inverse can be implemented with faster algorithms than Gaussian elimination. Thus, it is useful to transform labeled open graphs with more outputs than inputs in order to equalise both sizes.

The remainder of this section are the proofs of the theorems 3.1 and 3.3. The first proof is divided into three subsections. In the first one, we do preliminary simplifications of the **FlowSearch** problem. The second subsection presents an algebraic interpretation of the Pauli flow, by giving a correspondence between Pauli flow and matrix invertibility. Finally, in the third subsection, we show how **FlowSearch** instances can be transformed into **MaxRank** instances, and that those instances can be solved by a random polytime algorithm. The proof of theorem 3.3 is presented in the final subsection.

3.1 Reducing measurement options

We start by reducing the number of possible options for the measurement basis to just Pauli measurements.

Theorem 3.4. Suppose that a labelled open graph (G,I,O,λ) has Pauli flow. Then, there exists $\lambda' : \overline{O} \to \{X,Y,Z\}$ such that (G,I,O,λ') has Pauli flow.

Proof. We start by fixing a Pauli flow $(c,\prec)$ on (G,I,O,λ). The conditions for a planar measurement XY combine the requirements for the two Pauli measurements X and Y. Hence, swapping XY measurements to X preserves $(c,\prec)$ as the Pauli flow. Similarly, we can swap XZ to X and YZ to Z. □

There can be many different Pauli flows for a single labelled open graph. However, there exists a special type of Pauli flow known as the **focussed Pauli flow**. The following definition is based on [34, Definition 4.3].

Definition 3.5 (Focussed Pauli flow)**.** The Pauli flow $(c,\prec)$ is **focussed** when for all $v \in \overline{O}$ the following hold:

(F1) $\forall w \in (\overline{O} \setminus \{v\}) \cap c(v).\lambda(w) \in \{XY,X,Y\}$
(F2) $\forall w \in (\overline{O} \setminus \{v\}) \cap Odd(c(v)).\lambda(w) \in \{XZ,YZ,Y,Z\}$
(F3) $\forall w \in (\overline{O} \setminus \{v\}).\lambda(w) = Y \Rightarrow (w \in c(v) \Leftrightarrow w \in Odd(c(v)))$

Example 3.6. Consider the Pauli flow from the example 2.4 – it is not focussed as $Odd(c(D))$ contains X labelled vertices A and B. Changing $c(D)$ to $\{C,D\}$ results in $Odd(c(D)) = \emptyset$ and the flow becomes focussed.

Importantly, the existence of flow is equivalent to the existence of focussed Pauli flow, as captured by the following theorem [34, Lemma 4.6]:

Theorem 3.7 ([34, Lemma 4.6])**.** For any open labelled graph, if a Pauli flow exists then there also exists a focussed Pauli flow.

In the case of only Pauli measurements, the conditions from the focussed flow mean that the corrector sets can only consist of X and Y measured vertices and the corrected vertex, while the odd neighbourhoods of the corrector sets can only consist of the Z and Y measured vertices and the corrected vertex.

It is easier to search for the focussed Pauli flow, as the notion of the focussed flow in the case of Pauli bases does not require partial order as captured below. The proof follows from [34, Lemma B.11].

Lemma 3.8. If $(c,<)$ is a focussed Pauli flow for a labelled open graph (G,I,O,λ) with only Pauli measurements, then so is $(c,\emptyset)$.

Further, we can improve theorem 3.4 to only look for X and Z labelling, getting rid of the Y measurements:

Theorem 3.9. Suppose that a labelled open graph (G,I,O,λ) has Pauli flow. Then, there exists $\lambda' : \overline{O} \rightarrow \{X,Z\}$ such that (G,I,O,λ') has Pauli flow.

Proof. Let (G,I,O,λ) have Pauli flow. By theorem 3.4, there is $\lambda_1 : \overline{O} \rightarrow \{X,Y,Z\}$ such that (G,I,O,λ_1) has Pauli flow. By theorem 3.7 and lemma 3.8, (G,I,O,λ_1) has some focussed Pauli flow $(c,\emptyset)$. If there is no u with $\lambda_1(u) = Y$, the thesis follows. Otherwise, consider such u. Define $\lambda_2 : \overline{O} \rightarrow \{X,Y,Z\}$ as follows. $\lambda_2(v) = \lambda_1(v)$ for $v \neq u$. By (P9), $u \in p(u) \oplus u \in Odd(p(u))$. If $u \in p(u)$, we set $\lambda_2(u) = Z$ and when $u \in Odd(p(u))$, we set $\lambda_2(u) = X$. Then (G,I,O,λ_2) has Pauli flow $(c,<)$, where c is the same correction function as for (G,I,O,λ_1) and $<$ is given by $v < u$ for all $v \in \overline{O} \setminus \{u\}$, i.e. u is the last vertex in this order. We verify $(c,<)$ indeed is the Pauli flow. Conditions (P4), (P5), (P6) hold automatically. Similarly, (P7), (P8), (P9) hold for vertices other than u. The choice for the new label of u is done by using (P9) for the old flow, thus ensuring that c works for correction of u with new label. Hence, only (P1), (P2) and (P3) remain. By construction, u is last in the order, hence if $u \in c(v)$ or $u \in Odd(c(v))$, the necessary order condition is guaranteed to hold. When $v \in c(u)$ or $v \in Odd(c(u))$, then the $\lambda_2(v) = \lambda_1(v)$ ensures that no new order requirement appears. Similarly, $\emptyset$ order worked for other pairs of vertices. Thus, (P1), (P2) and (P3) all hold. By focussing the flow and relabelling one vertex labelled with Y at a time, we can get rid of all Y measured vertices, ending the proof. □

Thanks to the above lemmata, instead of looking for λ resulting in Pauli flow $(c,<)$, we can just look for λ into Pauli measurements resulting in a focussed Pauli flow $(c,\emptyset)$.

Finally, we can omit cases with $I \cap O \neq \emptyset$, by reducing those to have $I \cap O = \emptyset$.

Theorem 3.10. Let (G,I,O,λ) be a labelled open graph. Then (G,I,O,λ) has Pauli flow if and only if $(G', I \setminus O, O \setminus I, \lambda)$ does, where G' is G with vertices in $I \cap O$ removed.

Proof. Suppose $v \in I \cap O$. A correction function c in Pauli flow has codomain $\mathcal{P}(\overline{I})$ and $v \in I$ so v cannot be used in any correction set. Further, as $v \in O$, v is not measured and so $c(v)$ is undefined and the partial order does not operate on v. Therefore v does not impact any of the nine flow conditions in any way. □

Combining the above simplifications, we can restrict **FlowSearch** problem to cases (G,I,O) with $I \cap O = \emptyset$, where we are looking for $\lambda : \overline{O} \rightarrow \{X,Z\}$ such that (G,I,O,λ) has focussed Pauli flow.

Since the removal of Z-measured vertices preserves Pauli flow (which we mention later), it is also possible to view the above problem as follows: given (G,I,O), is there an induced open subgraph with the same set of inputs and outputs, that has Pauli flow with only X labels?

3.2 Algebraic interpretation of flow

Unless specified otherwise, from now on, we only consider (G, I, O) with $I \cap O = \emptyset$. The key construction used to translate the **FlowSearch** problem into a linear algebra problem is the reduced adjacency matrix [25] (note, that in that paper the word *induced* is used in place of *reduced*).

Definition 3.11. Let (G, I, O) be an open graph. We define the **adjacency matrix** A_G over $\mathbb{F}_2$ as a $|V| \times |V|$ matrix with $(A_G)_{u,v} = 1$ when $uv \in E$ and 0 otherwise. The **reduced adjacency matrix** $A_G |^{\overline{O}}_{\overline{I}}$ is the $|\overline{O}| \times |\overline{I}|$ minor of the adjacency matrix of G. The minor is obtained by removing the outputs' rows and the inputs' columns.

The key property of the reduced adjacency matrix linking it to the Pauli flow is right-invertibility. In [25], a version lining XY measurements only to the right-invertibility was established. Subsequently, this theorem was extended by Miriam Backens to work for both X and XY measurements:

Theorem 3.12. Let $\mathcal{G} = (G, I, O, \lambda)$ be a labelled open graph with $\lambda(v) \in \{X, XY\}$ for all $v \in \overline{O}$. Then $\mathcal{G}$ has focussed Pauli flow if and only if there exists a directed graph $F = (V, E_F)$ satisfying the following two properties:

- Let $E'_F = \{(u, v) \in E_F \mid \lambda(u) = XY\}$, then the subgraph $F' = (V, E'_F)$ is acyclic,
- $A_G |^{\overline{O}}_{\overline{I}} \cdot A_F |^{\overline{I}}_{\overline{O}} = Id_{\overline{O}}$, where $A_F |^{\overline{I}}_{\overline{O}}$ is the $\overline{I} \times \overline{O}$ minor of A_F obtained by removing inputs' rows and outputs' columns.

Further, the columns of $A_F |^{\overline{I}}_{\overline{O}}$ encode the correction sets of the vertices in $\overline{O}$.

The proof has not yet been published [28]. Note, that in the case of X measurements only, the graph F' in the theorem 3.12 is empty and hence it is automatically acyclic. Hence, the condition simplifies to just the existence of the right inverse.

Corollary 3.13. Let $\mathcal{G} = (G, I, O, \lambda)$ be a labelled open graph with $\lambda(v) = X$ for all $v \in \overline{O}$. Then $\mathcal{G}$ has focussed Pauli flow if and only if $A_G |^{\overline{O}}_{\overline{I}}$ is right-invertible over $\mathbb{F}_2$. Further, the columns of a potential right inverse of $A_G |^{\overline{O}}_{\overline{I}}$ encode the correction sets of the vertices in $\overline{O}$.

See figure 2a for an example of using corollary 3.13 and figure B.1 in appendix B for an example of using theorem 3.12.

The Z labelled vertices can be removed and introduced without affecting flow existence, as captured by the following lemmata. In these, $G[A]$ stands for the subgraph of G induced by vertices in A.

Lemma 3.14 (Removal of Z measured vertex [34, Lemma D.6])**.** Let (G, I, O, λ) be a labelled open graph with Pauli flow and with $\lambda(v) = Z$ for some $v \in \overline{O}$. Then v can be removed without affecting flow existence. In other words, $(G[V \setminus \{v\}], I, O, \lambda|_{\overline{O} \setminus \{v\}})$ has Pauli flow.

Lemma 3.15 (Introduction of Z measured vertex [23, Proposition 4.1])**.** Let (G, I, O, λ) have Pauli flow. Then a new Z measured vertex $v \notin V$ can be added to G, with any edges from v, without affecting flow existence. In other words, any labelled open graph (G', I, O, λ') has Pauli flow, where:

$$V(G') = V \cup \{v\} \qquad G'[V] = G \qquad \lambda'|_{V \setminus O} = \lambda \qquad \lambda'(v) = Z$$

The proof for the removal appears in [34, Lemma D.6], and is a reformulation of [2, Lemma 4.7].

Now, we extend the corollary 3.13 to also capture Z measurements, by considering a slightly different matrix that we call the flow matrix.

Definition 3.16 (Flow matrix)**.** Let $\mathcal{G} = (G, I, O, \lambda)$ be a labelled open graph with $\lambda(v) \in \{X, Z\}$ for all $v \in \overline{O}$. Let G_{disc} be G with Z labelled vertices disconnected from the rest of the graph. We define the **flow matrix** $M_{\mathcal{G}}$ as the sum of the reduced adjacency matrix $A_{G_{disc}} \mid^{\overline{O}}_{\overline{I}}$ and matrix that has 1 at the intersections of v row and v column for all v with $\lambda(v) = Z$, and 0 otherwise.

Theorem 3.17. Let $\mathcal{G} = (G, I, O, \lambda)$ be a labelled open graph with $\lambda(v) \in \{X, Z\}$ for all $v \in \overline{O}$. Then $\mathcal{G}$ has focussed Pauli flow if and only if $M_{\mathcal{G}}$ is right-invertible over $\mathbb{F}_2$.

Proof. Let G_{disc} be as in the definition 3.16. Let $G_X = (V_X, E_X)$ be G (equivalently G_{disc}) with Z-labelled vertices removed from the rest of the graph. Then, the following facts are equivalent:

1. $\mathcal{G}$ has focussed Pauli flow,
2. $\mathcal{G}_X := (G_X, I, O, \lambda_X)$ has focussed Pauli flow, where $\lambda_X = \lambda \mid_{\left\{v \in \overline{O} \mid \lambda(v) = X\right\}}$,
3. $A_{G_X} \mid^{V_X \setminus O}_{V_X \setminus I}$ is right-invertible,
4. $M_{\mathcal{G}_X}$ is right-invertible,
5. $M_{\mathcal{G}_{disc}}$ is right-invertible, where $\mathcal{G}_{disc} = (G_{disc}, I, O, \lambda)$,
6. $M_{\mathcal{G}}$ is right-invertible.

$(1 \Leftrightarrow 2)$: follows from lemmata 3.14 and 3.15 – the Z labelled vertices can be removed and introduced without affecting flow existence; the existences of Pauli flow and focussed Pauli flow are equivalent by theorem 3.7.

$(2 \Leftrightarrow 3)$: follows from corollary 3.13, as in G_X there are no Z labelled vertices.

$(3 \Leftrightarrow 4)$: the notions of reduced adjacency matrix and flow matrix agree in the case of only X measurements, thus $A_{G_X} \mid^{V_X \setminus O}_{V_X \setminus I} = M_{\mathcal{G}_X}$.

$(4 \Leftrightarrow 5)$: if there are no Z labelled vertices in G_{disc}, then $G_{disc} = G_X$ and matrices from 4 and 5 are equal. Otherwise, let $v \in \overline{O}$ be any vertex with $\lambda(v) = Z$. In the flow matrix $M_{\mathcal{G}_{disc}}$, the v row and the v column are 0 everywhere except for the intersection, as v is disconnected from other vertices by assumption. The intersection of the v row and the v column contains 1 by the construction of the flow matrix. Thus, $M_{\mathcal{G}_{disc}}$ is right-invertible if and only if its minor obtained by removing the v row and the v column is right-invertible. Performance of such removals for all Z labelled vertices results in $M_{\mathcal{G}_X}$, and thus $M_{\mathcal{G}_{disc}}$ is right-invertible if and only if $M_{\mathcal{G}_X}$ is.

$(5 \Leftrightarrow 6)$: by construction of flow matrix, $M_{\mathcal{G}_{disc}} = M_{\mathcal{G}}$ and the equivalence follows.

Thus, $1 \Leftrightarrow 6$, as required. □

The above theorem will be essential in the next subsection. See figure 2b for an example.

3.3 Reduction to MaxRank

Finally, we can transform **FlowSearch** into a special case of **MaxRank** problem.

Definition 3.18. Let (G, I, O) be any open graph, i.e. an input for **FlowSearch**. We define the **variable flow matrix** $M'_{G,I,O}$ as a matrix obtained as follows. (As a reminder $B := V \setminus \{I \cup O\}$.)

1. Start with the reduced adjacency matrix $A_G \mid^{\overline{O}}_{\overline{I}}$.
2. For each $v \in B$, multiply the v row by a variable x_v and the v column by a variable y_v.
3. For each $v \in B$, set the intersection of the v row and the v column to $(1 + x_v)(1 + y_v)$.

For an example, consider figure 2c.

$$\begin{array}{cccc|cccc}
 & C & D & E & F & G & H & \\
A & 1 & 1 & 0 & 0 & 0 & 0 & A \\
B & 1 & 1 & 1 & 0 & 0 & 0 & B \\
\cline{2-7}
C & 0 & 0 & 0 & 0 & 1 & 1 & C \\
D & 0 & 0 & 0 & 0 & 1 & 1 & D \\
E & 0 & 0 & 0 & 1 & 0 & 0 & E \\
 & C & D & E & F & G & H &
\end{array}$$

(a) The reduced adjacency matrix of the graph in 1a. This matrix is not right-invertible, so there is no Pauli flow.

$$\begin{array}{cccc|cccc}
 & C & D & E & F & G & H & \\
A & 1 & 0 & 0 & 0 & 0 & 0 & A \\
B & 1 & 0 & 1 & 0 & 0 & 0 & B \\
\cline{2-7}
C & 0 & 0 & 0 & 0 & 1 & 1 & C \\
D & 0 & 1 & 0 & 0 & 0 & 0 & D \\
E & 0 & 0 & 0 & 1 & 0 & 0 & E \\
 & C & D & E & F & G & H &
\end{array}$$

(b) The flow matrix of the graph in 1b. This matrix is right-invertible, so there is Pauli flow. It differs from the matrix in 2a in vertex D.

$$\begin{array}{cccc|cccc}
 & C & D & E & F & G & H & \\
A & y_C & y_D & 0 & 0 & 0 & 0 & A \\
B & y_C & y_D & y_E & 0 & 0 & 0 & B \\
\cline{2-7}
C & z_C & 0 & 0 & 0 & x_C & x_C & C \\
D & 0 & z_D & 0 & 0 & x_D & x_D & D \\
E & 0 & 0 & z_E & x_E & 0 & 0 & E \\
 & C & D & E & F & G & H &
\end{array}$$

(c) The variable flow matrix of the open graph in 1 (ignoring the labellings). z_v is a shorthand for $(1+x_v)(1+y_v)$ where $v \in \{C,D,E\}$.

Figure 2: Examples of the matrix interpretations of flows for labelled open graphs from 1

Theorem 3.19. The answer to the (G,I,O) instance of **FlowSearch** is $True$ if and only if there exists a valuation to $\{0,1\}$ of all variables in $M'_{G,I,O}$ resulting in a right-invertible matrix, i.e. when the answer to the instance of **MaxRank** given by the pair $(M'_{G,I,O}, |\overline{O}|)$ is "YES".

Proof. By theorem 3.9, we can consider only λ with codomain $\{X,Z\}$ and by theorem 3.10, we can can assume $I \cap O = \emptyset$.

($\Rightarrow$): let $\lambda\colon \overline{O} \longrightarrow \{X,Z\}$ be a measurement labelling for which (G,I,O,λ) has focussed Pauli flow. For $v \in B$, send x_v and y_v to 1 when $\lambda(v) = X$ and to 0 when $\lambda(v) = Z$. Then, under this valuation, $M'_{G,I,O}$ evaluates to $M_{(G,I,O,\lambda)}$ which is right-invertible by theorem 3.17.

($\Leftarrow$): let σ be a valuation sending variables in $M'_{G,I,O}$ to $\{0,1\}$ for which $M'_{G,I,O}$ is right-invertible. Suppose $\sigma(x_v) = 0 \neq 1 = \sigma(y_v)$ for some $v \in B$. Under σ, the v row in $M'_{G,I,O}$ equals 0 everywhere, and the matrix cannot be right-invertible as it does not have maximal row rank, contradiction. Now, suppose $\sigma(x_v) = 1 \neq 0 = \sigma(y_v)$ for some $v \in B$. Under σ, the v column in $M'_{G,I,O}$ equals 0 everywhere. Changing $\sigma(y_v)$ to 1 could change the entries of the v column, otherwise leaving the matrix unchanged. Thus, such a change cannot lower the row rank and $M'_{G,I,O}$ would stay right-invertible. Hence, there exists a valuation σ' resulting in right-invertible matrix with $\sigma'(x_v) = \sigma'(y_v)$ for all $v \in B$. Now, we define $\lambda\colon \overline{O} \longrightarrow \{X,Z\}$ as follows. For $i \in I$: $\lambda(i) = X$. For $v \in B$, set $\lambda(v) = X$ when $\sigma'(x_v) = \sigma'(y_v) = 1$ and $\lambda(v) = Z$ when $\sigma'(x_v) = \sigma'(y_v) = 0$. Then, $M_{(G,I,O,\lambda)}$ equals $M'_{G,I,O}$ under σ'. Thus, $M_{(G,I,O,\lambda)}$ is right-invertible and by theorem 3.17, (G,I,O,λ) has focussed Pauli flow, ending the proof. □

Two variables x_v and y_v for each $v \in B$ might seem unnecessary. Multiplying both the row and the column by x_v and setting the intersection to $1+x_v$ would also work. The problem is that then each variable no longer appears in one column or row only, and theorem 2.7 would not work.

Theorem 3.1 (Repeated)**. FlowSearch** is in RP.

Proof. By theorem 3.19, answering (G,I,O) instance of **FlowSearch** is equivalent to answering instance of **MaxRank** given by $(M'_{G,I,O}, |\overline{O}|)$ over the field $\mathbb{F}_2$. The entries of $M'_{G,I,O}$ are all multi-affine – they are either 0, 1, a variable, a product of two different variables or $(x_v+1)(y_v+1)$ for some $v \in B$. Hence, such matrix instances satisfy all conditions of theorem 2.7, and the problem is in RP. □

An analogous procedure can also determine whether a partial labelling $\lambda\colon \overline{O} \hookrightarrow \{X,Z\}$ can be extended to a full labelling with flow. To do so, rather than using **MaxRank** on $M'_{G,I,O}$, we can consider it on $M'_{G,I,O}$ with some variables evaluated to 1 or 0 depending on λ. It means, that the problem of finding λ such that (G,I,O,λ) has Pauli flow is also solvable in random polynomial time, as captured by the following corollary with initial $\lambda = \emptyset$.

Corollary 3.20. Given an open graph (G,I,O) and a partial measurement labelling λ with codomain $\{X,Z\}$, it is in RP to check if λ can be extended to a full labelling λ' such that (G,I,O,λ') has Pauli flow.

The pseudocodes of the algorithms arising from the above theorems and theorem 2.7 can be found in the appendix C. In the appendix D, we discuss the complexity and the possible implementation.

3.4 Inputs and outputs

Here, we show that given a labelled open graph with Pauli flow, it is always possible to reduce the number of outputs to match the number of inputs while preserving the existence of Pauli flow for some measurement labelling.

Theorem 3.3 (Repeated)**.** Suppose (G,I,O,λ) has Pauli flow and $|O| > |I|$. Then there exists a subset $O' \subseteq O$ such that $|O'| = |I|$ and a labelling λ' such that (G,I,O',λ') has Pauli flow.

Proof. Let $M = M_{(G,I,O,\lambda)}$ be the flow matrix of (G,I,O,λ) with Pauli flow and with $|O| > |I|$. By theorem 3.17, M is right-invertible. Hence, M has $|\overline{O}| \times |\overline{O}|$ invertible minor. Let C be the set of vertices corresponding to the columns that are not in such minor. Suppose that $o \in C \cap O$. The output o can be removed from the graph without breaking the flow existence – M without o column still contains an invertible square minor of maximal size. By lemma 3.15, we can equivalently change o to be Z labelled without breaking flow existence. Now, assume that $C \cap O = \emptyset$. In the flow matrix, the column and row of v with $\lambda(v) = Z$ are 0 except for the intersection. Therefore, the v column must be included in the maximal invertible minor and thus $v \notin C$. Hence, C is a subset of the set of X labelled vertices. Let $v \in C$. We can remove v from the graph, keeping the flow existence: removal of v column does not impact right-invertibility, and removal of v row cannot break right-invertibility either – other rows would remain linearly independent. By lemma 3.15, v can be reintroduced with the same neighbours as previously, but with $\lambda(v) = Z$. The same change can be applied to all vertices in C. Thus, if $|O| > |I|$ it is always possible to decrease the number of outputs or X measured vertices. As the second does not affect $|O| > |I|$, by repeating this process, eventually the number of outputs must decrease and match $|I|$. □

For examples, see figure E.1 in the appendix E. The procedure used in the proof above can be efficiently implemented by utilizing the basis-finding algorithm i.e. Gaussian elimination. The basis-finding procedure has another usage. In the case of X labelling only, suppose that we are given O but not I. We can then find I with $|I| = |O|$ resulting in flow, or determine that no flow exists for any set of inputs.

Lemma 3.21. Let G be a graph and $O \subseteq V$. Let $\lambda\colon \overline{O} \to \{X\}$. Then either (G,I,O,λ) does not have Pauli flow for any I, or (G,I,O,λ) has Pauli flow for some I with $|I| = |O|$.

Proof. Consider $(G,\emptyset,O,\lambda)$. Suppose that it does not have Pauli flow. Then, the reduced adjacency matrix of $(G,\emptyset,O)$ is not right-invertible. Changing inputs to a different set than $\emptyset$ corresponds to the removal of columns but not rows from the reduced adjacency matrix – it cannot make the matrix right-invertible. Hence there is no I for which (G,I,O,λ) has Pauli flow. Conversely, if $(G,\emptyset,O,\lambda)$ has Pauli flow, then we can choose $|\overline{O}| \times |\overline{O}|$ minor from the reduced adjacency matrix. The columns that are not chosen can be removed without breaking the flow, i.e. the corresponding vertices can be changed to be inputs. Note, that some output could be changed to also be an input. This process always turns $|V| - |\overline{O}| = |O|$ vertices into inputs, which ends the proof. □

Finally, again in the case of X labelling only, suppose that we are given inputs. Can we find a minimal (smallest) set of outputs resulting in Pauli flow? The answer is yes.

Lemma 3.22. Let G be a graph and $I \subseteq V$. Let $\lambda(v) = X$ for $v \in V$. Then a minimal O resulting in $(G, I, O, \lambda\mid_{\overline{O}})$ having Pauli flow can be efficiently found.

Proof. Let $M_\emptyset$ be the reduced adjacency matrix of $(G, I, \emptyset, \lambda)$. Let D be a set of rows forming the basis of the space given by all rows. Let C be the set of vertices whose rows are not in D. Then $O = C$ is the required minimal set – clearly $(G, I, O, \lambda\mid_{\overline{O}})$ has Pauli flow – in its reduced adjacency matrix M_O the rows are linearly independent, so the matrix is right-invertible. Also, O is minimal – any smaller set O' of outputs cannot result in a right-invertible reduced adjacency matrix, as such matrix would have more rows than M_O, and all of its rows would be from $M_\emptyset$. But M_O has the maximal number of linearly independent rows, as those rows form a basis of the space spanned by the rows of $M_\emptyset$. Note, that some inputs could be changed to also be outputs. □

4 Conclusions and further work

We have shown that given an open graph it is in RP to determine whether there exists a measurement labelling for which the open graph has Pauli flow. In other words, there is a random polynomial time algorithm deciding whether an open graph state can be used for any type of deterministic computation. To obtain this result, we developed an algebraic interpretation of flow for the case with two Pauli measurements X and Z, and then performed a reduction to a known problem from computational complexity. We have also shown that the algebraic interpretation can be useful when looking for the necessary properties that a set of inputs or outputs must satisfy. In particular, we showed that it is always possible to reduce the number of outputs to match the number of inputs when it is allowed to change some measurement labels to Z. Our results contribute to the general picture of what the Pauli flow structure is and which open graphs can exhibit it.

Sometimes a graph state can be prepared, but it might be difficult to check if such a state can be useful in MBQC. A possible approach can be checking other states in orbit [1]. Our result can be interpreted as answering whether an open graph can be used in any form of quantum computation, or to give some conditions on such computation that are necessary to achieve deterministic labelled open graphs.

The remainder is a discussion of possible future work.

Polynomial time When showing that some problem, **FlowSearch** in our case, is in RP, it is natural to ask whether a problem is also in P. It would be interesting to modify reduction to **MaxRank** to obtain instances for which polynomial time algorithms exist, like the instances in [18].

Other measurements The main trick we have used to show **FlowSearch** $\in$ RP, was the restriction of possible measurements to Pauli X and Z measurements for which we developed algebraic interpretation. A similar interpretation of XY planar measurements is known [25] and may be extended to allow Pauli Z measurements, but the structure is harder to work with due to the partial order requirements. Extending our results to also work for XY and Z measurements is an interesting direction for further research – X and Z measurements result in the Clifford fragment which can be classically simulated [15], while XY planar measurements are sufficient for universality even on cluster states [21]. It would also be interesting to check if our approach can be used to minimise the number of vertices that need to be Z measured to have flow. The main problem with adapting the presented methods to finding labels with planar measurements is the order. Without verifying the order (ignoring conditions (P1), (P2), (P3)), a reduction to the **MaxRank** problem should still be possible. However, order changes everything. There might be a flow where a particular vertex has a label X or Z. Yet, measuring such vertex in any plane can break the flow.

Altering the set of edges Another problem that can be interpreted as **MaxRank** instance is finding how to change the set of edges to get an open graph with flow. For instance, when all non-outputs are X labelled, we can put a new variable in the place of 0 entries corresponding to the lack of an edge between vertices. An interesting question would be to determine how many edges must be added to get flow or how many must be removed. If such a problem is tractable, it could have practical usage, where given an open graph state one could say how far the state is from one allowing deterministic computation. The number of edges that need to be flipped corresponds to the number of CZ gates that must be applied.

Circuit extraction Our results also connect to the ZX calculus, where Pauli flow is a necessary condition for efficient circuit extraction [34]. It would be interesting to expand on this connection, for instance, by attempting the classification of small open graphs, that, when contained in ZX diagram, are guaranteed to break flow, and thus should be avoided in any form of optimization utilizing ZX. Another connection could be circuit extraction from phase-free ZH – X and Z measurements are sufficient for universality in MBQC on hypergraph states [36]. There, the errors can be multi-qubit byproducts and their fixing is done by switching measurement bases. Our method of checking the existence of measurement labelling resulting in Pauli flow could be helpful when looking for measurements that can be swapped without removing the property of determinism (i.e. Pauli flow) in the subparts without any hyperedges.

Acknowledgement

I thank Miriam Backens, Tommy McElvanney, and Korbinian Staudacher for helpful discussions. Special thanks go to my supervisor Miriam Backens, for sharing theorem 3.12 with me. I also thank anonymous reviewers for their useful comments.

References

[1] Jeremy C. Adcock, Sam Morley-Short, Axel Dahlberg & Joshua W. Silverstone (2020): *Mapping Graph State Orbits under Local Complementation*. *Quantum* 4, p. 305, doi:10.22331/q-2020-08-07-305. arXiv:1910.03969.

[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] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich & Javier Verbel (2020): *Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems*. In Shiho Moriai & Huaxiong Wang, editors: *Advances in Cryptology – ASIACRYPT* 2020, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 507–536, doi:10.1007/978-3-030-64837-4_17.

[4] Anne Broadbent, Joseph Fitzsimons & Elham Kashefi (2009): *Universal Blind Quantum Computation*. In: 2009 50*th Annual IEEE Symposium on Foundations of Computer Science*, pp. 517–526, doi:10.1109/FOCS.2009.36.

[5] 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.

[6] Jonathan F. Buss, Gudmund S. Frandsen & Jeffrey O. Shallit (1999): *The Computational Complexity of Some Problems of Linear Algebra*. *Journal of Computer and System Sciences* 58(3), pp. 572–596, doi:10.1006/jcss.1998.1608.

[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] Vincent Danos & Elham Kashefi (2006): *Determinism in the One-Way Model*. *Physical Review A* 74(5), p. 052310, doi:10.1103/PhysRevA.74.052310.

[9] Niel de Beaudrap (2007): *A Complete Algorithm to Find Flows in the One-Way Measurement Model*, doi:10.48550/arXiv.quant-ph/0603072. arXiv:quant-ph/0603072.

[10] 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.

[11] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-Diagrams Can Be* #P-*Hard*. In: 49*th International Colloquium on Automata, Languages, and Programming* (*ICALP* 2022), *LIPIcs* 229, pp. 119:1–119:19, doi:10.4230/LIPIcs.ICALP.2022.119.

[12] Richard A. Demillo & Richard J. Lipton (1978): *A Probabilistic Remark on Algebraic Program Testing*. *Information Processing Letters* 7(4), pp. 193–195, doi:10.1016/0020-0190(78)90067-4.

[13] 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.

[14] Sergey B. Gashkov & Igor S. Sergeev (2013): *Complexity of Computation in Finite Fields*. *Journal of Mathematical Sciences* 191(5), pp. 661–685, doi:10.1007/s10958-013-1350-5.

[15] Daniel Gottesman (1998): *The Heisenberg Representation of Quantum Computers*, doi:10.48550/arXiv.quant-ph/9807006. arXiv:quant-ph/9807006.

[16] Nicholas J. A. Harvey, David R. Karger & Sergey Yekhanin (2006): *The Complexity of Matrix Completion*. In: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm - SODA '06*, ACM Press, Miami, Florida, pp. 1103–1111, doi:10.1145/1109557.1109679.

[17] Matt Hostetter (2020): *Galois: A performant NumPy extension for Galois fields*. https://github.com/mhostetter/galois. (accessed February 2024).

[18] Gábor Ivanyos, Marek Karpinski & Nitin Saxena (2010): *Deterministic Polynomial Time Algorithms for Matrix Completion Problems*. *SIAM Journal on Computing*, doi:10.1137/090781231.

[19] Frank Luebeck (2021): *Conway Polynomials for Finite Fields*. https://www.math.rwth-aachen.de/~Frank.Luebeck/data/ConwayPol/index.html. (accessed February 2024).

[20] Meena Mahajan & Jayalal M. N. Sarma (2010): *On the Complexity of Matrix Rank and Rigidity*. *Theory of Computing Systems* 46(1), pp. 9–26, doi:10.1007/s00224-008-9136-8.

[21] Atul Mantri, Tommaso F. Demarie & Joseph F. Fitzsimons (2017): *Universality of Quantum Computation with Cluster States and (X, Y)-Plane Measurements*. *Scientific Reports* 7(1), p. 42861, doi:10.1038/srep42861.

[22] Damian Markham & Elham Kashefi (2014): *Entanglement, Flow and Classical Simulatability in Measurement Based Quantum Computation*. In Franck van Breugel, Elham Kashefi, Catuscia Palamidessi & Jan Rutten, editors: *Horizons of the Mind. A Tribute to Prakash Panangaden*: *Essays Dedicated to Prakash Panangaden on the Occasion of His* 60*th Birthday*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 427–453, doi:10.1007/978-3-319-06880-0_22.

[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?* In Dave Bacon, Miguel Martin-Delgado & Martin Roetteler, editors: *Theory of Quantum Computation, Communication, and Cryptography*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 174–187, doi:10.1007/978-3-642-54429-3_12.

[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 (2022): *Characterising Determinism in MBQCs Involving Pauli Measurements*, doi:10.48550/arXiv.2207.09368. arXiv:2207.09368.

[28] Piotr Mitosek & Miriam Backens: *Unpublished Upcoming Paper*.

[29] Øystein Ore (1921): *Über Höhere Kongruenzen*. Norsk Matematisk Forenings Skrifter, Grøndahl.

[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] Robert Raussendorf, Daniel E. Browne & Hans J. Briegel (2002): *The One-Way Quantum Computer - a Non-Network Model of Quantum Computation*. *Journal of Modern Optics* 49(8), pp. 1299–1306, doi:10.1080/09500340110107487. arXiv:quant-ph/0108118.

[32] Robert Raussendorf, Daniel E. Browne & Hans J. Briegel (2003): *Measurement-Based Quantum Computation on Cluster States*. *Physical Review A* 68(2), p. 022312, doi:10.1103/PhysRevA.68.022312.

[33] Jacob T. Schwartz (1980): *Fast Probabilistic Algorithms for Verification of Polynomial Identities*. *Journal of the ACM* 27(4), pp. 701–717, doi:10.1145/322217.322225.

[34] 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.

[35] 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.

[36] Yuki Takeuchi, Tomoyuki Morimae & Masahito Hayashi (2019): *Quantum Computational Universality of Hypergraph States with Pauli-X and Z Basis Measurements*. *Scientific Reports* 9(1), p. 13585, doi:10.1038/s41598-019-49968-3.

[37] John van de Wetering (2020): *ZX-calculus for the Working Quantum Computer Scientist*, doi:10.48550/arXiv.2012.13966. arXiv:2012.13966.

[38] Richard Zippel (1979): *Probabilistic Algorithms for Sparse Polynomials*. In Edward W. Ng, editor: *Symbolic and Algebraic Computation*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 216–226, doi:10.1007/3-540-09519-5_73.

A More about MaxRank

In order to prove theorem 2.7, we need the following lemma.

Lemma A.1. A multi-affine polynomial is 0 over a finite field $\mathbb{F}_s$ if and only if it is 0 over $\mathbb{F}_{s^k}$ for any $k \in \mathbb{Z}_+$.

Proof. This follows from [6, Lemma 25 and Corollary 26]. □

Because of the above, instead of testing whether a multi-affine polynomial is 0 over $\mathbb{F}_s$, we can test whether it is 0 over some large field extension. In particular, the extension can be taken sufficiently large to ensure that the Schwartz-Zippel lemma [29, 33, 12, 38] applies (adapted to $\mathbb{F}_{s^k}$ only):

Theorem A.2. Let $P \in \mathbb{F}_{s^k}[x_1,\dots,x_n]$ be a non-zero polynomial of total degree d over $\mathbb{F}_{s^k}$. Let $a_1,\dots,a_n \in \mathbb{F}_{s^k}$ be chosen at random uniformly. Then:

$$\Pr[P(a_1,\dots,a_n) = 0] \leq \frac{d}{s^k}.$$

We can now prove theorem 2.7.

Proof of theorem 2.7. Let M, r be a matrix and an integer forming an input to **MaxRank** satisfying the conditions from the theorem statement. If $r > \min(m,n)$ or $r < 0$, the answer is "NO" and can be returned immediately, so assume $0 \leq r \leq \min(m,n)$. Consider any $r \times r$ minor M' of M. We show $\det M'$ is multi-affine: consider any variable x in M'. Then, x appears in at most one row or one column of M'. By performing Laplace expansion on M' in such row (column), we find that $\det M'$ is a sum of determinants of $(r-1)\times(r-1)$ minors of M' that do not contain x multiplied by elements of the row (column) used for the expansion that itself may contain x only in degree 1 due to multi-affinity assumption about matrix entries. Therefore, x appears in degree at most 1 in $\det M'$ and the same for all other variables of $\det M'$, so the determinant is multi-affine. Therefore, the method from [6, Theorem 28] applies. We present a modified version for clarity.

Consider the following procedure. Let $p \leq \frac{1}{2}$ be the desired error probability. It is sufficient to consider $p = \frac{1}{2}$, but we present also how to achieve arbitrarily small error probability. Given M, r, p, let k be such that $\mathbb{F}_{s^k}$ has at least $\frac{t}{p}$ elements, i.e. $k = \left\lceil \log_s \frac{t}{p} \right\rceil$. Let $a_1,\dots,a_t$ be a randomly chosen valuation of $x_1,\dots,x_t$ from $\mathbb{F}_{s^k}$. Let $r_a = \operatorname{rank} M(a_1,\dots,a_t)$, which can be found by Gaussian elimination i.e. in polynomial time. The computations over finite field are possible in time polynomial in $\log_2 s$ and k [14]. Return "YES" if and only if $r_a \geq r$. We show, that the following procedure shows RP containment.

Suppose, that the actual answer to the instance is "YES". Then, under some valuation, M has rank at least r. Under such valuation, M must have $r \times r$ reversible minor. Let M' be such minor. By the previous part, $\det M'$ is multi-affine. Let d be the total degree of $\det M'$. Then, $d \leq t$ again by multi-affinity. By lemma A.1, M' is 0 over $\mathbb{F}_s$ if and only if it is 0 over $\mathbb{F}_{s^k}$. Combining everything with the Schwartz-Zippel theorem A.2, we get that:

$$\Pr[\det M'(a_1,\dots,a_t) =_{\mathbb{F}_{s^k}} 0] \leq \frac{d}{s^k} \leq \frac{t}{s^k} \leq \frac{t}{t/p} = p$$

The same holds for all $r \times r$ minors of M that are invertible under some valuation. Hence, with error probability at most p, a random valuation from $\mathbb{F}_{s^k}$ results in a non-root of some $r \times r$ minor's determinant of M in which case rank of the minor under such valuation is r and so $\operatorname{rank} M$ is at least r, i.e. the procedure above would find $r_a \geq r$ and return "YES". Hence, the procedure described above returns the correct answer with probability at least $1 - p \geq \frac{1}{2}$.

Now suppose, that the answer to the instance is "NO". Then, all $r \times r$ minors of M must have determinants equal 0 over $\mathbb{F}_s$. By multi-affinity, they are also equal 0 over $\mathbb{F}_{s^k}$. Hence, a random valuation $a_1,\dots,a_t$ always results in $\operatorname{rank} M(a_1,\dots,a_t) \leq k$. Hence, the procedure described above returns "NO" with probability 1, ending the proof of containment in RP. □

Example A.3. Consider the following matrix over $\mathbb{F}_2$:

$$M = \begin{pmatrix} x_1 & 0 & 1 & 0 \\ 0 & x_2x_3 & 0 & 0 \\ 1 & 0 & x_1 & 0 \\ x_1 & x_2 & x_3 & 0 \end{pmatrix}$$

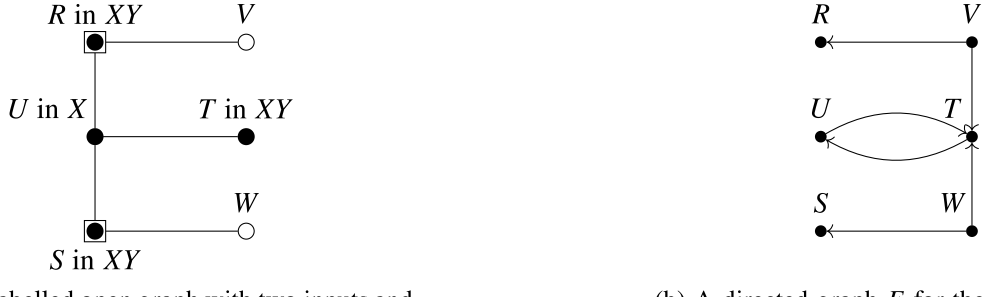
Setting $x_1 = 0$ and $x_2, x_3 = 1$ results in M having rank 3. However, there is no valuation resulting in the matrix having rank 4, as the fourth column contains 0s only. Hence, the answer to instances $(M,1),(M,2),(M,3)$ of **MaxRank** is "YES", but the answer to $(M,4)$ is "NO".

Some closely related problems also defined in [6] include **MinRank**, **Sing**, and **NonSing**. **MinRank** takes the same inputs as **MaxRank** and asks whether a rank $\leq r$ can be achieved. **Sing** and **NonSing** take a square matrix and ask whether the matrix can be made singular and non-singular respectively.

In general, these problems are hard or sometimes unsolvable. For instance, **MinRank** is undecidable when $R = \mathbb{Z}, E = S = \{0,1\}$ [6]. The problems are very natural and often appear when working on any linear algebra problems. **Sing** is useful in cryptography due to its hardness (for example, see [3]). It is also interesting from a complexity perspective (for example, see [20]).

We only work with **MaxRank** over finite fields (in fact, later we only consider $\mathbb{F}_{2^k}$). A variant where each variable can appear at most once is in P. A slightly more general version where a variable can appear in at most one row or one column but an unlimited number of times is also in P [18]. Note, that this result is not stronger than the presented theorem 2.7, as the entries of the matrix there cannot include products of variables. When each variable can appear at most twice in the matrix, but not necessarily in one row or one column, the problem already becomes NP-complete [16].

B Reduced adjacency matrix invertibility for X and XY measurements



(a) A labelled open graph with two inputs and two outputs containing Pauli flow. Vertex U is X labelled and all other non-outputs are XY labelled.

(b) A directed graph F for the labelled open graph from B.1a, obtained from theorem 3.12. It contains a cycle. The corresponding F' contains only one edge TU and is acyclic.

Figure B.1: An example of a labelled open graph with both X and XY measurements and explanation of theorem 3.12 acting on it.

$$\begin{array}{cccccc} & T & U & V & W & \\ R & \left(\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array}\right) & & & & \begin{array}{c} R \\ S \\ T \\ U \end{array} \\ & T & U & V & W & \end{array}$$

(c) Reduced adjacency matrix of the labelled open graph from B.1a.

$$\begin{array}{cccccc} & R & S & T & U & \\ \begin{array}{c} T \\ U \\ V \\ W \end{array} & \left(\begin{array}{cc|cc} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{array}\right) & & & & \begin{array}{c} T \\ U \\ V \\ W \end{array} \\ & R & S & T & U & \end{array}$$

(d) The inverse of matrix in B.1c, i.e. the matrix $A_F\,|^{\overline{I}}_{O}$ where F is as in B.1b.

Figure B.1 (continued): An example of a labelled open graph with both X and XY measurements and explanation of theorem 3.12 acting on it.

C Pseudocode for algorithms

In the pseudocode below, we do not explicitly construct an instance of matrix used for **MaxRank** problem, as, in practice, it might be difficult to explicitly construct a matrix with variables that can be substituted with values from large field extensions of $\mathbb{F}_2$. Instead, we only construct the matrix under some valuation.

Checks if a partial X, Z labelling can be extended so that (G, I, O, λ) has Pauli flow. The error probability must be bounded above by p.

```
procedure FlowSearchAux($G, I, O, \lambda, p$)
    $M \leftarrow A_G\,|^{\overline{O}}_{\overline{I}}$        ▷ Construction of reduced adjacency matrix
    $Vars \leftarrow \emptyset$        ▷ Initialize set of variables
    for $v \in B$        ▷ Detecting unlabelled vertices
        if $\lambda(v)$ is defined
            if $\lambda(v) == Z$        ▷ Updating the row and the column of $Z$ labelled vertex
                multiply $v$ row of $M$ by 0
                multiply $v$ column of $M$ by 0
                set the intersection of $v$ row and $v$ column of $M$ to 1
        else
            $Vars \leftarrow Vars \cup \{x_v, y_v\}$
    $k \leftarrow \left\lceil \log_2 \frac{|Vars|}{p} \right\rceil$        ▷ Minimal $k$ such that $\mathbb{F}_{2^k}$ has at least $\frac{|Vars|}{p}$ elements and the error probability is below $p$.
    Randomly sample $\sigma : Vars \longrightarrow \mathbb{F}_{2^k}$
    for $v \in B$        ▷ Construction of $M'_{G,I,O}$ under valuation $\sigma$, computations are done in $\mathbb{F}_{2^k}$
        multiply $v$ row of $M$ by $\sigma(x_v)$
        multiply $v$ column of $M$ by $\sigma(y_v)$
        set the intersection of $v$ row and $v$ column to $(\sigma(x_v)+1)(\sigma(y_v)+1)$
    Gaussian eliminate $M$
    return (rowrank $M == |\overline{O}|$)
```

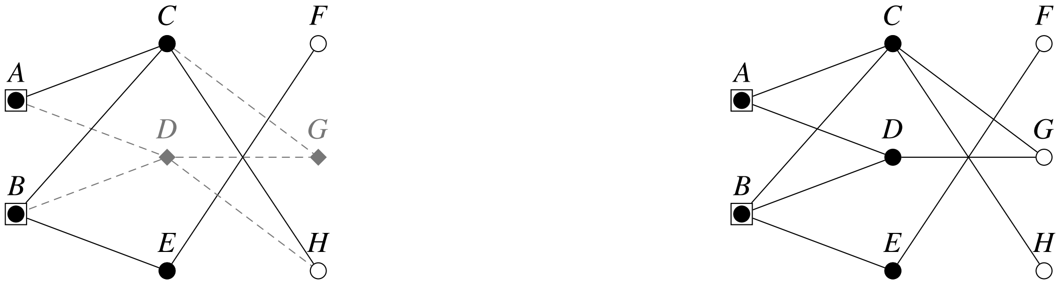
Main algorithm, returns $True$ if (G, I, O, λ) has Pauli flow for some λ. The error probability must

```
be bounded above by $p$.
procedure FlowSearch$(G, I, O, p)$
    return FlowSearchAux$(G, I, O, \emptyset, p)$

Finds $\lambda$ such that $(G, I, O, \lambda)$ has Pauli flow. Initially, checks whether any such $\lambda$ exists, up to error
probability $p$.
procedure FindLabelling$(G, I, O, p)$
    if not FlowSearch$(G, I, O, p)$
        return “NO $\lambda$ EXISTS”
    $\lambda \leftarrow \emptyset$                                         ▹ Initialization of $\lambda$
    for $v \in I$
        $\lambda(v) \leftarrow X$                                      ▹ Inputs must be $X$ labelled
    for $v \in B$
        $confirm_v \leftarrow False$
        $current_v \leftarrow X$                                  ▹ Initially attempt $X$ label
        while not $confirm_v$                      ▹ Alternate $X$ and $Z$ labels until one works
            $\lambda(v) \leftarrow current_v$
            if FlowSearchAux$(G, I, O, \lambda, p)$    ▹ Note, that the error probability could be increased
                                                  at the cost of possibly more tries being required.
                $confirm_v \leftarrow True$
            else
                if $current_v == X$
                    $current_v \leftarrow Z$
                else
                    $current_v \leftarrow X$
    return $\lambda$
```

D Complexity of algorithms

The most memory-expensive part of the algorithms is the creation of $M'_{G,I,O}$ under some valuation from $\mathbb{F}_{2^k}$ where k depends on the desired error probability p and the size of the input graph. Since $k = \left\lceil \log_2 \frac{|Vars|}{p} \right\rceil$ and $|Vars| \leq 2 \cdot |B|$, we get that $k \in O\left(\log_2 \frac{|B|}{p}\right)$. The elements of such fields can be represented using $O(k)$ long vectors over $\mathbb{F}_2$ with the time complexity of basic arithmetic operations on such field bounded above by $O(k^2)$ [14]. Therefore, the memory requirement can be bounded above by $O\left(|\overline{O}| \times |\overline{I}| \times \log_2 \frac{|B|}{p}\right) \in O\left(n^2 \log_2 \frac{n}{p}\right)$ where $n = |V|$. The most time-expensive part of the algorithms are the Gaussian eliminations. Each Gaussian elimination requires $O\left(n^3\right)$ basic operations in $\mathbb{F}_{2^k}$. Thus, a (not very efficient) upper bound for the time complexity is $O\left(n^3 \log_2^2 \frac{n}{p}\right)$ for the decision variant and expected $O\left(n^4 \log_2^2 \frac{n}{p}\right)$ for the actual finding of the labelling resulting in a Pauli flow. Many programming languages offer packages for efficient computation in finite fields. For instance, in Python one can use Galois package [17] which works very well for $\mathbb{F}_{2^k}$ with k such that the precomputed Conway polynomial [19] is known, for example, all $1 \leq k \leq 91$. Such values of k are sufficient for all reasonable computations, as the error can be dropped below $\frac{1}{2^{50}}$. At that point, it is more likely for a random cosmic beam to corrupt the computation than to get an error due to the probabilistic nature of the algorithms.

E Figure for reducing the number of outputs



(a) The open graph from 1b with flow matrix from 2b. Columns given by vertices C, D, E, F and H form the basis. Thus, the output G can be removed (equivalently: changed to be Z labelled).

(b) An example of a labelled open graph with Pauli flow in which no output can be immediately changed to be Z measured. However, changing D to be Z measured makes it possible to also change output G to be Z measured.

$$\begin{array}{c|ccc|ccc|c}
 & C & D & E & F & G & H & \\
A & 1 & 1 & 0 & 0 & 0 & 0 & A \\
B & 1 & 1 & 1 & 0 & 0 & 0 & B \\
\hline
C & 0 & 0 & 0 & 0 & 1 & 1 & C \\
D & 0 & 0 & 0 & 0 & 1 & 0 & D \\
E & 0 & 0 & 0 & 1 & 0 & 0 & E \\
 & C & D & E & F & G & H &
\end{array}$$

(c) Flow matrix of the open graph in E.1b. Columns given by vertices C, E, F, G, H form the basis. Thus, the vertex D can be changed to be Z labelled.

$$\begin{array}{c|ccc|ccc|c}
 & C & D & E & F & G & H & \\
A & 1 & 0 & 0 & 0 & 0 & 0 & A \\
B & 1 & 0 & 1 & 0 & 0 & 0 & B \\
\hline
C & 0 & 0 & 0 & 0 & 1 & 1 & C \\
D & 0 & 1 & 0 & 0 & 0 & 0 & D \\
E & 0 & 0 & 0 & 1 & 0 & 0 & E \\
 & C & D & E & F & G & H &
\end{array}$$

(d) The flow matrix after switching vertex D from E.1b to be Z labelled. Columns given by vertices C, D, E, F, H form a basis, so output G can be removed (changed to be Z labelled).

Figure E.1: Examples of labelled open graphs with more outputs than inputs, and how the number of outputs can be reduced to match the number of inputs.

A Graphical #SAT Algorithm for Formulae with Small Clause Density

Tuomas Laakkonen[1,a], Konstantinos Meichanetzidis[1,b], John van de Wetering [2,c]
[1] Quantinuum, 17 Beaumont Street, Oxford OX1 2NA, United Kingdom
[2] Informatics Institute, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
[a,b] {tuomas.laakkonen, k.mei}@quantinuum.com, [c] john@vdwetering.name

We study the counting version of the Boolean satisfiability problem **#SAT** using the ZH-calculus, a graphical language originally introduced to reason about quantum circuits. Using this, we generalize **#SAT** to a weighted variant we call **#SAT**$_\pm$, which is complete for the class GapP. We show there is an efficient linear-time reduction from **#SAT** to **#2SAT**$_\pm$, unlike previous reductions from **#SAT** to **#2SAT** which blow up the size of the formula by a polynomial factor. Our main conceptual contribution is that introducing weights to **#SAT** allows for more efficient translations, and we use this to remove the dependence on clause width k in this case. We observe that DPLL-style algorithms for **#2SAT** can be adapted to **#2SAT**$_\pm$ directly and hence the best-known upper bounds for **#2SAT** apply. Applying an upper bound for **#2SAT** in terms of variables gives us upper bounds for **#SAT** in terms of clauses and variables that are better than $O^*(2^n)$ for small clause densities of $\frac{m}{n} < 2.25$, and improve on previous average-case and worst-case bounds for $k \geq 6$ and $k \geq 4$, respectively. Applying a similar bound in terms of clauses produces a bound of $O^*(1.1740^L)$ in terms of the length of the formula. These are, to our knowledge, the first non-trivial upper bounds for **#SAT** that is independent of clause size, and in terms of formula length, respectively. Based on a result of Kutzkov, we find an improved bound on **#3SAT** for $1.2577 < \frac{m}{n} \leq \frac{7}{3}$. Finally, we use this technique to find an upper bound on the complexity of calculating amplitudes of quantum circuits in terms of the total number of gates. Our results demonstrate that graphical reasoning can lead to new algorithmic insights, even outside the domain of quantum computing that the calculus was intended for.

1 Introduction

A graphical calculus is a language consisting of diagrams that can be transformed according to specific graphical rewrite rules. Usually these diagrams correspond to some underlying mathematical object that would be hard to reason about directly—like a matrix, tensor, relation or some combinatorial object—and the rewrite rules preserve the semantics of this interpretation. There are for instance graphical calculi for linear algebra [9, 11, 13, 57], for studying concurrency [10, 12], and for finite-state automata [41].

Most relevant for this paper are the graphical calculi developed for studying quantum computing. The *ZX-calculus* [15, 16] can represent arbitrary linear maps between any number of qubits, and has different versions of rewrite rules that are *complete* (meaning the rules can prove any true equality) for various relevant fragments of quantum computing [2, 29, 31, 47]. It has seen use in a variety of areas like optimizing quantum computations [5, 6, 21, 30, 32], more effectively classically simulating quantum computations [33, 34], and several others like [14, 26, 45]; see [54] for a review.

There are a number of variations on the ZX-calculus that include different or additional generators [28, 48, 53]. The one we will use is the *ZH-calculus* [3, 4]. The ZH-calculus has turned out to be useful in a variety of areas [22,49,50], but in particular it has been shown to naturally encode Boolean satisfiability and counting problems [19]. We will build on this representation to show that this perspective leads to better algorithms for formulae that have a low number of clauses.

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 137–152, doi:10.4204/EPTCS.406.7

The Boolean satisfiability problem (**SAT**) is to determine whether a given Boolean formula has a satisfying assignment of variables, and is a canonical example of an **NP**-complete problem. Worst-case upper bounds for solving **SAT** instances can be phrased in terms of some relevant parameters for a Boolean formula $f : \{0,1\}^n \to \{0,1\}$ in conjunctive normal form: its number of variables n, the maximum clause size k, the number of clauses m, the clause density $\delta := \frac{m}{n}$, the number of literals L, and the maximal number of clauses a variable participates in. For instance, for fixed k, there are bounds $O^*(c_k^n)$ where $c_k < 2$ [20]. Yamamoto [56] showed that **SAT** can be solved in $O^*(2^{0.3033m})$ time (independent of k) . This hence implies a better than $O^*(2^n)$ runtime for clause densities $\delta < 3.297$, regardless of k. While for large k and large δ all known bounds converge to $O^*(2^n)$, for small k and arbitrary δ or vice-versa, better bounds are possible.

In this paper we will study the problem **#SAT**, which asks *how many* solutions a Boolean formula has. Hence, this is not a decision problem, but a counting problem. It is complete for the complexity class **#P**, which is the 'counting version' of **NP**. **#SAT** is believed to be a significantly harder problem than **SAT**. For instance, the entire polynomial hierarchy is contained in $\mathbf{P}^{\mathbf{\#SAT}}$ [44]. As it is **#P**-complete, it has applications in a variety of areas. For instance, tensor-network contraction (when suitably formalized) is in **#P** [17]. **#SAT** also has applications in the field of artificial intelligence, where it is usually referred to as *model counting* [7, Chapter 20]. Note that while for **SAT** the problem only becomes hard for $k \geq 3$, for **#SAT**, the problem is already hard for $k = 2$, as **#2SAT** is **#P**-complete [46]. In fact, previously in [37], we used the ZH-calculus to provide a proof of this and other reductions between related counting problems. In this work, we make use of similar techniques in a different direction to give improved upper bounds for **#SAT**.

We can phrase the known upper bounds to **#SAT** in terms of n, k, m and δ, although for **#SAT** much less is known. Similarly to **SAT**, good bounds are known for small k — $O^*(1.2377^n)$ for $k = 2$ [51], and $O^*(1.6423^n)$ for $k = 3$ [35]. There are also bounds known for arbitrary (but fixed) k in the worst-case setting [20] and the average-case setting [55]. Unlike **SAT**, bounds in terms of m are not known for **#SAT** independent of k, but only for $k = 2$, $O^*(1.1740^m)$ [52], and $k = 3$, $O^*(1.4142^m)$ [58].

In this paper we establish, to the best of our knowledge, the first algorithm for **#SAT** that is better than brute-force for low clause density, independent of the clause size k. Specifically, we prove the following theorem:

Theorem 1. *Given a CNF formula $\phi : \{0,1\}^n \to \{0,1\}$, we can count the number of satisfying assignments*

$$\#(\phi) = \#\{\vec{x} \in \{0,1\}^n \,|\, \phi(\vec{x}) = 1\}$$

in time $O^*(1.2377^{n+m_{\geq 3}})$, where $m_{\geq 3}$ is the number of clauses of width at least three. In particular, for clause density $\delta < 2.2503$, this gives a better than $O^*(2^n)$ bound, independent of maximal clause size k.

The worst-case bound for our algorithm improves the best-known bound for a variety of different parameters. We summarize this in Tables 1 and 2. Note that these tables also contain our results based on a more fine-grained analysis of **#3SAT** that is presented in Section 4.2, as well as a bound on **#SAT** in terms of literals L that is presented in Section 4.1. Furthermore, assuming the strong exponential time hypothesis, our results indicate that the 'hardest' density of **#SAT** must be some $\delta > 2.2503$. As far as we aware this is the first known bound on where the hardest clause density of **#SAT** lies.

While the bound of Theorem 1 is only effective at low densities, this is sufficient for many real-world use cases – in particular, for the unweighted **#SAT** instances from the Model Counting Competition 2020 [24], we found that we improve on the previous best worst-case bound for 88% of instances, and the average-case bound for 45% of instances, assuming the constant factors of both algorithms are equal (we take k to be the 90th percentile of clause widths to avoid few large clauses biasing the result in our

Problem	Previous Best	New Bound	Relevant Region	Algorithm
#kSAT for $k > 3$	$O^*(c_k^n)$, $c_k \to 2$ [20]	$O^*(1.2377^{n+m})$	$\delta < 2.2503$ (as $k \to \infty$)	[51]
#kSAT for $k > 3$	$O^*(c_k^n)$, $c_k \to 2$ [20]	$O^*(1.1740^L)$	$\frac{L}{n} < 4.3209$ (as $k \to \infty$)	[52]
#3SAT	$O^*(1.6423^n)$ [35]	$O^*(1.6350^n)$	$1.2577 < \delta \leq \frac{7}{3}$	[51] [35]

Table 1: The different bounds obtained in this paper, along with the corresponding best previous bounds, and the underlying algorithm on which they are based. The given relevant regions are the intervals of formula parameters where our bounds are valid and improve on the previous bound.

Improvement on	k	2	3	4	5	6	7	8	9
Average-Case	δ	–	–	–	–	–	< 0.968	< 1.106	< 1.207
Worst-Case	δ	–	$< 2.333^*$	< 2.077	< 2.170	< 2.212	< 2.231	< 2.241	< 2.246

Table 2: The maximum densities δ at which our upper bound improves on other bounds, as dependent on k. We include average [55] and worst-case bounds [20], both of which converge to $\delta = 2.2503$ as k increases. Entries marked with '–' are where our bound is always worse. The entry marked '$*$' uses the alternate bound presented in Section 4.2 and only improves on the previous best when also $\delta > 1.2577$.

favor). However, it is important to note that practical **#SAT** solvers perform much better than predicted by any of these upper bounds.

Our results are based on two observations. The first observation is that the standard CDP (Counting Davis-Putnam) algorithm [8] for solving **#SAT** can actually solve a more general problem that we dub **#SAT**$_\pm$. In this problem, variables x_i are labeled by a $\phi_i = \pm 1$ phase that determines whether a solution to f should be added or subtracted to the total, as determined by $\prod_{\vec{x}} \phi_i^{x_i}$. The second observation is that we can translate an arbitrary **#SAT** instance into a **#2SAT**$_\pm$ instance. This removes the dependence on maximal clause size k from our problem, and means we can use the known upper bounds to **#2SAT** that apply directly to the problem **#2SAT**$_\pm$ as well.

We found this last observation by writing a **#SAT** instance as a ZH-diagram as described by de Beaudrap *et al.* [19]. We extend their methods by relaxing the conditions on the types of diagrams we consider, which shows that ZH-diagrams also naturally represent **#SAT**$_\pm$ instances. The translation from a **#SAT** instance into a **#2SAT**$_\pm$ instance then follows from a known rewrite rule of the related ZXΔ-calculus [48]. We also find that **#SAT**$_\pm$ is in fact complete for the complexity class **GapP** [23], which is **#P** closed under negation.

In Section 2 we recall the definition of the ZH-calculus, how to encode **#SAT** instances as ZH-diagrams, and how the CDP algorithm for solving **#SAT** works. Then in Section 3 we present our main results: we show how to interpret CDP graphically inside the ZH-calculus, and we find a graphical reduction from **#SAT** to **#2SAT**$_\pm$. We end with a complexity analysis of combining this reduction with the CDP algorithm, modified to work for **#2SAT**$_\pm$. In Section 4 we study some variations on our algorithm: Section 4.1 presents a new bound of $O^*(1.1740^L)$ for **#SAT** that is in terms of number of literals; Section 4.2 gives a modified algorithm for **#3SAT** that is better for certain densities; and Section 4.3 presents the problem of **#SAT** where variables are labeled by arbitrary complex numbers, which we conjecture might be helpful for future improvements. In particular, in Appendix B we apply this technique to calculating amplitudes of quantum circuits, and show an upper bound in terms of the number of gates. We end with some concluding remarks in Section 5.

2 Preliminaries

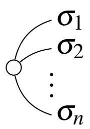
We say that a Boolean formula $\phi : \mathbb{B}^n \to \mathbb{B}$ is in conjunctive normal form if we have

$$\phi(x_1,\dots,x_n) = \bigwedge_{i=1}^{m} (c_{i1} \vee c_{i2} \vee \dots \vee c_{ik_i}) \tag{1}$$

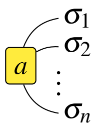
where each c_{ij} is x_l or $\neg x_l$ for some l. We say that ϕ has n variables, m clauses, maximum clause width $k = \max_i\{k_i\}$, density $\delta = \frac{m}{n}$ and number of literals $L = \sum_i k_i$.

2.1 The ZH-calculus

We use the ZH-calculus, a graphical language designed for reasoning about quantum computations [3]. Here we only recall the definition of the generators; see [54, Section 8] for a more detailed overview. The calculus is defined by rewrites on ZH-diagram, which are composed of two generators, called the Z-spider and the H-box. These are given by



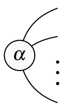
$$= \quad \delta_{\sigma_1\sigma_2\dots\sigma_n} \quad = \quad \begin{cases} 1 & \sigma_1 = \sigma_2 = \dots = \sigma_n \\ 0 & \text{otherwise} \end{cases}$$



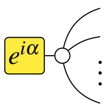
$$= \quad 1 + (a-1)\delta_{1\sigma_1\sigma_2\dots\sigma_n} \quad = \quad \begin{cases} a & 1 = \sigma_1 = \sigma_2 = \dots = \sigma_n \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

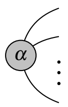
along with their interpretation as tensors. Each H-box is labeled by a constant $a \in \mathbb{C}$, with unlabeled H-boxes corresponding to $a = -1$. Diagrams composed from these can be interpreted as tensor networks: see [39, Section 4.1] for an introduction. In particular, parallel composition of generators corresponds to composing the tensors via tensor product, and connecting wires between generators corresponds to contracting the shared index. The tensors are symmetric over their indices, which implies that *only connectivity matters* – diagrams with the same topology represent the same tensor network.

We also have the following derived generators, defined as

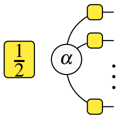


$$=$$





$$=$$



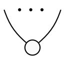
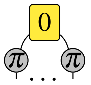
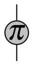
(3)

where the first is an extension of the Z-spider with $\alpha \in \mathbb{C}$ a phase, and the second is the X-spider. In this work, we consider any ZH-diagram to represent its underlying tensor network - hence, if there are n open wires, this represents a tensor with n indices. In particular, if there are no open wires we call this a scalar diagram, since this represents a scalar.

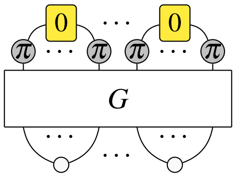
The ZH-calculus is equipped with a set of rewrite rules, which are shown in Appendix A. They are sound with respect to the tensor representation of ZH-diagrams, and also complete: for any pair of diagrams with identical tensor representations, there is a proof that they are equal using these rules.

2.2 #SAT instances as ZH-diagrams

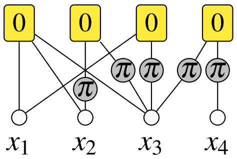
In previous work, de Beaudrap *et al.* [19] gave a translation from **#SAT** instances into ZH-diagrams, which we adopt here. In particular, a CNF Boolean formula ϕ is mapped into a ZH-diagram by translating clauses to zero-labeled H-boxes, variables to Z-spiders and negation to X-spiders:

Variables $\Longleftrightarrow$  Clauses $\Longleftrightarrow$  Negation $\Longleftrightarrow$  (4)

These are combined by connecting the variables to the clauses that they occur in. For unnegated literals, they are connected with a wire. For negated literals, they are connected via an X-spider. This yields the following

$\#(\phi) =$  (5)

where G defines the connections. The scalar value of this diagram is exactly equal to the number of satisfying assignments to ϕ. This can be simplified by canceling all adjacent X-spiders, leaving an X-spider connected between a variable and a clause if and only if that variable is contained in that clause *unnegated*. For example, for $\phi(x_1,x_2,x_3,x_4) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_4)$, we have:

$\#(\phi) =$  (6)

Therefore, a diagram derived this way has m H-boxes, n Z-spiders, at most L X-spiders, and the maximum clause size k corresponds to the maximum degree of any H-box. Note that a complete graphical calculus for **SAT** was introduced recently in [27]. However, the semantics of their diagrams directly correspond to a matrix of True or False values, and hence cannot represent **#SAT** instances directly. In this sense it is similar to the modified ZH-diagrams of [19] where they set $2 = 1$ and consider diagrams over the Boolean semi-ring.

2.3 The CDP algorithm for #SAT

The Counting David-Putnam (CDP) algorithm is an algorithm for solving **#SAT** that was first introduced in 1999 [8], as an extension of the DPLL algorithm [18] for **SAT** solving. It is effectively an optimized depth-first search over all possible assignments of variables in a Boolean formula. The algorithm is based on the following two rules:

1. *Unit Propagation:* The following rewrite holds for any clauses A_i and B_i not containing some literal x:

$$x \wedge (A_1 \vee x) \wedge \cdots \wedge (A_n \vee x) \wedge (B_1 \vee \neg x) \wedge \cdots \wedge (B_m \vee \neg x) = B_1 \wedge \cdots \wedge B_m \tag{7}$$

2. *Variable Branching:* For any variable x appearing in a formula f, the number of satisfying assignments of f is the sum of the numbers of satisfying assignments for

$$f_1 = f \wedge x \qquad \text{and} \qquad f_2 = f \wedge \neg x \tag{8}$$

Algorithm 1: The CDP [8] algorithm for solving **#SAT**.

```
Input: A CNF formula $f$ with $n$ variables and $m$ clauses.
Output: The value of $\#\{\vec{x} \in \{0,1\}^n \mid f(\vec{x}) = 1\}$.
if $f$ contains the clauses $x$ and $\neg x$ for some variable $x$ then
    return 0
end
if $f$ has no clauses remaining then
    return $2^n$
end
Apply unit propagation to $f$ until it is no longer possible.
Pick a variable $x$ that occurs in $f$ and generate $f_1 = f \wedge x$ and $f_2 = f \wedge \neg x$.
return CDP($f_1$) + CDP($f_2$)
```

since in each such assignment, either x or $\neg x$.

The CDP algorithm, given in Algorithm 1, applies these two rules recursively until either a contradiction occurs and so the formula is unsatisfiable, or the formula has no clauses remaining, in which case the number of satisfying assignments is 2^n. Clearly, at each recursive step, the formulas to be considered have fewer variables than at the previous steps, so this procedure terminates.

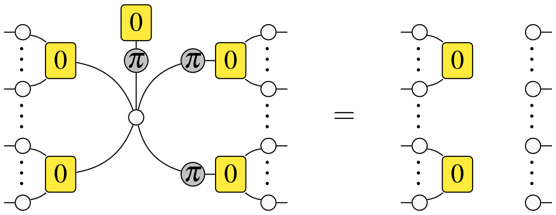
This works better in practice than naively checking every assignment because unit propagation can eliminate many variables, thus removing whole branches from the computation tree. Additionally, the substitution of the assignment into the formula is done incrementally via unit propagation, rather than repeated for every assignment.

3 Results

3.1 Interpreting CDP diagrammatically

To interpret CDP diagrammatically we will first see how the two rules apply to the ZH-diagrams for **#SAT** instances detailed in Section 2.2. The proofs of these lemmas and all the following results are postponed to Appendix E.

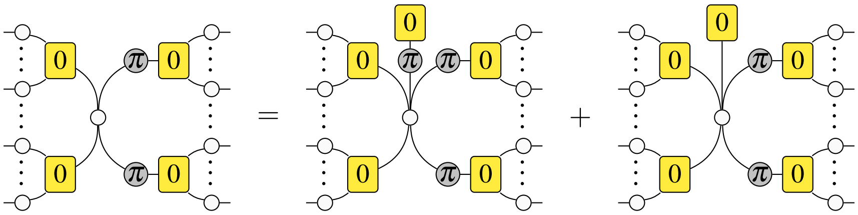
Lemma 1. *The following diagrammatic equivalent to the unit propagation rule holds (without loss of generality, we assume the literal to be propagated is not negated):*



(9)

It can be read as follows - on the left-hand side, the zero H-box with one leg is the clause with a single non-negated literal x, the H-boxes on the left represent the clauses $B_i \vee \neg x$ while the H-boxes on the right represent the clauses $A_i \vee x$. On the right-hand side, we see that the clauses B_i remain, while the clauses $A_i \vee x$ have been removed entirely. Because the variable x is now no longer mentioned, the Z-spider representing it is also removed.

Lemma 2. *The following diagrammatic equivalent to the variable branching rule holds:*



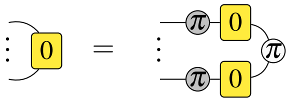
On the left-hand side, we have a variable connected to arbitrary clauses, whereas on the right-hand side we have two terms, each with a clause of one literal introduced onto the variable.

Modifications of the CDP algorithm are used extensively in practice, with some of the best solvers like sharpSAT [43] and Cachet [42] making use of this technique. All the best-known theoretical upper bounds on runtime are based on careful analysis of this algorithm. Note that we left the choice of variable to branch on unspecified - choosing this wisely is crucial to obtaining good runtimes, as we will see in the next section.

3.2 Reduction from #SAT to #2SAT$_\pm$

Valiant [46] showed that **#2SAT** is **#P**-complete, which implies that there is a polynomial-time Turing reduction from **#SAT** to **#2SAT**. Since good upper bounds are known for **#2SAT**, reducing **#SAT** to **#2SAT** is one strategy to obtain better than brute-force bounds that are independent of k. However, the polynomial-time reduction guaranteed by **#P**-completeness maps instances with n variables and m clauses to instances with $O(nm)$ variables, which destroys any advantage we could have gained from faster **#2SAT** algorithms. Instead we present the following linear-time reduction to a weighted variant of **#SAT**:

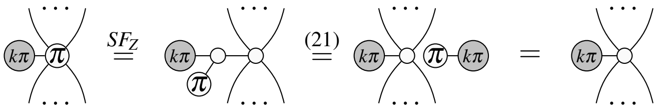
Lemma 3. *The following diagrammatic equivalence holds:*



(10)

This directly generalizes the BW *axiom of the* ∆ZX-*calculus [48].*

This lemma allows us to remove clauses with degree greater than two, at the cost of introducing extra Z-spiders. If we applied this to every clause in a **#SAT** diagram with n variables and m clauses, we would have a diagram with $n+m$ spiders and furthermore, with the exception of π-phases on m of these spiders, this diagram would represent a **#2SAT** instance. Therefore, we will relax our definition of **#SAT** to permit π-phases appearing on the Z-spiders (i.e. on the variables). Note that we have the following



$$\cdot \begin{cases} 1 & \text{if } k=0 \\ -1 & \text{if } k=1 \end{cases} \qquad (11)$$

which implies that these diagrams are the same as **#SAT** instances, but the sign is flipped whenever a variable corresponding to a Z-spider with a π-phase is assigned to be true. The overall sign of the diagram for a particular assignment of variables is given by the parity of the assignment of such variables. We can extend the **#SAT** problem as follows to handle this natively.

Definition 1. *The problem* ***#SAT***$_\pm$ *is defined as follows. Given a CNF formula* $f(x_1,\ldots,x_n)$ *with* n *variables and* m *clauses, and a set* $N \subseteq \{1,\ldots,n\}$*, compute the quantity*

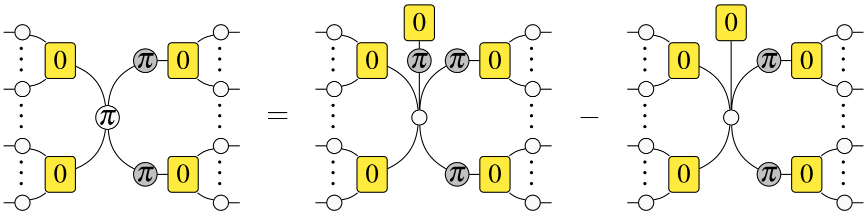
$$\textbf{\#SAT}_\pm(f,N) = \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f(\vec{x}) \; - \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f(\vec{x}) \tag{12}$$

where a vector $\vec{x} \in \mathbb{B}^n$ *has even or odd* N*-parity if* $\bigoplus_{i\in N} x_i = 0$ *or* 1*, respectively.*

Theorem 2. ***#SAT***$_\pm$ *is GapP-complete*

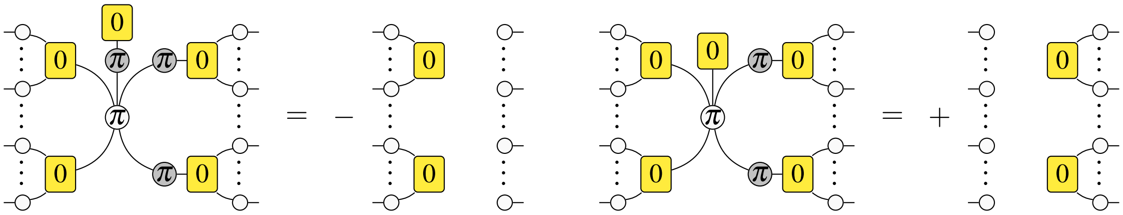
Since **#SAT**$_\pm$ is in GapP, which is strictly harder than #P (for instance, GapP is the closure of #P under subtraction [23]), it may seem that an upper bound for this problem is guaranteed to be worse. However, this is not the case, as the following diagrammatic arguments show that the DPLL algorithm can be easily adapted to handle the sign change as Algorithm 2.

Lemma 4. *The following diagrammatic equivalent to the variable branching rule holds for variables with* π*-phases:*



(13)

Lemma 5. *The following diagrammatic equivalent to the unit propagation rule holds for variables with* π*-phases:*



(14)

With a smart choice of the variables to branch on, the worst-case runtime of Algorithm 2 can be bounded in exactly the same way as the regular CDP algorithm and its variants. This is because the bounds we consider here (e.g [25, 35, 51, 52]) are obtained from two principles:

- Exploiting the fact that the conjunction of unrelated **#SAT** instances combine multiplicatively. Diagrammatically, this corresponds to the parallel composition of scalar diagrams being defined as scalar multiplication. This remains true for the case when phases are present on Z-spiders.

- By analyzing which classes of sub-formulas can occur in any instance, and how they are affected by the unit propagation and branching rules. Diagrammatically, this corresponds to a case analysis on the possible subgraphs of the diagram, ignoring scalar factors. Since unit propagation and branching are unaffected (except for scalar factors) by the presence of phases on the Z-spiders (see Lemmas 4 and 5), this also applies directly in this case.

Algorithm 2: The CDP$_{\pm}$ algorithm solving **#SAT**$_{\pm}$.

```
Input: A CNF formula $f$ with $n$ variables and $m$ clauses, and a set of variables $N \subseteq \{1,\dots,n\}$.
Output: The value of #SAT$_{\pm}(f,N)$.
if $f$ contains an empty clause then
    return $0$
else if $f$ has no clauses then
    return $2^n$
else
    Pick $i \in \{1,\dots,n\}$ according to some strategy.
    $f_1 \leftarrow$ Unit-Propagate$(f \wedge \neg x_i)$
    $f_2 \leftarrow$ Unit-Propagate$(f \wedge x_i)$
    if $i \in N$ then
        return CDP$_{\pm}(f_1,N) -$ CDP$_{\pm}(f_2,N)$
    else
        return CDP$_{\pm}(f_1,N) +$ CDP$_{\pm}(f_2,N)$
    end
end
```

Algorithm 3: The CDP$_{\pm}^{\rightarrow 2}$ algorithm for **#SAT**.

```
Input: A CNF formula $f$ with $n$ variables and $m$ clauses.
Output: The value of #SAT$(f)$.
Generate $f'$ by applying Lemma 3 to every clause in $f$ of width at least three.
Set $N$ to be all the variables labeled with $\pi$-phases.
return CDP$_{\pm}(f',N)$
```

In particular, these bounds do not depend on the specific scalar factors of each diagram, or the way that separate diagrams generated by branching are recombined. This means that the $O^*(1.2377^{\text{variables}})$ bound of Wahlström [51] can be adapted directly. By applying Lemma 3 to any **#SAT** diagram and then applying CDP$_{\pm}$ to the resulting diagram directly, we can evaluate **#SAT** instances in time $O^*(1.2377^{n+m})$ $= O^*(2^{0.3068n+0.3068m})$, which is certainly better than the bound given by decomposing into a sum of diagrams. We will refer to this method, given in Algorithm 3, as CDP$_{\pm}^{\rightarrow 2}$.

Theorem 3 (Restatement of Theorem 1)**.** *Given a CNF formula $f : \{0,1\}^n \rightarrow \{0,1\}$, we can count the number of satisfying assignments $\#\{\vec{x} \in \{0,1\}^n \,|\, f(\vec{x}) = 1\}$ in time $O^*(1.2377^{n+m_{\geq 3}})$, where $m_{\geq 3}$ is the number of clauses of width at least three.*

Proof. Apply the algorithm CDP$_{\pm}^{\rightarrow 2}$ to f using Wahlström's [51] $O^*(1.2377^{\text{variables}})$ algorithm for solving **#2SAT**. Then $m_{\geq 3}$ new (negative) variables will be created by applying Lemma 3, so the overall runtime is given by $O^*(1.2377^{n+m_{\geq 3}})$. □

It remains to ask, when is CDP$_{\pm}^{\rightarrow 2}$ actually useful? First, note that if only positive variables are picked for branching, the action of CDP$_{\pm}$ on a translated **#SAT** diagram is *exactly* the same as the action of regular CDP on the original diagram. Therefore, we would only expect gains when decomposing some of the negative variables (i.e clauses), and thus it is natural to suspect that this bound will only be useful for instances with few clauses.

3.3 Complexity analysis

For instances with a fixed maximum density δ_{max}, and assuming the worst-case of $m_{\geq 3} = m$, we have that $m \leq n\delta_{max}$, and so the runtime of $\mathrm{CDP}_{\pm}^{\to 2}$ is bounded by $O^*(2^{0.3068(1+\delta_{max})n})$. Firstly, we can see that this is better than the naive $O^*(2^n)$ whenever $\delta_{max} < 2.2503$. Since this is independent of k, it means that for any $\delta_{max} < 2.2503$ and sufficiently large k, this beats both the worst-case bound of Dubois [20] and the average-case bound of Williams [55].

Concretely, $\mathrm{CDP}_{\pm}^{\to 2}$ is better than the average-case bounds of Williams [55] whenever $\delta_{max} < 1.217$ and $k \geq 6$, and better than the worst-case bounds of Dubois [20] whenever $k \geq 3$ and $\delta_{max} < 1.858$. The exact bounds for each k are given in Table 2. Clearly, $\mathrm{CDP}_{\pm}^{\to 2}$ offers no improvement on **#2SAT**, but it is also not directly applicable to **#3SAT** - when $\delta < 1.6$, the $O^*(1.4142^m)$ bound of Zhou et al [58] is sharper, and when $\delta \geq 1.6$ the $O^*(1.6423^n)$ bound of Kutzkov [35] is sharper.

For **SAT**, it has been shown that there is a phase transition in the satisfiability of a random formula as δ passes some threshold. Instances with densities near this threshold are known to be hard to solve, and it is known that this threshold scales exponentially with k [1]. However, no bound is known for the equivalent 'hardest' density in **#SAT**. Assuming SETH, our result indicates that the 'hardest' density of **#SAT** must be some $\delta > 2.2503$, since otherwise **#SAT** (and hence **SAT**) could be solved in time better than $O^*(2^n)$.

3.4 A non-diagrammatic argument for $\mathrm{CDP}_{\pm}^{\to 2}$

While we originally found this reduction and algorithm using the ZH-calculus and prefer its diagrammatic presentation, in Appendix C we present a self-contained argument for the reduction from **#SAT** to **#2SAT**$_{\pm}$ without any diagrams that may be more intuitive to readers unfamiliar with graphical calculi.

4 Variations on the main result

4.1 Bounding #SAT in terms of literals

Note that $\mathrm{CDP}_{\pm}^{\to 2}$ maps a **#SAT** instance of m clauses to a **#2SAT** instance of at most L clauses (with negative variables), where L is the number of literals in the original instance. Therefore, applying the upper bound of $O^*(1.1740^m)$ on **#2SAT** found by Wang and Gu [52], we can bound the runtime of $\mathrm{CDP}_{\pm}^{\to 2}$ in terms of literals as $O^*(1.1740^L)$. This implies a better than $O^*(2^n)$ runtime whenever the average degree $\hat{d} = \frac{L}{n}$ of variables in an instance satisfies $\hat{d} < 4.3209$, or the maximal number of clauses d a variable participates in is $d \leq 4$. As far as we are aware this is the first bound of this type for **#SAT** with unrestricted clause width, but similar bounds are known for **SAT** - e.g the $O^*(1.0646^L)$ of Peng and Xiao [40]. If k is small, then better bounds for **#SAT** follow trivially from bounds in terms of m - e.g for $k = 2$, $m \leq \frac{L}{2}$, so [52] implies $O^*(1.0835^L)$.

4.2 An algorithm for low-density #3SAT instances

In the previous section, we noted that $\mathrm{CDP}_{\pm}^{\to 2}$ can't beat existing bounds for **#3SAT** on its own. This is because we have to assume that $m = m_{\geq 3}$ in the worst-case. However, we can use a technique introduced by Kutzkov [35] to take advantage of the extra structure afforded by **#3SAT** and introduce extra branching steps which allow us to assume that $m_{\geq 3} < m$.

Assume we have some **#3SAT** instance f. Every time we branch on a variable x that occurs in d 3-clauses in f, in both branches at least d 3-clauses are eliminated (since each clause will be totally

Algorithm 4: The algorithm CDP$^{3\to 2}_{\pm}$ for solving **#3SAT**.

```
Input: A CNF formula $f$ with $n$ variables and $m$ clauses.
Output: The value of #SAT$(f)$.
if $\delta_3 > \frac{2}{3}$ then
    Pick $x$ in $f$ with maximal 3-degree.
    $f_1 \leftarrow$ Unit-Propagate$(f \wedge \neg x_i)$
    $f_2 \leftarrow$ Unit-Propagate$(f \wedge x_i)$
    return CDP$^{3\to 2}_{\pm}(f_1) +$ CDP$^{3\to 2}_{\pm}(f_2)$
else
    return CDP$^{\to 2}_{\pm}(f)$
end
```

removed in one branch and become a 2-clause in the other). If we know that the density of 3-clauses in f is δ_3, then the average number of 3-clauses a variable is connected to (its average 3-degree) is $3\delta_3$. Therefore, whenever $d-1 < 3\delta_3 \leq d$, there must exist a variable with 3-degree at least d, and so by branching on the variable with the highest 3-degree, we remove at least d 3-clauses.

Following Kutzkov [35], suppose x is the number of variables needed to reduce δ_3 to at most $\frac{d-1}{3}$ by repeatedly branching on the variable with the highest 3-degree. Then we need

$$\frac{d}{3}n - dx \leq \frac{d-1}{3}(n-x) \tag{15}$$

thus $x \leq \frac{n}{2d+1}$, so in the limit of large n, we have $x = \frac{n}{2d+1}$ in the worst case, and $n - \frac{n}{2d+1}$ variables remain unassigned. Therefore, the number of variables we need to branch on to reduce δ_3 to at most $\frac{2}{3}$ is:

$$n_{2/3} = n - n\prod_{i=3}^{d}\left(1 - \frac{1}{2i+1}\right) \tag{16}$$

Since $m_{\geq 3} = \delta_3 n$, after performing this branching on $n_{2/3}$ variables, we will have $2^{n_{2/3}}$ instances, each with $m_{\geq 3} \leq \frac{2}{3}(n - n_{2/3})$, so these can be evaluated with CDP$^{\to 2}_{\pm}$. This strategy, formalized as Algorithm 4, therefore has an overall time bound of:

$$O^*(2^{n_{2/3}})O^*(1.2377^{(n-n_{2/3})+m_{\geq 3}}) \leq O^*(2^{n_{2/3}}1.2377^{(n-n_{2/3})(1+\frac{2}{3})}) = O^*(2^{0.5128n+0.4872n_{2/3}}) \tag{17}$$

In order to calculate the running-time bound for a given maximum δ_3, we can plug $d = \lceil 3\delta_3 \rceil$ into Equation (16) to calculate $n_{2/3}$ as a fraction of n. For example, if $\delta_3 < \frac{5}{3}$ then $d = 5$ and $n_{2/3} = 0.3074n$, yielding a time of

$$O^*(2^{(0.5128+0.4872\cdot 0.3074)n}) = O^*(1.5829^n).$$

Suppose then that $\delta = \delta_3$ (i.e the worst-case), then this bound is better than the bound of Zhou [58] whenever $\delta > 1.2577$ (i.e $d \geq 4$ but not $d = 3$, comparing the $O^*(1.5463^n)$ complexity for $d = 4$ to Zhou's $O^*(1.4142^m)$ to find the exact cutoff point) and the $O^*(1.6423^n)$ bound of Kutzkov [35] whenever $\delta \leq \frac{7}{3}$ (i.e for $d \leq 7$), yielding complexities of $O^*(1.5463^n)$ ($d = 4$) to $O^*(1.6350^n)$ ($d = 7$) respectively. It is possible this bound could be extended to a yield an improved bound on general **#3SAT** using a case analysis similar to Kutzkov's, but this is quite complicated so we postpone exploring this to future work.

4.3 Solving #SAT with arbitrary phases

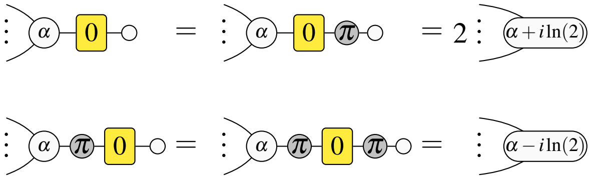
While allowing π-phases on variables has allowed us to find a simple reduction from **#SAT** to **#2SAT**$_\pm$ which can be solved with CDP$_\pm$, the CDP algorithm easily extends to arbitrary phases in the same way. Indeed let us define a generalization of the **#SAT** problem, **#SAT**$_\mathscr{A}$ - this is exactly the weighted model counting problem where the weights are restricted to $\mathscr{A}$.

Definition 2. *The problem **#SAT**$_\mathscr{A}$ for $\mathscr{A} \subseteq \mathbb{C} \setminus \{0\}$ is defined as follows. Given a CNF formula $f(x_1, \dots)$ with n variables and m clauses and a vector $A \in \mathscr{A}^n$, compute the quantity:*

$$\textbf{\#SAT}_\mathscr{A}(f, A) = \sum_{\vec{x} \in \mathbb{B}^n} \left(\prod_{i=1}^{n} A_i^{x_i} \right) f(\vec{x}) \tag{18}$$

It is easy to see then that **#SAT** $=$ **#SAT**$_{\{1\}}$, and **#SAT**$_\pm$ $=$ **#SAT**$_{\{1,-1\}}$: let $A_j = -1$ if $j \in N$ and $A_j = 1$ otherwise. A straightforward adaptation of the CDP algorithm can solve **#SAT**$_\mathscr{A}$; see Algorithm 5.

The complex numbers in $\mathscr{A}$ on the variables of an instance can be easily represented in the ZH-calculus, by generalizing Eq. (3). The variable branching and unit propagation rules then generalize from Eq. (22). The advantage of working with **#SAT**$_\mathscr{A}$ is that by expanding $\mathscr{A}$, we are afforded additional rewriting rules on the corresponding ZH-diagrams. Moving from $\{1\}$ to $\{1, -1\}$ allowed us to rewrite arbitrary arity zero-labeled H-boxes into arity two H-boxes. Further expanding this to $\{\frac{k}{2} \mid k \in \mathbb{Z}\}$ allows us the following rule, removing (up to a scalar) any variables that only occur once in a formula:



(19)

This is a limited form of pure-literal elimination, a rewrite rule that is usually only valid in **SAT** and not **#SAT**. Applying this simplification to the formula recursively (after unit propagation in the CDP algorithm, i.e between lines 7 and 8 of Algorithm 1), we may assume that every variable has degree at least two. This would allow improvement on early bounds such as [20], while being much simpler. Therefore, an interesting avenue for further research would be investigating how this approach of weighting variables could be used to simplify **#SAT** instances or find upper bounds on runtime. Another example is given in Appendix B where we show how this leads to an algorithm for simulating quantum circuits with runtime in terms of the total number of gates.

5 Conclusion

In this paper, we used the ZH-calculus to study the **#SAT** problem and produced an upper bound which does not depend on the clause width k. We believe bounds of this kind were previously only known for the decision variant **SAT** [56]. The bound is less than $O^*(2^n)$ whenever the clause density $\delta = \frac{n}{m}$ is smaller than 2.2503, suggesting that the 'hardest' density of **#SAT** problems must be some $\delta > 2.2503$, assuming the strong exponential time hypothesis. We found these bounds by rephrasing the **#SAT** problem in terms of ZH-diagrams, and generalizing known rewrite rules to give a reduction from **#SAT** to **#2SAT**$_\pm$, a weighted variant of **#SAT** that can be solved with Wahlström's [51] variant of the CDP algorithm [8]. Using a more involved analysis and algorithm we also improved on the upper bound for **#3SAT** for $1.2577 < \delta < \frac{7}{3}$. In addition, using a result of Wang and Gu [52], we produced an explicit bound of

Algorithm 5: The CDP$_{\mathscr{A}}$ algorithm for solving **#SAT**$_{\mathscr{A}}$.

```
Input: A CNF formula $f$ with $n$ variables and $m$ clauses, and $A \in \mathscr{A}^n$.
Output: The value of #SAT$_{\mathscr{A}}(f,A)$.
if $f$ contains an empty clause then
    return 0
else if $f$ has no clauses then
    return $2^n$
else
    Pick $i \in \{1,\ldots,n\}$ according to some strategy.
    $f_1 \leftarrow$ Unit-Propagate$(f \wedge \neg x_i)$
    $f_2 \leftarrow$ Unit-Propagate$(f \wedge x_i)$
    return CDP$_{\mathscr{A}}(f_1,A) + A_i$CDP$_{\mathscr{A}}(f_2,A)$
end
```

$O^*(1.1740^L)$ for **#SAT** in terms of the number of literals L, to our knowledge the first such non-trivial bound for **#SAT**. A summary of all the bounds obtained in this paper is presented in Table 1. We suggest extending this technique of reducing **#SAT** to weighted **#SAT** as an avenue of future research.

Our results show that graphical calculi can lead to concrete algorithmic improvements in areas where significant research has already been done, even when originally intended for a different domain like quantum computing. An interesting question then is in which other domains we can make improvements by framing the problem using graphical reasoning.

Acknowledgments

We thank Matty Hoban and the anonymous QPL reviewers for helpful feedback. The majority of this work was done while TL was a student at the University of Oxford, and the main result is also presented in an MSc thesis [36]. JvdW acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101018390.

References

[1] Dimitris Achlioptas, Assaf Naor & Yuval Peres (2005): *Rigorous Location of Phase Transitions in Hard Optimization Problems*. Nature 435(7043), pp. 759–764, doi:10.1038/nature03602.

[2] 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.

[3] Miriam Backens & Aleks Kissinger (2019): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*. Electronic Proceedings in Theoretical Computer Science 287, pp. 23–42, doi:10.4204/EPTCS.287.2. arXiv:1805.02175.

[4] Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering & Sal Wolffs (2021): *Completeness of the ZH-calculus*, doi:10.48550/arXiv.2103.06610. arXiv:2103.06610.

[5] 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.

[6] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Techniques to Reduce $\pi/4$-Parity-Phase Circuits, Motivated by the ZX Calculus*. 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. 131–149, doi:10.4204/EPTCS.318.9.

[7] A. Biere, M. Heule & H. van Maaren (2009): *Handbook of Satisfiability*. IOS Press, Incorporated, doi:10.3233/FAIA336.

[8] E. Birnbaum & E. L. Lozinskii (1999): *The Good Old Davis-Putnam Procedure Helps Counting Models*. *Journal of Artificial Intelligence Research* 10, pp. 457–477, doi:10.1613/jair.601. arXiv:1106.0218.

[9] Guillaume Boisseau & Robin Piedeleu (2022): *Graphical Piecewise-Linear Algebra*. In Patricia Bouyer & Lutz Schröder, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 101–119, doi:10.1007/978-3-030-99253-8_6.

[10] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2019): *Diagrammatic Algebra: From Linear to Concurrent Systems*. *Proc. ACM Program. Lang.* 3(POPL), doi:10.1145/3290338.

[11] Filippo Bonchi, Robin Piedeleu, Pawel Sobociński & Fabio Zanasi (2019): *Graphical Affine Algebra*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–12, doi:10.1109/LICS.2019.8785877.

[12] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2014): *A Categorical Semantics of Signal Flow Graphs*. In Paolo Baldan & Daniele Gorla, editors: *CONCUR 2014 – Concurrency Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 435–450, doi:10.1007/978-3-662-44584-6_30.

[13] 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.

[14] Enrique Cervero Martín, Kirill Plekhanov & Michael Lubasch (2022): *Barren plateaus in quantum tensor network optimization*. doi:10.48550/arXiv.2209.00292. arXiv:2209.00292.

[15] Bob Coecke & Ross Duncan (2008): *Interacting quantum observables*. In: *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, doi:10.1007/978-3-540-70583-3_25.

[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. arXiv:0906.4725.

[17] Carsten Damm, Markus Holzer & Pierre McKenzie (2002): *The Complexity of Tensor Calculus*. *Computational Complexity* 11(1/2), pp. 54–89, doi:10.1007/s00037-000-0170-4.

[18] Martin Davis, George Logemann & Donald Loveland (1962): *A Machine Program for Theorem-Proving*. *Communications of the ACM* 5(7), pp. 394–397, doi:10.1145/368273.368557.

[19] Niel de Beaudrap, Aleks Kissinger & Konstantinos Meichanetzidis (2021): *Tensor Network Rewriting Strategies for Satisfiability and Counting*. *Electronic Proceedings in Theoretical Computer Science* 340, pp. 46–59, doi:10.4204/EPTCS.340.3. arXiv:2004.06455.

[20] Olivier Dubois (1991): *Counting the Number of Solutions for Instances of Satisfiability*. *Theoretical Computer Science* 81(1), pp. 49–64, doi:10.1016/0304-3975(91)90315-S.

[21] 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.

[22] Richard D. P. East, Pierre Martin-Dussaud & John van de Wetering (2021): *Spin-networks in the ZX-calculus*. doi:10.48550/arXiv.2111.03114. arXiv:2111.03114.

[23] Stephen A Fenner, Lance J Fortnow & Stuart A Kurtz (1994): *Gap-Definable Counting Classes*. *Journal of Computer and System Sciences* 48(1), pp. 116–148, doi:10.1016/S0022-0000(05)80024-8.

[24] Johannes K. Fichte, Markus Hecher & Florim Hamiti (2020): *The Model Counting Competition 2020*, doi:10.48550/arXiv.2012.01323. arXiv:2012.01323.

[25] Martin Fürer & Shiva Prasad Kasiviswanathan (2007): *Algorithms for Counting 2-Sat Solutions and Colorings with Applications*. In Ming-Yang Kao & Xiang-Yang Li, editors: *Algorithmic Aspects in Information and Management*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 47–57, doi:10.1007/978-3-540-72870-2_5.

[26] 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.

[27] Tao Gu, Robin Piedeleu & Fabio Zanasi (2022): *A Complete Diagrammatic Calculus for Boolean Satisfiability*. doi:10.48550/arXiv.2211.12629. arXiv:2211.12629.

[28] Amar Hadzihasanovic (2015): *A diagrammatic axiomatisation for qubit entanglement*. In: 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science, IEEE, pp. 573–584, doi:10.1109/LICS.2015.59.

[29] Amar Hadzihasanovic, Kang Feng Ng & Quanlong Wang (2018): *Two Complete Axiomatisations of Pure-state Qubit Quantum Computing*. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18, ACM, New York, NY, USA, pp. 502–511, doi:10.1145/3209108.3209128.

[30] Michael Hanks, Marta P. Estarellas, William J. Munro & Kae Nemoto (2020): *Effective Compression of Quantum Braided Circuits Aided by ZX-Calculus*. Physical Review X 10, p. 041030, doi:10.1103/PhysRevX.10.041030.

[31] 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.

[32] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. Physical Review A 102, p. 022406, doi:10.1103/PhysRevA.102.022406.

[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 & 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.

[35] Konstantin Kutzkov (2007): *New Upper Bound for the #3-SAT Problem*. Information Processing Letters 105(1), pp. 1–5, doi:10.1016/j.ipl.2007.06.017.

[36] Tuomas Laakkonen (2022): *Graphical Stabilizer Decompositions For Counting Problems*. Master's thesis, University of Oxford. Available at https://www.cs.ox.ac.uk/people/aleks.kissinger/theses/laakkonen-thesis.pdf.

[37] Tuomas Laakkonen, Konstantinos Meichanetzidis & John van de Wetering (2023): *Picturing Counting Reductions with the ZH-Calculus*. Electronic Proceedings in Theoretical Computer Science 384, p. 89–113, doi:10.4204/eptcs.384.6.

[38] Kang Feng Ng & Quanlong Wang (2018): *Completeness of the ZX-calculus for Pure Qubit Clifford+T Quantum Mechanics*, doi:10.48550/arXiv.1801.07993. arXiv:1801.07993.

[39] Román Orús (2014): *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*. Annals of Physics 349, pp. 117–158, doi:10.1016/j.aop.2014.06.013.

[40] Junqiang Peng & Mingyu Xiao (2021): *Further Improvements for SAT in Terms of Formula Length*, doi:10.48550/arXiv.2105.06131. arXiv:2105.06131.

[41] Robin Piedeleu & Fabio Zanasi (2021): *A String Diagrammatic Axiomatisation of Finite-State Automata*. In Stefan Kiefer & Christine Tasson, editors: Foundations of Software Science and Computation Structures, Springer International Publishing, Cham, pp. 469–489, doi:10.1007/978-3-030-71995-1_24.

[42] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz & Toniann Pitassi (2004): *Combining Component Caching and Clause Learning for Effective Model Counting*. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings.

[43] Marc Thurley (2006): *sharpSAT – Counting Models with Advanced Component Caching and Implicit BCP*. In Armin Biere & Carla P. Gomes, editors: *Theory and Applications of Satisfiability Testing - SAT 2006*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 424–429, doi:10.1007/11814948_38.

[44] Seinosuke Toda (1991): *PP Is as Hard as the Polynomial-Time Hierarchy*. *SIAM Journal on Computing* 20(5), pp. 865–877, doi:10.1137/0220053.

[45] Alex Townsend-Teague & Konstantinos Meichanetzidis (2021): *Classifying Complexity with the ZX-Calculus: Jones Polynomials and Potts Partition Functions*. doi:10.48550/arXiv.2103.06914. arXiv:2103.06914.

[46] Leslie G. Valiant (1979): *The Complexity of Enumeration and Reliability Problems*. *SIAM Journal on Computing* 8(3), pp. 410–421, doi:10.1137/0208032.

[47] 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.

[48] Renaud Vilmart (2019): *A ZX-Calculus with Triangles for Toffoli-Hadamard, Clifford+T, and Beyond*. *Electronic Proceedings in Theoretical Computer Science* 287, pp. 313–344, doi:10.4204/EPTCS.287.18. arXiv:1804.03084.

[49] Renaud Vilmart (2021): *Quantum Multiple-Valued Decision Diagrams in Graphical Calculi*. In Filippo Bonchi & Simon J. Puglisi, editors: *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 202, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 89:1–89:15, doi:10.4230/LIPIcs.MFCS.2021.89.

[50] Renaud Vilmart (2021): *The Structure of Sum-over-Paths, Its Consequences, and Completeness for Clifford*. In Stefan Kiefer & Christine Tasson, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 531–550, doi:10.1007/978-3-030-71995-1_27.

[51] Magnus Wahlström (2008): *A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances*. In Martin Grohe & Rolf Niedermeier, editors: *Parameterized and Exact Computation*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 202–213, doi:10.1007/978-3-540-79723-4_19.

[52] Honglin Wang & Wenxiang Gu (2013): *The Worst Case Minimized Upper Bound in #2-SAT*. In Wei Lu, Guoqiang Cai, Weibin Liu & Weiwei Xing, editors: *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, Lecture Notes in Electrical Engineering, Springer, Berlin, Heidelberg, pp. 675–682, doi:10.1007/978-3-642-34522-7_72.

[53] Quanlong Wang (2021): *An Algebraic Axiomatisation of ZX-calculus*. In Benoît Valiron, Shane Mansfield, Pablo Arrighi & Prakash Panangaden, editors: *Proceedings 17th International Conference on Quantum Physics and Logic, Paris, France, June 2 - 6, 2020*, *Electronic Proceedings in Theoretical Computer Science* 340, Open Publishing Association, pp. 303–332, doi:10.4204/EPTCS.340.16.

[54] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. doi:10.48550/arXiv.2012.13966. arXiv:2012.13966.

[55] Ryan Williams (2004): *On Computing K-CNF Formula Properties*. In Enrico Giunchiglia & Armando Tacchella, editors: *Theory and Applications of Satisfiability Testing*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 330–340, doi:10.1007/978-3-540-24605-3_25.

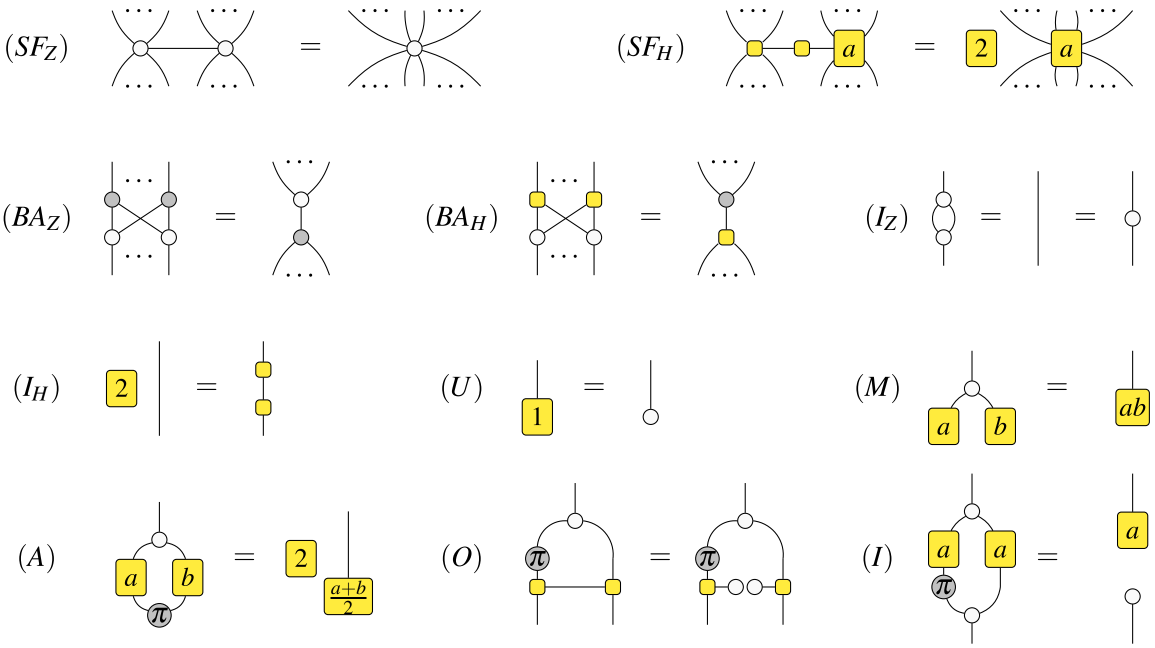
[56] Masaki Yamamoto (2005): *An Improved O(1.234m̂)-Time Deterministic Algorithm for SAT*. In Xiaotie Deng & Ding-Zhu Du, editors: *Algorithms and Computation*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 644–653, doi:10.1007/11602613_65.

[57] Fabio Zanasi (2015): *Interacting Hopf Algebras: the theory of linear systems*. Ph.D. thesis, Ecole Normale Superieure de Lyon, doi:10.48550/arXiv.1805.03032. Available at https://arxiv.org/abs/1805.03032.

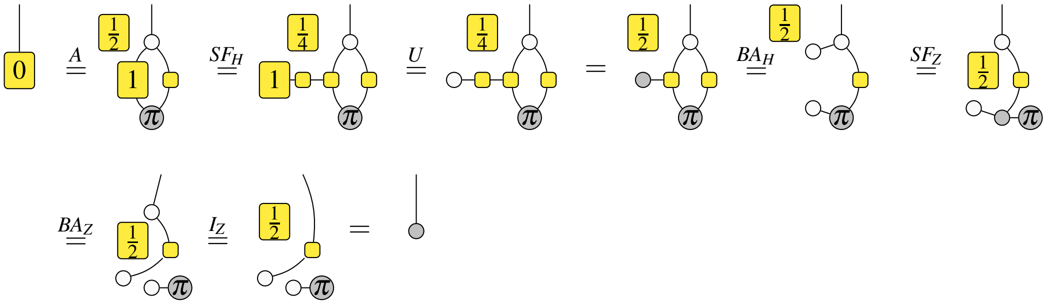
[58] Junping Zhou, Minghao Yin & Chunguang Zhou (2010): *New Worst-Case Upper Bound for #2-SAT and #3-SAT with the Number of Clauses as the Parameter*. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, AAAI Press, Atlanta, Georgia, pp. 217–222, doi:10.48550/arXiv.1006.1537.

A Rewriting Rules

The ZH-calculus [3] has the following rewriting rules:

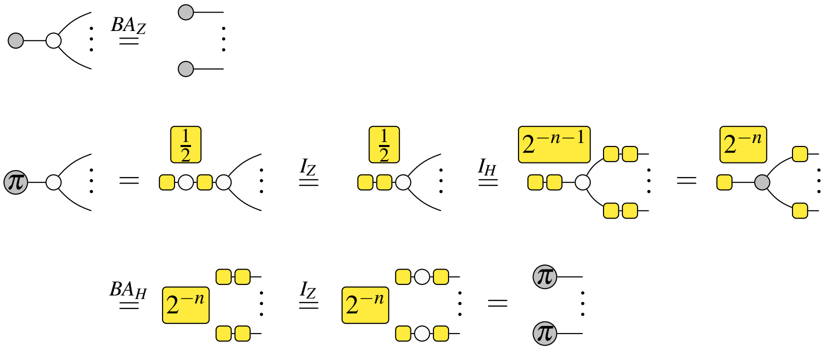


We will also used some derived rules. In particular, we have that



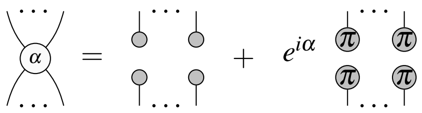
(20)

and also that:



(21)

Finally, we recall the presentation of the Z-spider as a sum over basis states:



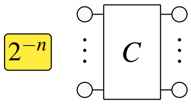
(22)

B Upper Bounds on Quantum Circuit Simulation

Extending the ideas from Section 4.3, we can produce an upper bound on the time required to exactly compute the amplitude of a quantum computation. This task is naturally represented in terms of tensor networks [39], which can be presented in terms of a graphical calculus, see [54, page 6] for an example in the ZX-calculus. Note that while this task is easily shown to be **#P**-hard, it is not obviously in **#P** and so it is hard to relate directly to **#SAT**. By relaxing our definition of **#SAT** to **#SAT**$_{\mathbb{C}}$, we can do this translation more straightforwardly, but still apply existing knowledge about **#SAT**.

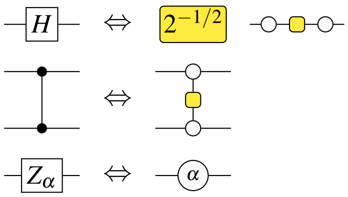
Specifically, given a quantum circuit C with n qubits and G gates representing a unitary U over the CZ, Hadamard, and Z-phase gate set, the quantity $\langle +|U|+\rangle$ can be represented by the following ZH-diagram:

$\langle +|U|+\rangle =$



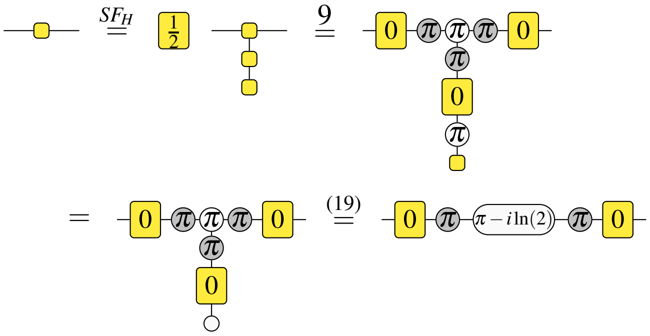
(23)

Where the portion marked C is assembled from the following components



(24)

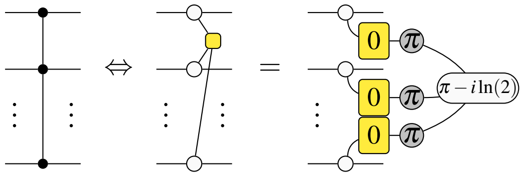
by placing them according to their connections in the quantum circuit. However, we can translate these to **#2SAT**$_{\mathbb{C}}$ diagrams, up to scalar factors, by applying the following rewrite:



(25)

To map $\langle +|U|+\rangle$ into this form, we apply this translation and then spider fusion wherever possible. This leaves a **#2SAT**$_{\mathbb{C}}$ diagram and a scalar factor. Therefore, to compute the value of $\langle +|U|+\rangle$, we can compute the model count of the corresponding **#2SAT**$_{\mathbb{C}}$ instance by CDP$_{\mathbb{C}}$ and multiply it by the scalar factor. Since Hadamard gates and CZ gates each add two clauses, and Z-phase gates add no clauses, we have $m \leq 2G$. Hence, by applying the $O^*(1.1740^m)$ bound for **#2SAT** by Wang and Gu [52], we can calculate $\langle +|U|+\rangle$ in time $O^*(1.3783^G)$. This is better than statevector simulation whenever $G \leq 2.16n$.

It is also possible to work with circuits that contain $\mathbf{C}^k\mathbf{Z}$ gates directly, since we have the following translation, as before:



(26)

Then each gate adds at most $k+1$ clauses to the **#2SAT** instance, leading to an overall runtime of $O^*(1.1740^{(k+1)G})$. For instance, for $k=2$ this is $O^*(1.6181^G)$, which is substantially better than the corresponding runtime given by decomposing CCZ gates into single- and two-qubit gates (which would be $O^*(6.8552^G)$, as each CCZ would require twelve clauses). It may be possible to give better bounds by analyzing the spider fusion that occurs in this translation and applying bounds for **#2SAT** in terms of variables rather than clauses, but we postpone this to future work.

C A Non-diagrammatic Argument for $\mathrm{CDP}_{\pm}^{\to 2}$

Suppose we have a function $\phi : \mathbb{B}^n \to \mathbb{B}$ given by

$$\phi(x_1,\ldots,x_n) = \bigwedge_{i=1}^{m} (c_{i1} \vee c_{i2} \vee \cdots \vee c_{ik_i})$$

then we can define $\phi' : \mathbb{B}^{n+m} \to \{-1,0,1\}$ given by:

$$\phi'(x_1,\ldots,x_n,y_1\ldots,y_m) = (-1)^{\Sigma_i y_i}\left[\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij} \vee \neg y_i)\right] = (-1)^{\Sigma_i y_i}\left[\bigwedge_{i=1}^{m} \neg\left(y_i \wedge \bigvee_{j=1}^{k_i} c_{ij}\right)\right]$$

Lifting Boolean logic to integer arithmetic, we have that

$$\begin{aligned}
\phi'(x_1,\ldots,x_n,y_1\ldots,y_m) &= (-1)^{\Sigma_i y_i}\prod_{i=1}^{m}\left(1 - y_i\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\right) = (-1)^{\Sigma_i y_i}\sum_{S\subseteq[1,m]}\prod_{i\in S}(-y_i)\left[\bigvee_{j=1}^{k_i} c_{ij}\right] \\
&= \sum_{S\subseteq[1,m]}(-1)^{|S|+\Sigma_i y_i}\left[\bigwedge_{i\in S} y_i\right]\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right] \\
&= \sum_{S\subseteq[1,m]}(-1)^{\Sigma_{i\notin S} y_i}\left[\bigwedge_{i\in S} y_i\right]\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]
\end{aligned}$$

and summing over all possibilities for y_i,

$$\begin{aligned}
\sum_{y_1,\ldots,y_m\in\mathbb{B}}\phi'(x_1,\ldots,x_n,y_1\ldots,y_m) &= \sum_{S\subseteq[1,m]}\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\sum_{y_1,\ldots,y_m\in\mathbb{B}}(-1)^{\Sigma_{i\notin S} y_i}\left[\bigwedge_{i\in S} y_i\right] \\
&= \sum_{S\subseteq[1,m]}\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\cdot\begin{cases}1 & |S| = m \\ 0 & |S| < m\end{cases} = \left[\bigwedge_{i=1}^{m}\bigvee_{j=1}^{k_i} c_{ij}\right] = \phi(x_1,\ldots,x_n)
\end{aligned}$$

therefore, to compute $\#(\phi)$, it is sufficient to sum over all values of ϕ':

$$\#(\phi) = \sum_{x_1,\ldots,x_n\in\mathbb{B}}\phi(x_1,\ldots,x_n) = \sum_{x_1,\ldots,x_n,y_1,\ldots,y_m\in\mathbb{B}}\phi'(x_1,\ldots,x_n,y_1,\ldots,y_m)$$

Finally, we can see that

$$\begin{aligned}
\#(\phi) &= \sum_{x_1,\dots,x_n,y_1,\dots,y_m \in \mathbb{B}} (-1)^{\sum_i y_i} \left[\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{k_i} (\neg c_{ij} \vee \neg y_i) \right] \\
&= \sum_{\substack{x_1,\dots,x_n,y_1,\dots,y_m \in \mathbb{B} \\ \sum_i y_i \text{ even}}} \left[\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{k_i} (\neg c_{ij} \vee \neg y_i) \right] - \sum_{\substack{x_1,\dots,x_n,y_1,\dots,y_m \in \mathbb{B} \\ \sum_i y_i \text{ odd}}} \left[\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{k_i} (\neg c_{ij} \vee \neg y_i) \right] \\
&= \mathbf{\#SAT}_{\pm} \left(\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{k_i} (\neg c_{ij} \vee \neg y_i), \{y_1, \dots, y_m\} \right)
\end{aligned}$$

by definition, and thus this defines a reduction from **#SAT** to **#2SAT**$_\pm$ which can be computed using Algorithm 2.

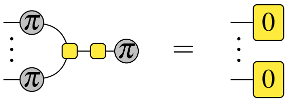
The argument that Wahlström's bound [51] can be applied to Algorithm 2 is the same as in Section 3.2: since the structure of the algorithm remains the same (regardless of scalar factors, unit propagation and branching still generate the same structures), and because unrelated instances combine multiplicatively in that

$$\mathbf{\#SAT}_{\pm}(f_1(x_1,\dots,x_n) \wedge f_2(y_1,\dots,y_m), N_1 \cup N_2) = \mathbf{\#SAT}_{\pm}(f_1(x_1,\dots,x_n), N_1)\mathbf{\#SAT}_{\pm}(f_2(y_1,\dots,y_m), N_2)$$

then the bound doesn't distinguish between Algorithm 1 and 2, so it applies directly.

D Additional Lemmas

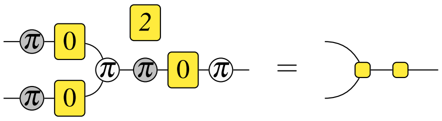
Lemma 6. *The following diagrammatic equivalence holds:*



(27)

Proof. This follows from Lemma 2.28 in [4] and Equation (20). □

Lemma 7. *The following diagram equivalence holds:*

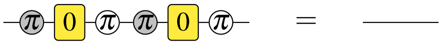


(28)

This is translated from a quantum circuit identity of Ng and Wang [38] and is a generalization of the rules HT *and* BW *from the* ΔZX$_\pi$*-calculus [48], a system for describing tensor networks that is closely related to ZH-calculus.*

Proof. This can be verified by concrete calculation of the matrices. □

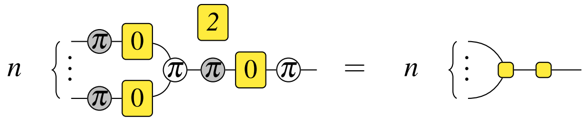
Lemma 8. *The following diagram equivalence holds:*



(29)

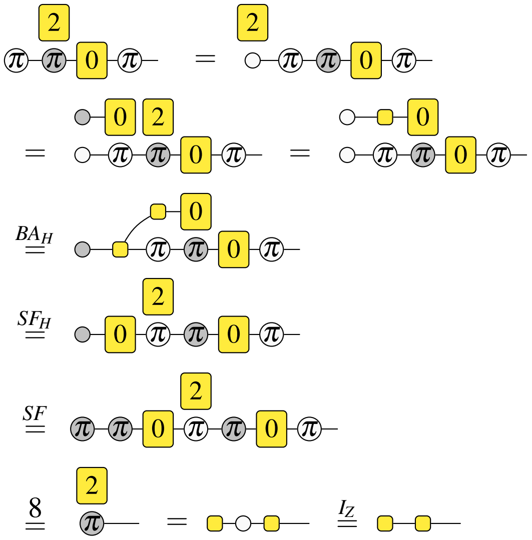
Proof. This can be verified by concrete calculation of the matrices. □

Lemma 9. *The following diagram equivalence holds for all $n \geq 0$:*



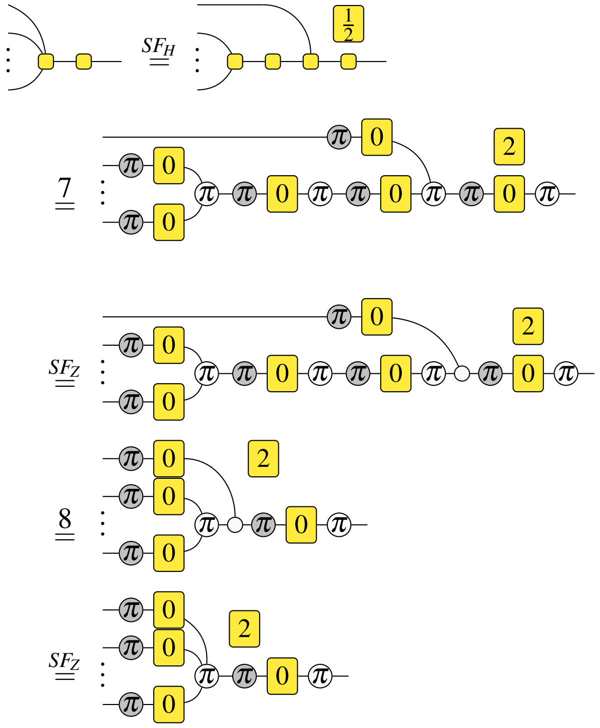
(30)

Proof. We proceed by induction on n. For the case of $n = 0$:



(31)

For the case of $n = k+1$, assuming the result holds for k:

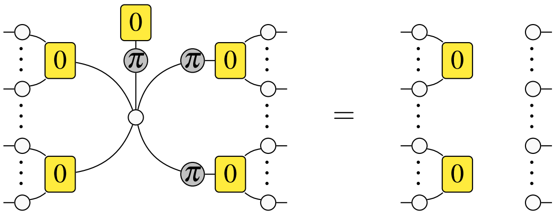


(32)

□

E Proofs of Theorems

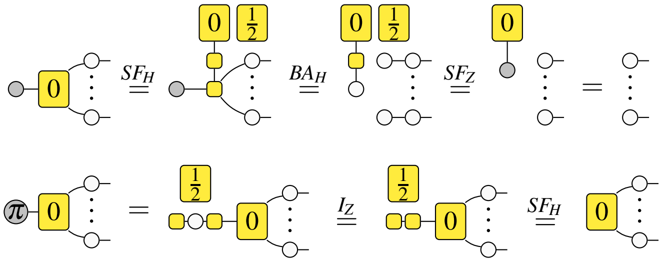
Lemma 1. *The following diagrammatic equivalent to the unit propagation rule holds (without loss of generality, we assume the literal to be propagated is not negated):*



(9)

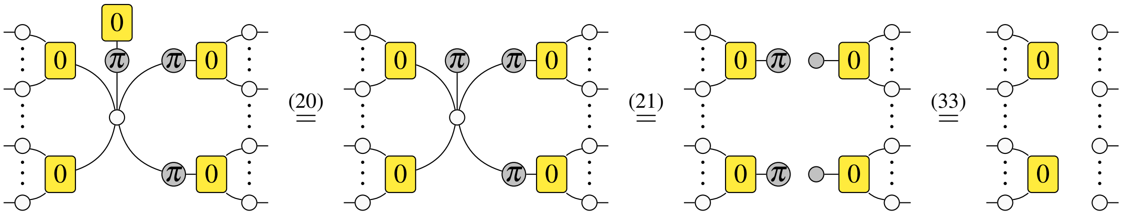
It can be read as follows - on the left-hand side, the zero H-box with one leg is the clause with a single non-negated literal x, the H-boxes on the left represent the clauses $B_i \vee \neg x$ while the H-boxes on the right represent the clauses $A_i \vee x$. On the right-hand side, we see that the clauses B_i remain, while the clauses $A_i \vee x$ have been removed entirely. Because the variable x is now no longer mentioned, the Z-spider representing it is also removed.

Proof. First, note that



(33)

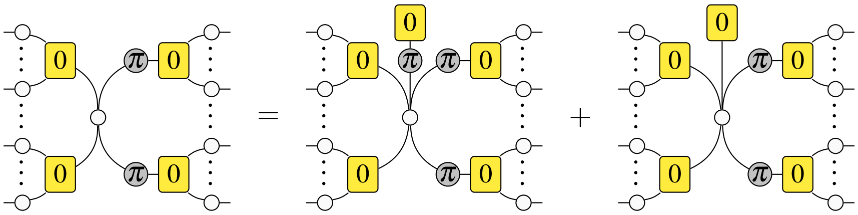
and so we have that



(34)

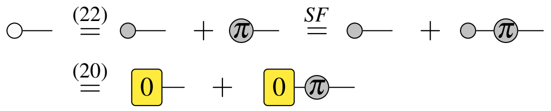
which completes the proof. □

Lemma 2. *The following diagrammatic equivalent to the variable branching rule holds:*



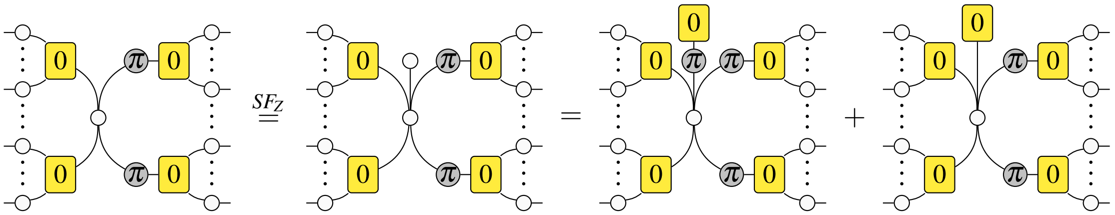
On the left-hand side, we have a variable connected to arbitrary clauses, whereas on the right-hand side we have two terms, each with a clause of one literal introduced onto the variable.

Proof. This follows from writing the Z-spider as a sum



(35)

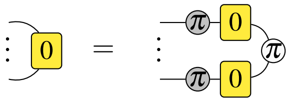
and applying the SF_Z rule to the central spider:



(36)

□

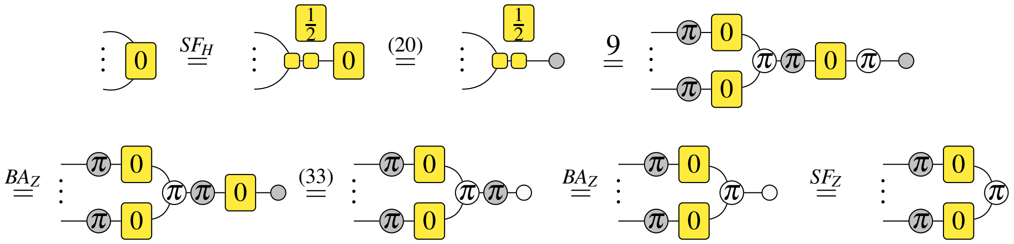
Lemma 3. *The following diagrammatic equivalence holds:*



(10)

This directly generalizes the BW *axiom of the* ΔZX*-calculus [48].*

Proof. We can see the following



which completes the proof. □

Theorem 2. *#SAT$_\pm$ is GapP-complete*

Proof. Suppose there is a problem M in **GapP**, then the aim is to compute $\text{gap}_M = \#M - \#\overline{M}$ where $\#M$ ($\#\overline{M}$) is the number of accepting (rejecting) paths of a non-deterministic Turing machine M. Clearly $\#M$ and $\#\overline{M}$ are both in #P. Because the Cook-Levin reduction from NP to **SAT** is parsimonious, **#SAT** is #P-complete, and there exist CNF formulae f_M and $f_{\overline{M}}$ such that $\textbf{\#SAT}(f_M) = \#M$ and $\textbf{\#SAT}(f_{\overline{M}}) = \#\overline{M}$. Define the following formula:

$$f_{M-\overline{M}} = \left(\bigwedge_{i=1}^{m_M} C_M^i \vee z\right) \wedge \left(\bigwedge_{j=1}^{n_M} \neg x_M^j \vee \neg z\right) \wedge \left(\bigwedge_{i=1}^{m_{\overline{M}}} C_{\overline{M}}^i \vee \neg z\right) \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x_{\overline{M}}^j \vee z\right)$$

where C_M^i ($C_{\overline{M}}^i$) and x_M^j ($x_{\overline{M}}^j$) are the clauses and variables of f_M ($f_{\overline{M}}$) respectively, and z is a fresh variable. When z is false, $f_{M-\overline{M}}$ reduces to

$$f_{M-\overline{M}} \wedge \neg z = \left(\bigwedge_{i=1}^{m_M} C_M^i\right) \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x_{\overline{M}}^j\right) = f_M \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x_{\overline{M}}^j\right)$$

which has exactly $\mathbf{\#SAT}(f_M) = \#M$ satisfying solutions, and likewise with z true, we have

$$f_{M-\overline{M}} \wedge z = f_{\overline{M}} \wedge \left(\bigwedge_{j=1}^{n_M} \neg x_M^j\right)$$

which has exactly $\mathbf{\#SAT}(f_{\overline{M}}) = \#\overline{M}$ satisfying solutions. Finally let $N = \{z\}$, then an assignment of $f_{M-\overline{M}}$ has odd or even N-parity exactly when z is true or false respectively. Therefore,

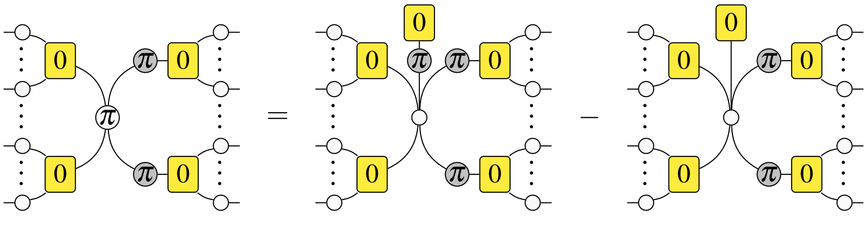
$$\begin{aligned}\mathbf{\#SAT}_{\pm}(f_{M-\overline{M}}, N) &= \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f_{M-\overline{M}}(\vec{x}) - \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f_{M-\overline{M}}(\vec{x}) = \mathbf{\#SAT}(f_{M-\overline{M}} \wedge \neg z) - \mathbf{\#SAT}(f_{M-\overline{M}} \wedge z) \\ &= \mathbf{\#SAT}(f_M) - \mathbf{\#SAT}(f_{\overline{M}}) = \#M - \#\overline{M} = \mathrm{gap}_M\end{aligned}$$

so there is a polynomial-time counting reduction from **GapP** to $\mathbf{\#SAT}_{\pm}$, and it must be **GapP**-hard. Furthermore, let f, N be an instance of $\mathbf{\#SAT}_{\pm}$ with variables x_i, then clearly we have that

$$\mathbf{\#SAT}_{\pm}(f,N) = \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f(\vec{x}) - \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f(\vec{x}) = \mathbf{\#SAT}\left(f \wedge \neg\left(\bigoplus_{i\in N} x_i\right)\right) - \mathbf{\#SAT}\left(f \wedge \left(\bigoplus_{i\in N} x_i\right)\right)$$

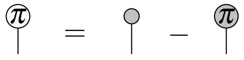
since $g = 1 \iff g$ and $g = 0 \iff \neg g$, so $\neg(\bigoplus_{i\in N} x_i)$ is the same as asserting even N-parity (and likewise odd N-parity). But **GapP** is the closure of #P under subtraction, so $\mathbf{\#SAT}_{\pm}(f,N)$ is in **GapP**. Thus it follows that $\mathbf{\#SAT}_{\pm}$ is **GapP**-complete. □

Lemma 4. *The following diagrammatic equivalent to the variable branching rule holds for variables with π-phases:*



(13)

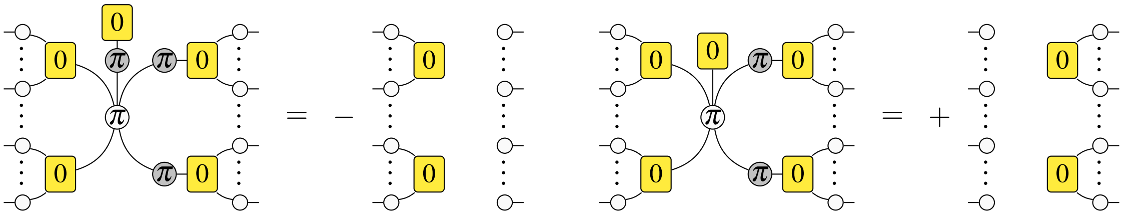
Proof. This follows from the sum



(37)

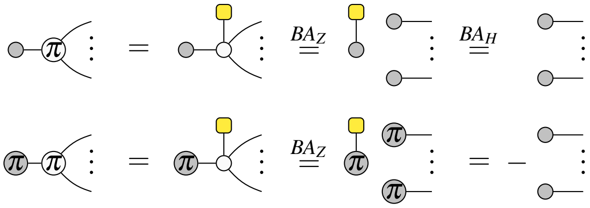
together with the proof of Lemma 2. □

Lemma 5. *The following diagrammatic equivalent to the unit propagation rule holds for variables with π-phases:*



(14)

Proof. This follows from the identities



(38)

together with the proof of Lemma 1. □

Quantum Algorithms for Compositional Text Processing

Tuomas Laakkonen, Konstantinos Meichanetzidis, Bob Coecke
Quantinuum, 17 Beaumont Street, Oxford OX1 2NA, United Kingdom
{tuomas.laakkonen, k.mei, bob.coecke}@quantinuum.com

Quantum computing and AI have found a fruitful intersection in the field of natural language processing. We focus on the recently proposed DisCoCirc framework for natural language, and propose a quantum adaptation, QDisCoCirc. This is motivated by a compositional approach to rendering AI interpretable: the behavior of the whole can be understood in terms of the behavior of parts, and the way they are put together.

For the model-native primitive operation of text similarity, we derive quantum algorithms for fault-tolerant quantum computers to solve the task of question-answering within QDisCoCirc, and show that this is BQP-hard; note that we do not consider the complexity of question-answering in other natural language processing models. Assuming widely-held conjectures, implementing the proposed model classically would require super-polynomial resources. Therefore, it could provide a meaningful demonstration of the power of practical quantum processors.

The model construction builds on previous work in compositional quantum natural language processing. Word embeddings are encoded as parameterized quantum circuits, and compositionality here means that the quantum circuits compose according to the linguistic structure of the text. We outline a method for evaluating the model on near-term quantum processors, and elsewhere we report on a recent implementation of this on quantum hardware.

In addition, we adapt a quantum algorithm for the closest vector problem to obtain a Grover-like speedup in the fault-tolerant regime for our model. This provides an unconditional quadratic speedup over any classical algorithm in certain circumstances, which we will verify empirically in future work.

1 Introduction

Artificial intelligence permeates a wide range of areas of activity, from academia to industrial applications, with natural language processing (NLP) standing center stage. In parallel, quantum computing has seen a surge in development, and practical quantum processors are reaching scales where small-scale quantum algorithms are becoming feasible. The merging of these two fields has given rise to the young area of research on quantum natural language processing (QNLP). Here we focus on the 'first wave' of QNLP, starting around 2016 [14, 28, 37, 40, 54], although there are also other approaches [51, 52]. One important feature that distinguishes that first wave from other work, and from the broad field of quantum machine learning [6, 21, 46], is that it provides a path towards *explainability* and *interpretability*.

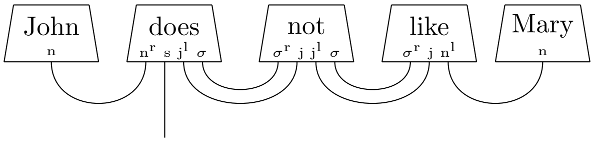
Despite all of the impressive advancements of contemporary artificial intelligence – which is synonymous with machine learning methods – these advancements have been achieved by training black-box deep-learning models on large amounts of data. In order to arrive at general applicability and high performance, what is often compromised by such setups, at least in terms of model architecture, is exactly explainability and interpretability. When things go wrong unexpectedly, you typically don't know why. Therefore, several attempts have been put forward recently to render black-box deep-learning models more explainable and interpretable. For example, methods for *post-hoc* mechanistic interpretability have recently formed an active area of research [9]. Conversely, first-wave QNLP made use of an earlier

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 162–196, doi:10.4204/EPTCS.406.8

quantum-inspired model for language [11, 16] that had been crafted precisely with the features of explainability and interpretability in mind. These features were achieved through *compositionality*.

For us, compositionality means that the behavior of the whole can be understood in terms of the behavior of parts, and the way they are put together [13]. In the case of language this includes linguistic meaning as well as linguistic structure such as grammar and coreference. An early compositional framework for natural language that combined linguistic meaning and linguistic structure was DisCoCat [16, 24, 29, 44]. Somewhat surprisingly, this was enabled by the foundational research program of categorical quantum mechanics [1, 15]: DisCoCat emerged from the observation that the category-theoretic structure of categorical quantum mechanics perfectly matched Lambek's model of linguistic structure in terms of pregroups [31, 32]. In this way, linguistic structure mediates how word-meanings interact in order to form phrase- and sentence-meanings – see [11, 14] for more details. Just as it is the case in categorical quantum mechanics, sentences in DisCoCat take the form of diagrams:

"John does not like Mary"



where the boxes capture word-meanings and the wires represent the grammatical interactions between those word-meanings, yielding the meaning of the sentence as a whole. The fact that DisCoCat essentially was derived from a quantum mechanical formalism, enabled one to use it to easily craft quantum NLP models, which we will refer to as QDisCoCat [14, 39, 54]. This was the foundation of the 'first wave' of QNLP.

QDisCoCat models have been implemented on quantum hardware for the task of sentence classification [28, 37, 40], but they have some shortcomings. Firstly, QDisCoCat diagrams have to be converted into quantum circuits in order to fit them on a quantum computer, and consequently, quantum algorithms derived from them require post-selection – no lower bounds are known for the post-selection probability, so these algorithms may require exponential time in the worst case.[1] A related issue is that the reliance of DisCoCat on 'categorial grammars' [30,31], which cannot natively represent texts larger than individual sentences. Both problems are addressed in the DisCoCirc framework [12, 49], where sentences are represented by circuits, and these circuits can be further composed to represent texts. We call these *text circuits*. This bypasses the need for conversion of diagrams into circuits, making post-selection unnecessary. Besides these, DisCoCirc enjoys a number other advantages that are discussed in [49], and has been studied within a wide variety of contexts [19, 43].

In this work, we propose QDisCoCirc, an adaptation of DisCoCirc which provides *efficient quantum algorithms* for several NLP tasks within the model. In particular, in Section 4 we show that there exists an NLP task, namely question-answering, which is BQP-hard to solve within the QDisCoCirc framework. Therefore, evaluating this model can provide a demonstration of the power of practical quantum processors, as simulating the model classically would require super-polynomial resources (assuming widely-held hardness conjectures). Furthermore, this would be less abstract compared to experiments such as those based on random circuit sampling [2, 18], which are convincing demonstrations of quantum hardware, but do not attempt to solve a widely applicable task. Importantly, we do not claim that the

[1] Sample-efficient sentence-level QNLP models can be constructed [25], but they are not compositional in the above sense.

underlying NLP task of question-answering cannot be solved efficiently with a classical machine, only that the QDisCoCirc model in particular would be hard to evaluate in this case.

We believe that QDisCoCirc models are worth considering because of the interpretability and explainability they afford. However, other concrete implementations of the DisCoCirc framework could also be proposed – for example, based on classical neural networks – and in any quantum model, it is necessary to justify why it should be preferred over a classical model, given the extreme difficulty of constructing quantum devices. We argue that a quantum model is most natural for this framework. DisCoCirc, via DisCoCat, is ultimately derived from pregroup grammars [32], where the composition of spaces is their tensor product. This has been argued as essential to fully capture the semantics of language [16, 24]. In fact, we have the following result:

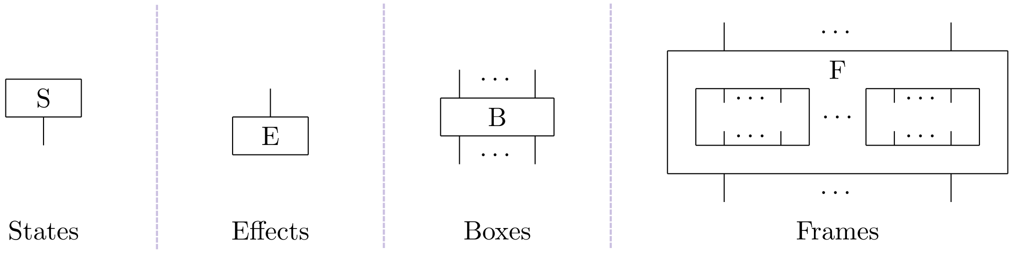
Proposition (Informal, [17, Theorem 2.1] & [15, Theorem 5.49])**.** *If a model has the same abstract structure as pregroup grammars, composition of spaces must be analogous to the tensor product.*

This can be made precise in several ways, as discussed in [15, 17]. This fixes the compositional structure of our model, but leaves the underlying Hilbert space undefined. As is customary in machine learning, we take this to be numerical vectors with the usual dot product, and in this case these models become tensor networks. Since large general tensor networks are not feasible to evaluate, both for quantum and classical devices, we must pick a subset which can be efficiently evaluated. We choose to base our model on quantum circuits, which represent the largest such subset currently known [7].

Finally, in Section 5 we give an algorithm that achieves for QDisCoCirc a Grover-like quadratic speedup over any classical algorithm for tasks such as text similarity, under certain technical assumptions about the dataset which can be verified empirically. This algorithm improves over the equivalent algorithm for QDisCoCat [54], which considers a technological regime where fault-tolerant scalable quantum computers and large quantum random access memories (QRAM) are available, while this is unlikely to be feasible in the near term. In contrast, our algorithm uses exponentially fewer bits of QRAM. Additionally, in Section 3 we present algorithms that do not use a QRAM, but are quadratically slower than the algorithm given in Section 5. They require much fewer quantum resources, and so are practical to implement on near-term machines. In Section 3, we show how text circuits can be mapped into parameterized quantum circuits to realize these algorithms.

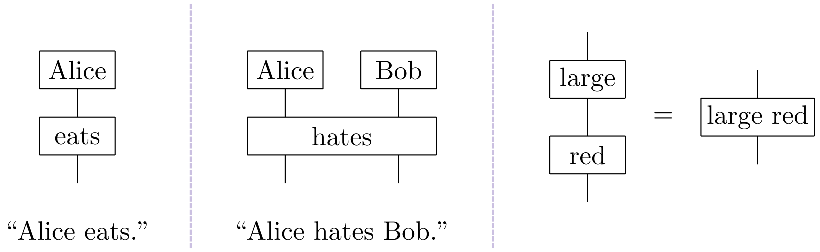
2 DisCoCirc

In the DisCoCirc framework, texts are the result of the composition of sentences, analogously to how sentences are compositions of words. Texts are represented by text circuits, which are read top to bottom. Boxes represent processes, and every box has input wires and output wires. In such circuits, only connectivity matters and so the information flow is explicit in the circuit. A circuit is composed of *generators*, or atomic processes, each of which has a linguistic interpretation – see [35] for more information. Text circuits are composed of the following set of generators, which we call *states*, *effects*, *boxes*, and *frames*:



Generators can be composed – sequential composition of processes can be done by connecting the output wires of one process with the input wires of another process, and parallel composition is performed by placing boxes side by side. An effect is to be understood as a 'test', or the inverse, of a state. Applying an effect to a state, by joining all of their wires, results in a circuit with no open wires, which we call a *scalar*.

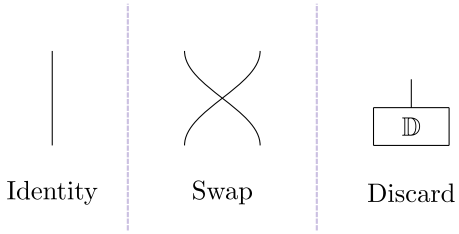
In this approach to modeling text as circuits, nouns are 'first class citizens', and are represented as states. The idea is that noun states are carried along wires, and so all wires here carry the same type, the noun type. Boxes are processes that transform states. Examples of these are adjectives or intransitive and transitive verbs, that act on nouns and pairs of nouns respectively. In general, generators with compatible shapes can be serially composed:



"Alice eats."

"Alice hates Bob."

We also have some special generators – the identity, the swap, and the discard effect:



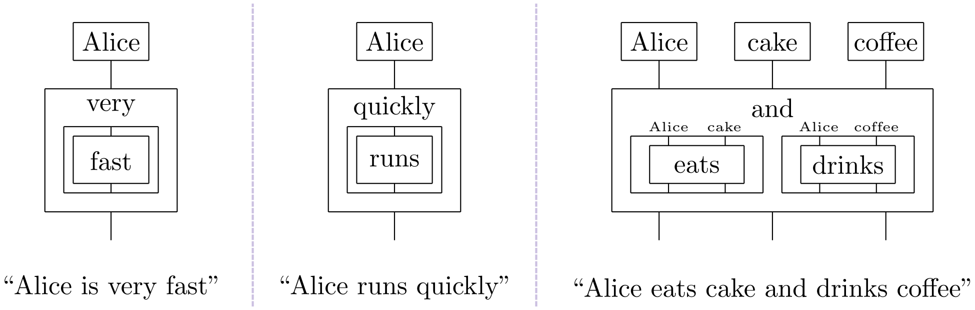
Identity

Swap

Discard

These interact with each other in a well-behaved way [15], so that *only connectivity matters* – we can bend and move wires freely so that equality of circuits is considered up to connectivity between inputs and outputs of generators. We consider any circuit to be state, effect, box, or frame if it has a compatible pattern of input and output wires. Additionally, we associate each generator with an inverse, which when composed serially yields the identity. Inverses for composite circuits are constructed by mirroring the whole circuit vertically and replacing every individual generator with its inverse.

Frames are super-maps that transform boxes, for example, intensifiers that act on adjectives, adverbs that act on verbs, or conjunctions that act on phrases:

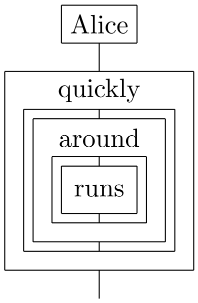


"Alice is very fast"

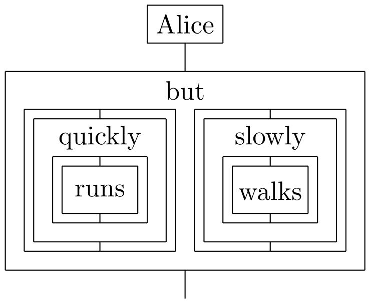
"Alice runs quickly"

"Alice eats cake and drinks coffee"

Note that not all wires may occur in all the holes in each frame. In this case, the interior wires can be labeled with exterior wires to which they correspond (note that this need not be a single wire). Frames can also be serially composed:



“Alice quickly runs around”



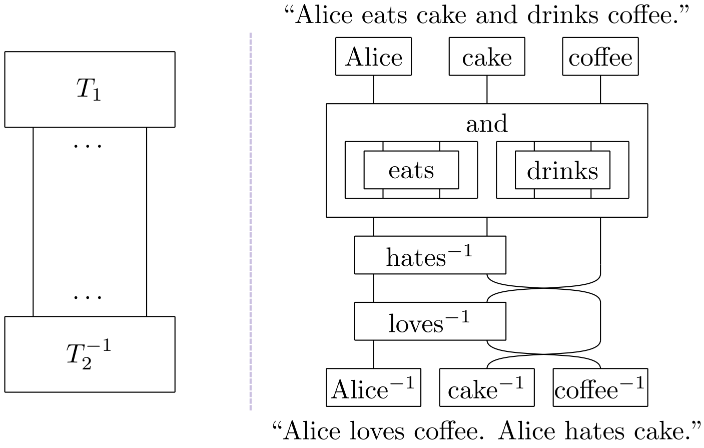
“Alice runs quickly but walks slowly”

The composition of the generators of text circuits is not free; not every composition is allowed – for instance, wires may not pass through frames. Compositions are restricted in the sense that text circuit are the output of a text-to-circuit *parser* [35], which for this work we assume as given – the parser generates parse trees for each sentence that are converted to text circuits, and the wires corresponding to common nouns are connected using coreference resolution so that generators concerning the same nouns throughout the text act on the same wires in the circuit. Arbitrary text circuits are *local*, as the number of input and output wires of every box is upper bounded by a constant that is independent of the text size. Furthermore, we assume that text circuits derived from a text via the parser are acyclic.

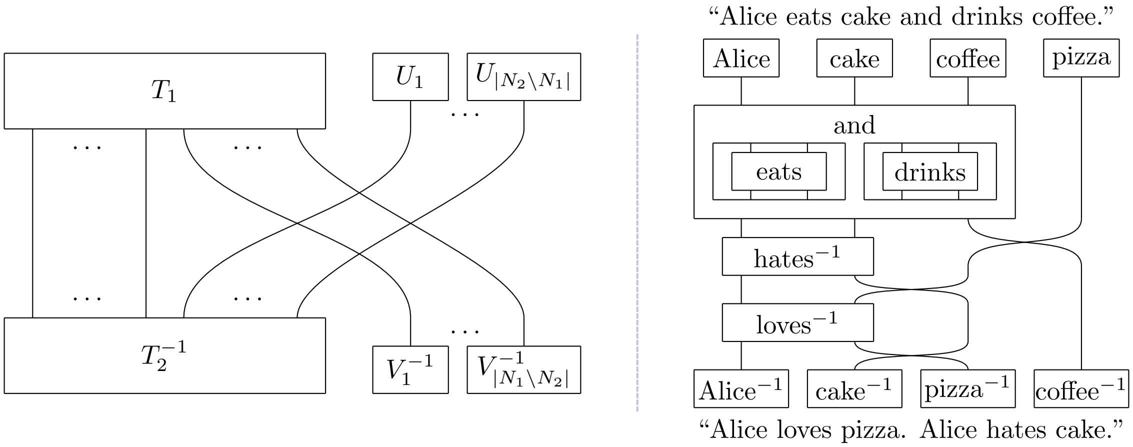
2.1 Native NLP Tasks

The range of NLP tasks is wide and diverse. Here, we consider tasks that are *native* to DisCoCirc. That is, we formulate simple NLP tasks using the compositional structure of the model. The key idea is that text circuits constructed by the parser pipeline are states which carry the information of the text. Then this information is retrieved by constructing appropriate *tests* in terms of effects. In general, the effects are constructed as (inverses of) text circuits themselves. The primitive operation that we will use to define NLP tasks is that of *similarity*, or overlap, which is tested by serially composing effects onto states. In this way, we can represent each task as a single, scalar, text circuit.

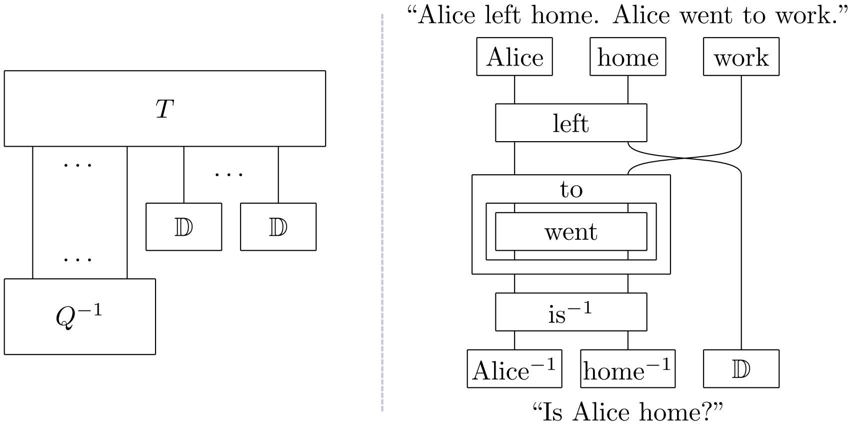
First we consider the text-level analogue of sentence similarity as presented in the DisCoCat framework [54], ie *text similarity*. Suppose T_1 and T_2 are text circuits defining states obtained from the parser, defined on the wires generated by same set of nouns N. We form a scalar text circuit corresponding to their similarity by serially composing one with the inverse of the other. For example:



In the case that the circuits are not defined on the same set of nouns, we can still compute the similarity. Suppose T_1 and T_2 be defined on the sets of nouns N_1 and N_2. Let $U_1, U_2, \ldots, U_{|N_2 \setminus N_1|}$ be the noun states present in T_2 but not T_1, and similarly $V_1, V_2, \ldots, V_{|N_1 \setminus N_2|}$ be the noun states present in T_1 but not T_2. By parallel composing T_1 with U_i and T_2 with V_i, we form two text circuits that are defined on the same wires. For example:



While this task compares the whole of both texts, we can extract more specific information about a text by only comparing against a subset of the nouns that we wish to know about. We call this task *question answering*. In this task, we want to ask a question about a subset of the nouns in the text. To do this, we formulate the question as a statement (for example, ‘Is Alice home?’ becomes ‘Alice is home.’) and then we compute the similarity of the text with the inverse of the question in the same way as above, except that we discard the noun wires in the text that are not in the subset used by the question. For example:



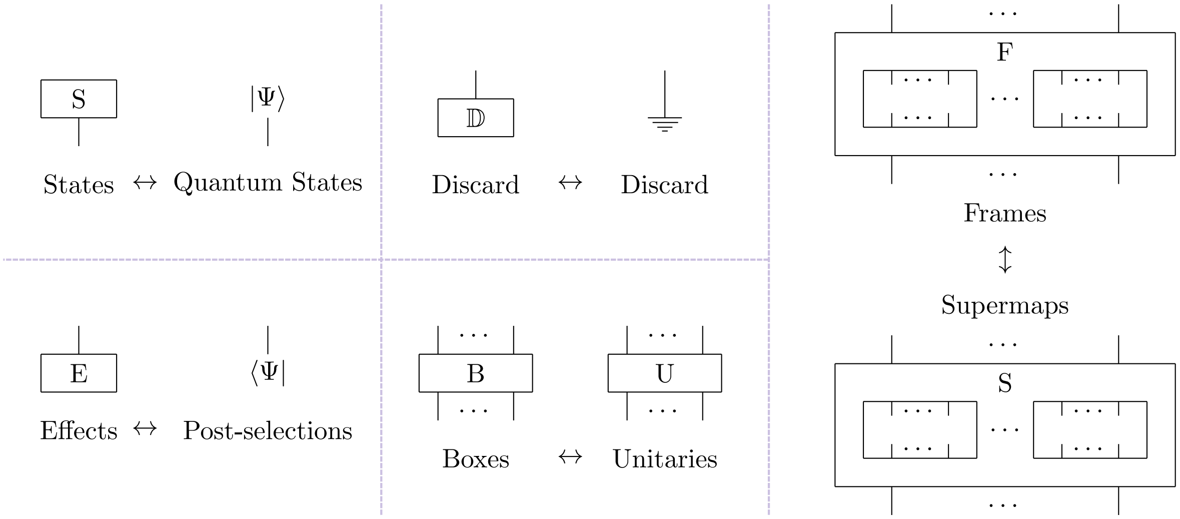
In the case of questions which have several different possible answers, we formulate such a question statement and thus text circuit for each possible answer. For example, 'Is Alice home?' becomes a pair of question statements 'Alice is home.' and 'Alice is not home.' Then we compare the scalar text circuits formed in each case to determine the answer to the question.

In Section 6, we show a collection of other model-native NLP tasks that enjoy a natural representation as text circuits. In general, DisCoCirc itself does not prescribe a particular semantics for any of these tasks (for instance, we leave the way that scalar circuits are computed or compared unspecified) — it provides a method for constructing the text circuits representing the input texts, as well as an intuitive framework for designing circuits corresponding to a particular task. Moreover, once an implementation of the model is decided, the word-embeddings that form the boxes will be derived to solve a specific task. The aim of DisCoCirc is only to incorporate linguistic structure into the model, so it is not imperative that these task-circuits are defined rigorously (for example, as a statement of discourse representation theory), but rather that they are 'intuitively correct' so that the imparted structure is helpful in practice. In the next section, we will construct a specific implementation of DisCoCirc that fills in these details.

3 QDisCoCirc Models

To construct a concrete DisCoCirc model that performs a model-native NLP task for any potential text and task, we apply a structure-preserving map to the corresponding text circuit. This is achieved by defining a map that is applied to every generator individually, in a way that respects the relations defined on the generators. As discussed in the introduction, a natural choice of space in which such circuits may exist is tensor networks. As tensor networks are hard to evaluate in general, we propose QDisCoCirc, a family of models based on quantum circuits, which are a restricted family of tensor networks, and can be efficiently evaluated on a quantum computer.

In QDisCoCirc, text circuits are translated to quantum circuits with postselections by replacing each generator with a quantum operation or state and each wire with some fixed number of qubits. Furthermore, the inverses of generators are given by their adjoints. The generators are mapped as follows:



The specific form of supermaps obtained via this mapping will be discussed shortly. We identify post-selected quantum circuit diagrams with the density matrices they represent (which are not normalized in general); thus, scalar text circuits are indeed transformed to scalars representing the probability of successfully post-selecting the given circuit.

To solve the model-native tasks defined above using this model, we observe that all the scalar text circuits for the tasks take the form of A serially composed with B^{-1} for some circuits A and B (this is true also for the tasks presented in Section 6). This corresponds to a text similarity operation. Through the quantum semantic functor, this operation maps to quantum state overlap. According to the Born rule, these result to expressions of the form $F(\Psi,\Phi) = |\langle\Psi \mid \Phi\rangle|^2$ for some quantum states Ψ and Φ in the pure case, and in the general case $F(\rho,\sigma) = \operatorname{tr}(\rho\sigma)$ for density matrices ρ and σ.

However, in the case that both ρ and σ are mixed, this does not satisfy the properties we would expect from a similarity operation, such as $F(\rho,\rho) = 1$ and $F(\rho,\sigma) < 1$ if $\rho \neq \sigma$. There are a few quantities [33] such as the Uhlmann-Josza fidelity $\operatorname{tr}\left(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2$ which do satisfy these properties for mixed states, but unfortunately they cannot be computed efficiently using a quantum computer. Therefore, we will ensure text circuits obtained from the parser pipeline are translated to pure quantum states, so that for the tasks defined previously, *at most one* of the states representing A and B is mixed.

Since the parser does not generate discards, we just need to make sure none of the generators are mapped to mixed operations. This is trivially true for states, effects, and boxes, but frames are more problematic. Since states map to quantum states and boxes to unitaries, frames become maps between unitaries – supermaps – by analogy. However, some frames in text circuits may contain more wires for one of their arguments than are input and output to the frame – this arises in many cases, for instance in reflexive structures. Furthermore, general quantum supermaps can be written as pairs of unitaries conjugating their arguments, along with a side channel consisting of some auxiliary qubits. In either case, we cannot use auxiliary qubits or wires because these would then be subsequently discarded, making the state mixed. Therefore, we instead constrain define our mapping from text circuits such that these situations do not occur, see Appendix A for details.

3.1 Solving Model-Native Tasks

After decomposing the frames in a text circuit, the resulting QDisCoCirc quantum circuit will contain only quantum states and unitaries labeled with the words from which they originated. For each task, we assume we are given these states and unitaries as quantum circuits, which we call *quantum word embeddings*, such that the circuits defined for the task are evaluated as expected. We will consider in future work how to pick word embeddings such that this is the case. In particular, for text-similarity tasks, that the degree of semantic similarity between the two texts is indicated by $F(\rho,\sigma) = \mathrm{tr}(\rho\sigma)$. We can solve tasks based on this primitive using the *swap test* [10]:

Theorem 1. *The text-similarity primitive of QDisCoCirc models between two text circuits A and B can be approximated to precision ε in polynomial time on a quantum computer. In particular, given $\varepsilon > 0$ and ρ, σ the quantum states derived from A and B respectively, the quantity $\mathrm{tr}(\rho\sigma)$ can be determined to precision ε with failure probability δ by a quantum circuit of size*

$$O\left(\frac{(|A|+|B|)\log(\frac{1}{\delta})}{\varepsilon^2}\right)$$

where $|A|$ and $|B|$ are the number of elementary quantum gates comprising ρ and σ.

Proof. We construct a quantum circuit based on the swap test [10, Figure 1]. This gives a circuit of size $O(|A|+|B|)$ for which the first qubit has measurement probability $\frac{1}{2}+\frac{\mathrm{tr}(\rho\sigma)}{2}$. By a Hoeffding bound, $O(\log(\frac{1}{\delta})\varepsilon^{-2})$ samples of this circuit suffice to estimate $\mathrm{tr}(\rho\sigma)$ to precision ε with failure probability δ. □

To solve the question-answering task in the QDisCoCirc model, we are given a text T and a set of questions Q_i and wish to find the Q_i which maximizes $\mathrm{tr}(\rho_T,\rho_{Q_i})$. We can do this by evaluating $\mathrm{tr}(\rho_T,\rho_{Q_i})$ for each i using the method above, and then taking the maximum of these. We show below that this also runs in polynomial time on a quantum computer. To compare this algorithm to classical algorithms, it is important to consider the scaling of ε carefully. The best-known classical method to calculate $\mathrm{tr}(\rho\sigma)$ would at least require performing statevector simulation of the quantum circuit. In this case, we would expect a total time of at least $O((|A|+|B|)N^n\,\mathrm{polylog}(\varepsilon^{-1}))$ and memory requirement of $O(N^n\log(\varepsilon^{-1}))$, where $\log(N)$ is the number of qubits per wire, and n is the number of wires in the text circuits – this is an exponentially worse dependency on N and n.

On the other hand, the scaling in terms of ε is much better in the classical case. For instance, consider question answering. If the number of nouns referenced in the question is constant c, then only $c\log(N)$ qubits will not be discarded. In this case, if the quantum states corresponding to the questions are sufficiently uniformly distributed then $\varepsilon = O(N^{-c})$ should suffice to resolve the maximum similarity, by an argument of Wiebe et al [53], which gives exponentially better dependence in n. However, considering the case of full text-text similarity with no discarding, the quantities $\mathrm{tr}(\rho\sigma)$ may grow as small as N^{-n} in general, and so if small relative error is needed, $\varepsilon = O(N^{-n})$ may be required, which makes the complexity at least as bad as the classical algorithm. Therefore, for any given dataset and word embeddings, it would be important to analyze the scaling of ε in terms of N and n, either analytically or empirically.

4 Hardness for Question Answering

We will now consider the problem of question answering in the QDisCoCirc framework more formally. In particular, we will show that for the appropriate choices of texts and word embeddings, question

Texts Embeddings	*Worst-Case*	*Average-Case*	*Typical*
Worst-Case	BQP-hard (Thm.3)	BQP-hard (Thm.6)	Empirical only
Average-Case	BQP-hard (Thm.4)	Unknown	Empirical only
Typical	Testable (Thm.5)	Unknown	Empirical only

Table 1: The hardness results shown in this paper. The embeddings axis refers to the choice of word embeddings V, whereas the texts axis refers to the choice of context T and questions $\{Q_i\}$. 'Average-case' hardness refers to a 'natural' distribution - for embeddings this is the Haar distribution, whereas for texts we define a distribution. 'Typical' hardness refers to texts and embeddings taken from actual human-generated data. Since typical texts are always bounded by definition, only empirical evidence can be given for hardness in this case.

answering is BQP-hard, and that there is an efficient quantum algorithm to solve it. Note these results only apply to question answering strictly as specified by the QDisCoCirc framework, not as a general NLP task. We call this task QDisCoCirc-QA to emphasize the distinction. Proofs for the theorems in this section are given in Appendix B.

Definition 1. *The problem* QDisCoCirc-QA *is defined as follows: given a set of word embeddings V, a context text T, and a set of k question texts $\{Q_i\}$, determine any j such that*

$$\left| \operatorname{tr}\left(\rho_T(\rho_{Q_j} \otimes I)\right) - \max_i \operatorname{tr}\left(\rho_T(\rho_{Q_i} \otimes I)\right) \right| < \varepsilon$$

where $\rho_T = U_T|0\rangle\langle 0|U_T^\dagger$, $\rho_{Q_i} = U_{Q_i}|0\rangle\langle 0|U_{Q_i}^\dagger$, and U_T, U_{Q_i} are the QDisCoCirc text circuits generated from T and Q_i respectively over V.

Theorem 2. QDisCoCirc-QA *can be solved on a quantum computer in time*

$$O\left(\frac{k \log\left(\frac{k}{\delta}\right) |V|_w(|T| + \max_i |Q_i|)}{\varepsilon^2}\right)$$

with precision ε and failure probability δ, where $|V|_w$ is the maximum number of elementary gates of any word embedding in V, and $|T|$, $|Q_i|$ are the number of elementary gates in U_T and U_{Q_i} respectively.

Note that if the separation between the values $\operatorname{tr}(\rho_T(\rho_{Q_i} \otimes I))$ is smaller than ε, then the formulation of QDisCoCirc-QA does not guarantee that the 'correct' answer (in the sense of maximum similarity) will be returned. We assume that word embeddings exist such that ε need not be too small, as discussed in Section 3. Now we tackle the hardness results in the first column of Table 1 - that is, we will show that for some sets of word embeddings V it is possible to construct context texts T and questions $\{Q_i\}$ for which QDisCoCirc-QA is BQP-hard.

Theorem 3. *Suppose that a set of word embeddings V satisfies the following:*

1. *The operations of V use one qubit for each input wire,*
2. *V contains arbitrarily many proper nouns,*
3. *V contains at least two adjectives that generate a dense subset of $SU(2)$,*
4. *V contains at least one transitive verb that is entangling*

then for any fixed $\varepsilon < \frac{1}{7}$, QDisCoCirc-QA *is BQP-hard.*

Supposing that we have word embeddings with enough nouns (and thus enough qubits) to perform the reduction in the previous theorem, the particular unitaries in the word embeddings aren't important – Haar-random word embeddings suffice with high probability. This shows that QDisCoCirc-QA is hard in the average-case over word embeddings.

Theorem 4. *Given a set of word embeddings V, suppose that operations in V are independent Haar-random unitaries. Then conditions three and four of Theorem 3 are almost surely satisfied for all word embeddings containing at least two adjectives and one transitive verb.*

Moreover, suppose we are given a set of word embeddings (for instance, that can solve a particular task), then we give a method to test if QDisCoCirc-QA is hard in this case.

Theorem 5. *Given a specific set of word embeddings V it is possible to check numerically that the conditions of Theorem 3 are satisfied.*

To show average-case hardness of QDisCoCirc-QA in terms of texts (second column in Table 1), we will first define a distribution over QDisCoCirc circuits of a given size, and then adapt Theorem 3 to show that we can draw polynomially sized texts T, $\{Q_i\}$ from this distribution and construct a set of word embeddings V such that solving the corresponding question answering instance is BQP-hard. Note that this distribution is not particularly representative of human-generated texts.

Definition 2. *For every size k, we define a distribution $\mathscr{D}_k$ of text circuits with k unique states:*

- *There are no frames, and there are $\Omega(k^\gamma)$ boxes with high probability for some $\gamma > 4$.*
- *The label of each box is drawn independently from a distribution of words that obey Heap's law and Zipf's law exactly.*
- *The number of inputs to each box is drawn independently from a distribution A such that $P(A = 2)$ is nonzero and each input to a box is selected uniformly randomly from all possible wires.*

We are now ready to show the reduction. However, to make this work we need the number of words that only occur once in a given text (*hapax legomena*) to be bounded below by a polynomial in the total size of the text. Clearly, this cannot hold for all sizes for any finite vocabulary. For typically-sized texts, this assumption is justified statistically by Heap's law and Zipf's law [34, 45] — it has been shown that for large corpora, roughly half of the vocabulary are hapax legomena, and hence their number scales like $\propto N^k$ for some $k \approx 0.5$ [3]. However, since we are considering the limit of arbitrarily-sized texts which do not exist in practice (all human texts are bounded), we require that this behavior can be extrapolated.

Theorem 6. *There exists a polynomial $f(m,n)$, such that given an oracle to solve* QDisCoCirc-QA *instances with arbitrary word embeddings and text circuits drawn from $\mathscr{D}_{f(m,n)}$, we can perform arbitrary quantum computations with m 2-local gates on n qubits with high probability in polynomial time. That is,* QDisCoCirc-QA *for worst-case word embeddings is average-case BQP-hard over texts drawn from $\mathscr{D}_{f(m,n)}$.*

Since QDisCoCirc-QA is BQP-hard, we expect that it is not easy to solve with a classical computer, and that quantum algorithms to solve this ought to provide super-polynomial speedups. Therefore, supposing that (a) there exists a set of word embeddings for which a QDisCoCirc model is accurate on a dataset of interest, (b) the resulting quantum circuits are not easy to simulate classically (in the sense of Theorem 5), (c) a quantum device exists with sufficient fidelity to provide meaningful results, and (d) the precision ε required is not too high (in the sense of Section 3.1), then it would be possible to perform an experiment on a quantum device which is hard to simulate classically but solves a task of interest. However, it is important to note that for many datasets this task can be solved classically by other means (notwithstanding the arguments presented in favor of QDisCoCirc in Section 1), so this would *not* constitute a demonstration of quantum advantage over *every* classical algorithm.

5 Further Quadratic Speedups

In this section, we present a quantum algorithm for obtaining polynomial speedups for the task of text question answering that we presented in Sections 3 and 4. Specifically, our results are obtained by adapting and improving on the approach of the work of Zeng and Coecke [54]. These algorithms provide a quadratic improvement over the simpler swap test algorithm, as well as being quadratically faster than any classical algorithm for solving the same task. While they require a QRAM [23], it is used to store exponentially fewer bits as compared to [54].

Basheer et al [5] give a quantum algorithm for accelerating the k-nearest neighbors problem as follows. Given the vectors $v_0 \in \mathbb{C}^N$ and $\{v_1, \dots, v_k\} \subset \mathbb{C}^N$ such that $||v_i|| = 1$ and quantum amplitude encoding oracles $\mathscr{V}|0\rangle = |v_0\rangle$ and $\mathscr{W}|i\rangle|0\rangle = |i\rangle|v_i\rangle$ with gate complexities T_V and T_W, then there is a quantum algorithm to find $i = \arg\max_{j\geq 1} |\langle v_0 \mid v_j\rangle|$ with error at most ε and fixed positive success probability in time $\tilde{O}\left(\sqrt{k}\varepsilon^{-1}(T_V + T_W + \log^2 N)\right)$. The algorithm takes quantum states, encodes the fidelity via a partial swap test, then digitizes this with the quantum analog-to-digital conversion algorithm of Mitarai et al [41] before finding the minimum via Dürr-Høyer search [22]. In the case that $\mathscr{V}(|0\rangle\langle 0|) = \rho_0$ and $\mathscr{W}(|i\rangle\langle i| \otimes |0\rangle\langle 0|) = |i\rangle\langle i| \otimes \rho_i$ are actually operators preparing mixed states, then the quantity calculated will be $i = \arg\max_{j\geq 1} \operatorname{tr}(\rho_0\rho_i)$.

5.1 Application to DisCoCirc

Since DisCoCirc is based on a circuit architecture, it is natural to work with the *word embeddings* described as *quantum circuits*, and construct the amplitude encoding of texts directly as quantum circuits. We will apply the nearest-neighbor algorithm to the following task: given a text T_0 and a set of texts T_i for $k \geq i > 0$, find i such that T_0 and T_i are the most similar, and thus that maximizes $\operatorname{tr}(\rho_0\rho_i)$, where ρ_i is the density matrix corresponding to the QDisCoCirc circuit for T_i. This extends both the question-answering and text-similarity tasks given in Section 3.

Unlike the previous section where we assumed that the boxes in QDisCoCirc were arbitrary unitaries, we now suppose that each box with d wires is a particular instance of a parameterized quantum circuit ansatz. Let us define the following hyperparameters: (a) each noun wire is mapped to a set of qubits with Hilbert space dimension N, (b) each QDisCoCirc circuit considered has at most w unitaries and n noun wires total, (c) each unitary in the quantum circuit acts on at most d wires, (d) there are V different word embeddings given, (e) the number of gates in the ansatz for d wires of dimension N scales as $A_g(N,d)$, (f) the number of parameters in the ansatz for d wires of dimension N scales as $A_p(N,d)$, and (g) each parameter for the ansatz is stored with P bits of precision.

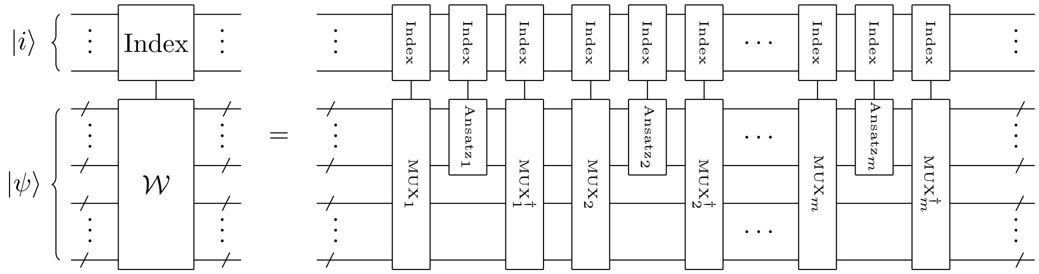
To solve this task, we can straightforwardly apply the quantum k-nearest neighbors algorithm discussed above, with $\mathscr{V}(|0\rangle\langle 0|) = \rho_0$ and $\mathscr{W}(|i\rangle\langle i| \otimes |0\rangle\langle 0|) = |i\rangle\langle i| \otimes \rho_i$. Clearly, $\mathscr{V}$ is given by the quantum circuit preparing ρ_0, however for $\mathscr{W}$ we need a significantly more complicated circuit. This is explicitly constructed in Appendix C, and requires the following QRAM oracles:

- $\text{Param}_{ij}|k\rangle|0\rangle = |k\rangle|\phi_{ijk}\rangle$ where ϕ_{ijk} is a P-bit fixed-precision binary encoding of the jth parameter for the ansatz of the ith unitary in the circuit preparing ρ_k,
- $\text{Width}_i|k\rangle|0\rangle = |k\rangle|W_{ik}\rangle$ where W_{ik} is a binary encoding of the width of the ith unitary in the circuit preparing ρ_k,
- $\text{Index}_{ij}|k\rangle|0\rangle = |k\rangle|I_{ijk}\rangle$ where I_{ijk} is an encoding of the position of the jth argument to unitary i in the circuit preparing ρ_k. Note that the encoding is not a mapping from zero-indexed position to binary, but it is of size $O(\log(w))$ – see Appendix C.

Each of these oracles can be implemented using a series of calls (one for each bit) to bucket-brigade QRAMs of size k. Therefore, $O(kw(PA_p(N,d)+\log(d)+d\log(w)))$ bits of QRAM is required in total, each taking time $O(\log(k))$ to query.

If we assume that $A_p(N,d) = O(\text{poly}(\log(N),d))$ (this is true for any of the widely-used ansatze defined by Sim et al [47], for example), that $w = O(\text{poly}(n))$ (which is justified by Heap's law), and that P and d are fixed, then this is $O(k\text{poly}(n,\log(N)))$ bits total. By comparison, adapting the strategy employed in [54] to QDisCoCirc would require at most $O(kN^n)$ bits total, so our approach provides a significant saving.

The oracle for $\mathscr{W}$ is constructed by alternating layers of multiplexers, which bring the qubits for each unitary to the top of the circuit, and parameterized ansatze to implement each unitary. Each of these load parameters from QRAM to specify their behavior. The overall construction is as follows:



Using this, we derive the following result:

Theorem 7. *Given a text T_0 and a set of texts T_i for $k \geq i > 0$, we can find i that maximizes* $\operatorname{tr}(\rho_0\rho_i)$*, with error at most ε, in time*

$$\tilde{O}\left(\frac{\sqrt{k}}{\varepsilon}(wd\log(k)(n\log(N)+PA_g(N,d))+\log^2(N))\right)$$

on a quantum computer with a constant success probability, where ρ_i is the density matrix corresponding to the QDisCoCirc quantum circuit for T_i.

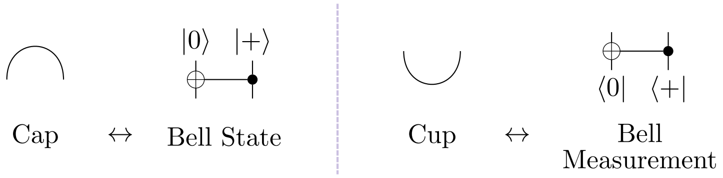
Proof. We apply the quantum k-nearest neighbor algorithm of Basheer et al [5] with $\mathscr{V}$ and $\mathscr{W}$ defined above. These can be implemented with $O(A_g(N,d))$ and $O(wd\log(k)(n\log(N)+PA_g(N,d)))$ gates respectively, see Appendix C for details. We require controlled versions of these, but this only increases the gate count by a constant factor. □

Comparing this to the naive swap test-based algorithm for solving the question-answering task defined in Section 4, it is clear that this method achieves a quadratic speedup both in k and ε, up to extra logarithmic factors needed to construct the oracles. Comparing against a classical algorithm, note that for any given ε, this algorithm is better than any classical algorithm, which clearly must be at least $O(k)$, and so for large values of k we would expect to see a speedup as compared to any classical algorithm to solve the same task. However, as discussed in Section 3, the classical algorithm scales much better in terms of ε, namely $O(\log(\frac{1}{\varepsilon}))$ vs $O(\varepsilon^{-1})$, and so the same concerns about the dependence on N and n apply here. Additionally for this algorithm, for a given dataset, it is necessary to analyze the scaling of ε in terms of k – for instance, if $\varepsilon = o(k^{-\frac{1}{2}})$ is required, then there would be no quadratic speedup in k compared to the classical algorithm.

6 Other Model-Native Tasks

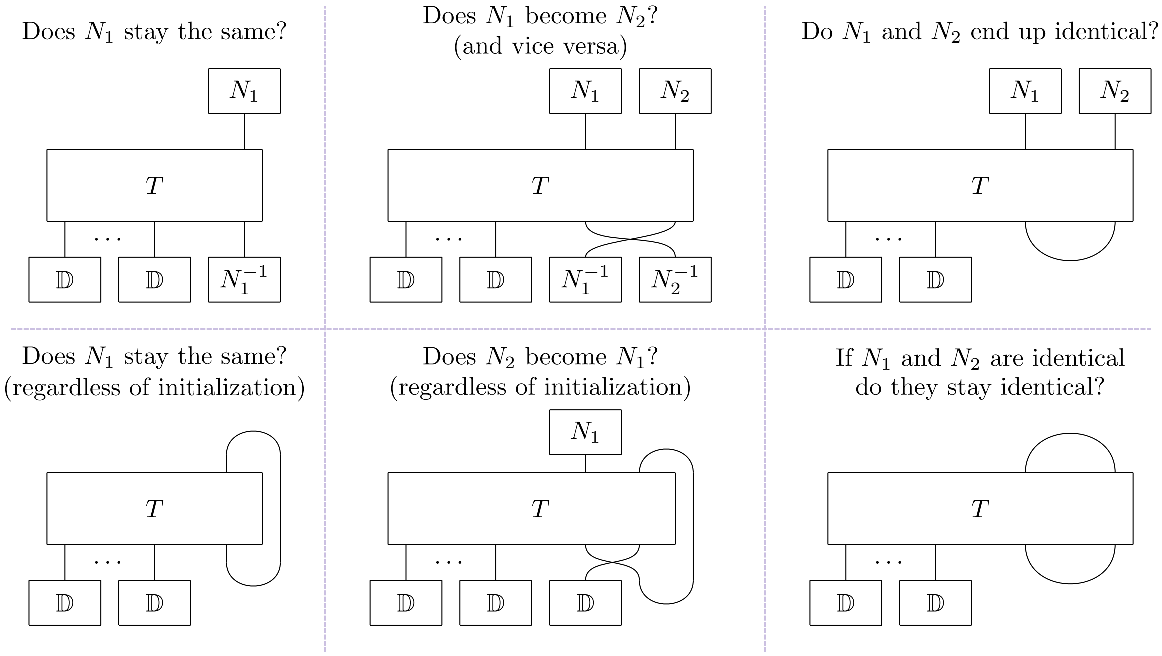
In the previous sections, we have considered only the fundamental primitive of text-similarity, and its local version that we call question-answering. However, we can also propose some simple model-native text processing tasks regarding ‘character development’. All tasks we introduce here have a natural representation as text circuits in the DisCoCirc framework.

To do so, we introduce two new generators: the cap and cup. The cap semantically represents a joint state of two nouns which are unspecified but identical. Likewise, the cup is the corresponding effect that tests for this situation.



Their translations into QDisCoCirc are naturally defined as the Bell state and measurement respectively. These generators are both self-inverse, and obey certain relations along with the identity and swap generators to ensure that they are well-behaved [15] - in particular, to maintain the property that only connectivity of the diagram matters.

We can use these to construct several tasks. For example, measuring a ‘character arc’ — comparing the similarity of a noun state wire before and after it has been acted on by a text, either with itself or between a pair of nouns. We can also form a diagrammatic trace using caps, cups, and identity wires, which we can use to measure a similar quantity in a way that is independent of the initial noun state (i.e. how much *any* noun changes if it was taking that particular role in the text). Let T be a text and N_1 and N_2 be two nouns present in it, then we can form the following text circuits:



Does N_1 stay the same?
Does N_1 become N_2? (and vice versa)
Do N_1 and N_2 end up identical?
Does N_1 stay the same? (regardless of initialization)
Does N_2 become N_1? (regardless of initialization)
If N_1 and N_2 are identical do they stay identical?
N_1
N_2
T
$\mathbb{D}$
N_1^{-1}
N_2^{-1}
$\cdots$

Further, we can have combinations of the above, by applying noun effects and traces in the same circuit. This is to be understood as testing for properties in conjunction. From this, we see the motivation for including cups, caps, and inverses in our framework – it allows us to represent more intuitively some natural classes of tasks as a single text circuit. Additionally, while the results in Sections 4 and 5 are tailored to the sentence-similarity and question-answering tasks, we would expect that similar theorems hold for all the tasks defined here as well.

7 Discussion and Future Work

In this paper we focused on the DisCoCirc framework because of its explainability and interpretability in compositional terms. For the primitive task of question-answering within the QDisCoCirc model, we showed BQP-hardness. Further, we adapted known quantum algorithms to achieve for QDisCoCirc a Grover-like quadratic speedup over any classical algorithm for tasks such as text similarity, under certain technical conditions.

Recalling that the way in which we constructed QDisCoCirc is by applying a map from text circuits to quantum circuits, other variants of this mapping could be explored in the future. For example, there are families of quantum circuits that are *classically efficiently simulable*, such as those constructed from nearest-neighbor match-gates [27], which have been used successfully to numerically optimize large quantum circuits [38] and could be used as 'warm start' initializations for QDisCoCirc models. Viewing quantum circuits as special cases of tensor networks, we can also consider models based on tensor networks (although they would be costly to compute). This would constitute compositional tensor-network machine learning, which is an emerging subfield in machine learning [26, 48].

Overall, our approach can be viewed as providing sparse structural priors to NLP models, potentially aiding generalization performance or allowing more efficient training. Such an approach could also provide a structure that may guide how mechanistic interpretability methods can be applied [36]. Most importantly, we prioritized transparent model-building, with performance and broad applicability as a secondary target.

The results here are presented in a way compatible with the parser introduced by Liu, Shaikh, Rodatz, Yeung, and Coecke [35], which mechanically transforms a subset of English text into DisCoCirc text circuits. Given a different parser, the definition of the mapping from text circuits to quantum circuits might need modification. However, the results of Section 4 and 5 regarding quantum algorithms would still be applicable. These results concern fault-tolerant quantum computers (either via deep quantum circuits, or by the assumption of quantum random access memories).

Focusing on more near-term hardware, and given access to a parser and state-of-the-art near-term quantum processors [4], allows us to perform heuristic explorations based on the algorithm in Section 3.1. In a separate paper [20], we report on performing tasks such as question-answering within the noisy intermediate-scale quantum regime. We train the quantum word embeddings that encode the semantics of texts in-task, as was done in previous work with DisCoCat [37, 40]. However, it is also promising to experiment with training these in a task-agnostic way.

Furthermore, QDisCoCirc allows us to *put to the test* the principle of *compositionality* with respect to training, i.e. to quantify to what extent a compositional model aids generalization to larger instances after being trained on instances up to a certain size. This setup is motivated by the possibility that a compositional approach to quantum machine learning with parameterized circuits bypasses the fundamental problem posed by barren plateaus [42]. Our results in [20] are very promising in this respect.

Acknowledgments We thank Matty Hoban for discussions on complexity theory, Harry Buhrman, Richie Yeung, Luuk Coopmans and Ilyas Khan for comments on the manuscript, the QDisCoCirc team

(Saskia Bruhn, Tiffany Duneau, Gabriel Matos, Anna Pearson, and Katerina Saiti) for helpful discussions refining the model, and the parser-pipeline team (Jonathon Liu, Razin Shaikh, and Benjamin Rodatz) for answering all of our questions.

References

[1] S. Abramsky & B. Coecke (2004): *A categorical semantics of quantum protocols*. In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS), pp. 415–425, doi:10.48550/arXiv.quant-ph/0402130. arXiv:quant-ph/0402130.

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven & John M. Martinis (2019): *Quantum Supremacy Using a Programmable Superconducting Processor*. Nature 574(7779), pp. 505–510, doi:10.1038/s41586-019-1666-5.

[3] R Harald Baayen (2001): *Word frequency distributions*. 18, Springer Science & Business Media, doi:10.1007/978-94-010-0844-0.

[4] Charles H. Baldwin, Karl Mayer, Natalie C. Brown, Ciarán Ryan-Anderson & David Hayes (2022): *Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations*. Quantum 6, p. 707, doi:10.22331/q-2022-05-09-707.

[5] Afrad Basheer, A. Afham & Sandeep K. Goyal (2021): *Quantum k-nearest neighbors algorithm*, doi:10.48550/arXiv.2003.09187. arXiv:2003.09187.

[6] Marcello Benedetti, Erika Lloyd, Stefan Sack & Mattia Fiorentini (2019): *Parameterized quantum circuits as machine learning models*. Quantum Science and Technology 4(4), p. 043001, doi:10.1088/2058-9565/ab4eb5.

[7] Ethan Bernstein & Umesh Vazirani (1997): *Quantum Complexity Theory*. SIAM Journal on Computing 26(5), pp. 1411–1473, doi:10.1145/167088.167097.

[8] Adam Bouland & Tudor Giurgica-Tiron (2021): *Efficient Universal Quantum Compilation: An Inverse-free Solovay-Kitaev Algorithm*, doi:10.48550/arXiv.2112.02040. arXiv:2112.02040.

[9] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan & Christopher Olah (2023): *Towards Monosemanticity: Decomposing Language Models With Dictionary Learning*. Transformer Circuits Thread. `https://transformer-circuits.pub/2023/monosemantic-features/index.html`.

[10] H. Buhrman, R. Cleve, J. Watrous & R. De Wolf (2001): *Quantum fingerprinting*. Physical Review Letters 87(16), p. 167902, doi:10.1103/PhysRevLett.87.167902.

[11] S. Clark, B. Coecke, E. Grefenstette, S. Pulman & M. Sadrzadeh (2014): *A quantum teleportation inspired algorithm produces sentence meaning from word meaning and grammatical structure*. Malaysian Journal of Mathematical Sciences 8, pp. 15–25, doi:10.48550/arXiv.1305.0556. ArXiv:1305.0556.

[12] B. Coecke (2019): *The mathematics of text structure*, doi:10.48550/arXiv.1904.03478. ArXiv:1904.03478.

[13] B. Coecke (2021): *Compositionality as we see it, everywhere around us*. arXiv preprint arXiv:2110.05327, doi:10.48550/arXiv.2110.05327.

[14] B. Coecke, G. de Felice, K. Meichanetzidis & A. Toumi (2020): *Foundations for Near-Term Quantum Natural Language Processing*, doi:10.48550/arXiv.2012.03755. ArXiv preprint arXiv:2012.03755.

[15] B. Coecke & A. Kissinger (2017): *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.

[16] B. Coecke, M. Sadrzadeh & S. Clark (2010): *Mathematical foundations for a compositional distributional model of meaning*. In J. van Benthem, M. Moortgat & W. Buszkowski, editors: A Festschrift for Jim Lambek, Linguistic Analysis 36, pp. 345–384, doi:10.48550/arXiv.1003.4394. Arxiv:1003.4394.

[17] Bob Coecke, Fabrizio Genovese, Stefano Gogioso, Dan Marsden & Robin Piedeleu (2018): *Uniqueness of Composition in Quantum Theory and Linguistics*. Electronic Proceedings in Theoretical Computer Science 266, pp. 249–257, doi:10.4204/eptcs.266.17.

[18] Matthew DeCross, Reza Haghshenas, Minzhao Liu, Yuri Alexeev, Charles H. Baldwin, John P. Bartolotta, Matthew Bohn, Eli Chertkov, Jonhas Colina, Davide DelVento, Joan M. Dreiling, Cameron Foltz, John P. Gaebler, Thomas M. Gatterman, Christopher N. Gilbreth, Johnnie Gray, Dan Gresh, Nathan Hewitt, Ross B. Hutson, Jacob Johansen, Dominic Lucchetti, Danylo Lykov, Ivaylo S. Madjarov, Karl Mayer, Michael Mills, Pradeep Niroula, Enrico Rinaldi, Peter E. Siegfried, Bruce G. Tiemann, Curtis Volin, James Walker, Ruslan Shaydulin, Marco Pistoia, Steven. A. Moses, David Hayes, Brian Neyenhuis, Russell P. Stutz & Michael Foss-Feig (2024): *The computational power of random quantum circuits in arbitrary geometries*, doi:10.48550/arXiv.2406.02501. arXiv:2406.02501.

[19] Tiffany Duneau (2021): *Solving Logical Puzzles in DisCoCirc*. Journal of Cognitive Science 22(3), pp. 355 – 389. Available at `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85125423576&partnerID=40&md5=4c6db945056372334795ff799ea0f56f`. Type: Article.

[20] Tiffany Duneau, Saskia Bruhn, Gabriel Matos, Tuomas Laakkonen, Katerina Saiti, Anna Pearson, Konstantinos Meichanetzidis & Bob Coecke: *Scalable and interpretable quantum natural language processing: an implementation on trapped ions*. In preparation.

[21] Vedran Dunjko & Hans J Briegel (2018): *Machine learning & artificial intelligence in the quantum domain: a review of recent progress*. Reports on Progress in Physics 81(7), doi:10.1088/1361-6633/aab406. Available at `https://dx.doi.org/10.1088/1361-6633/aab406`.

[22] Christoph Durr & Peter Hoyer (1999): *A Quantum Algorithm for Finding the Minimum*, doi:10.48550/arXiv.quant-ph/9607014. arXiv:quant-ph/9607014.

[23] V. Giovannetti, S. Lloyd & L. Maccone (2008): *Quantum random access memory*. Physical review letters 100(16), p. 160501, doi:10.1103/PhysRevLett.100.160501.

[24] E. Grefenstette & M. Sadrzadeh (2011): *Experimental Support for a Categorical Compositional Distributional Model of Meaning*. In: The 2014 Conference on Empirical Methods on Natural Language Processing., pp. 1394–1404, doi:10.48550/arXiv.1106.4058. ArXiv:1106.4058.

[25] Carys Harvey, Richie Yeung & Konstantinos Meichanetzidis (2023): *Sequence Processing with Quantum Tensor Networks*, doi:10.48550/arXiv.2308.07865. arXiv:2308.07865.

[26] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley & E Miles Stoudenmire (2019): *Towards quantum machine learning with tensor networks*. Quantum Science and Technology 4(2), p. 024001, doi:10.1088/2058-9565/aaea94.

[27] Richard Jozsa & Akimasa Miyake (2008): *Matchgates and classical simulation of quantum circuits*. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 464(2100), p. 3089–3106, doi:10.1098/rspa.2008.0189.

[28] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark & B. Coecke (2021): *lambeq: An Efficient High-Level Python Library for Quantum NLP*. arXiv preprint arXiv:2110.04236, doi:10.48550/arXiv.2110.04236.

[29] D. Kartsaklis & M. Sadrzadeh (2013): *Prior disambiguation of word tensors for constructing Sentence vectors.* In: The 2013 Conference on Empirical Methods on Natural Language Processing., ACL, pp. 1590–1601. `https://aclanthology.org/D13-1166/`.

[30] J. Lambek (1958): *The mathematics of sentence structure.* American Mathematics Monthly 65, doi:10.1080/00029890.1958.11989160.

[31] J. Lambek (1999): *Type grammar revisited.* Logical Aspects of Computational Linguistics 1582, doi:10.1007/3-540-48975-4_1.

[32] J. Lambek (2008): *From Word to Sentence.* Polimetrica, Milan. ISBN: 8876991174. `https://www.math.mcgill.ca/barr/lambek/pdffiles/2008lambek.pdf`.

[33] Yeong-Cherng Liang, Yu-Hao Yeh, Paulo E M F Mendonça, Run Yan Teh, Margaret D Reid & Peter D Drummond (2019): *Quantum fidelity measures for mixed states.* Reports on Progress in Physics 82(7), p. 076001, doi:10.1088/1361-6633/ab1ca4.

[34] Henry Lin & Max Tegmark (2017): *Critical Behavior in Physics and Probabilistic Formal Languages.* Entropy 19(7), p. 299, doi:10.3390/e19070299.

[35] Jonathon Liu, Razin A. Shaikh, Benjamin Rodatz, Richie Yeung & Bob Coecke (2023): *A Pipeline For Discourse Circuits From CCG*, doi:10.48550/arXiv.2311.17892. arXiv:2311.17892.

[36] Ziming Liu, Eric Gan & Max Tegmark (2023): *Seeing is Believing: Brain-Inspired Modular Training for Mechanistic Interpretability*, doi:10.48550/arXiv.2305.08746. arXiv:2305.08746.

[37] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis & Bob Coecke (2023): *QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer.* Journal of Artificial Intelligence Research 76, p. 1305–1342, doi:10.1613/jair.1.14329.

[38] Gabriel Matos, Chris N. Self, Zlatko Papić, Konstantinos Meichanetzidis & Henrik Dreyer (2023): *Characterization of variational quantum algorithms using free fermions.* Quantum 7, p. 966, doi:10.22331/q-2023-03-30-966.

[39] Konstantinos Meichanetzidis, Stefano Gogioso, Giovanni de Felice, Nicolò Chiappori, Alexis Toumi & Bob Coecke (2021): *Quantum Natural Language Processing on Near-Term Quantum Computers.* Electronic Proceedings in Theoretical Computer Science 340, p. 213–229, doi:10.4204/eptcs.340.11.

[40] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice & Bob Coecke (2023): *Grammar-aware sentence classification on quantum computers.* Quantum Machine Intelligence 5(1), doi:10.1007/s42484-023-00097-1.

[41] Kosuke Mitarai, Masahiro Kitagawa & Keisuke Fujii (2019): *Quantum analog-digital conversion.* Phys. Rev. A 99, p. 012301, doi:10.1103/PhysRevA.99.012301. Available at `https://link.aps.org/doi/10.1103/PhysRevA.99.012301`.

[42] Michael Ragone, Bojko N. Bakalov, Frédéric Sauvage, Alexander F. Kemper, Carlos Ortiz Marrero, Martin Larocca & M. Cerezo (2023): *A Unified Theory of Barren Plateaus for Deep Parametrized Quantum Circuits*, doi:10.48550/arXiv.2309.09342. arXiv:2309.09342.

[43] Benjamin Rodatz, Razin A. Shaikh & Lia Yeh (2021): *Conversational Negation using Worldly Context in Compositional Distributional Semantics*, doi:10.48550/arXiv.2105.05748. arXiv:2105.05748.

[44] M. Sadrzadeh, S. Clark & B. Coecke (2013): *The Frobenius anatomy of word meanings I: subject and object relative pronouns.* Journal of Logic and Computation 23, pp. 1293–1317, doi:10.1093/logcom/ext044. ArXiv:1404.5278.

[45] Yukie Sano, Hideki Takayasu & Misako Takayasu (2012): *Zipf's Law and Heaps' Law Can Predict the Size of Potential Words.* Progress of Theoretical Physics Supplement 194, p. 202–209, doi:10.1143/ptps.194.202.

[46] Maria Schuld, Ilya Sinayskiy & Francesco Petruccione (2014): *An introduction to quantum machine learning.* Contemporary Physics 56(2), p. 172–185, doi:10.1080/00107514.2014.964942.

[47] Sukin Sim, Peter D. Johnson & Alán Aspuru-Guzik (2019): *Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms*. Advanced Quantum Technologies 2(12), doi:10.1002/qute.201900070.

[48] Andrei Tomut, Saeed S. Jahromi, Sukhbinder Singh, Faysal Ishtiaq, Cesar Muñoz, Prabdeep Singh Bajaj, Ali Elborady, Gianni del Bimbo, Mehrazin Alizadeh, David Montero, Pablo Martin-Ramiro, Muhammad Ibrahim, Oussama Tahiri Alaoui, John Malcolm, Samuel Mugel & Roman Orus (2024): *CompactifAI: Extreme Compression of Large Language Models using Quantum-Inspired Tensor Networks*, doi:10.48550/arXiv.2401.14109. arXiv:2401.14109.

[49] V. Wang-Mascianica, J. Liu & B. Coecke (2023): *Distilling Text into Circuits*. arXiv preprint arXiv:2301.10595, doi:10.48550/arXiv.2301.10595.

[50] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin & Tomas Mikolov (2015): *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*, doi:10.48550/arXiv.1502.05698. arXiv:1502.05698.

[51] Dominic Widdows, Willie Aboumrad, Dohun Kim, Sayonee Ray & Jonathan Mei (2024): *Natural Language, AI, and Quantum Computing in 2024: Research Ingredients and Directions in QNLP*. arXiv preprint arXiv:2403.19758, doi:10.48550/arXiv.2403.19758.

[52] Dominic Widdows, Aaranya Alexander, Daiwei Zhu, Chase Zimmerman & Arunava Majumder (2024): *Near-term advances in quantum natural language processing*. Annals of Mathematics and Artificial Intelligence, doi:10.1007/s10472-024-09940-y.

[53] N. Wiebe, A. Kapoor & K. M. Svore (2015): *Quantum nearest-neighbour algorithms for machine learning*. Quantum Information and Computation 15, pp. 318–358, doi:10.26421/QIC15.3-4-7.

[54] William Zeng & Bob Coecke (2016): *Quantum Algorithms for Compositional Natural Language Processing*. Electronic Proceedings in Theoretical Computer Science 221, p. 67–75, doi:10.4204/eptcs.221.8.

[55] Jun Zhang, Jiri Vala, Shankar Sastry & K. Birgitta Whaley (2003): *Exact Two-Qubit Universal Quantum Circuit*. Physical Review Letters 91(2), doi:10.1103/physrevlett.91.027903.

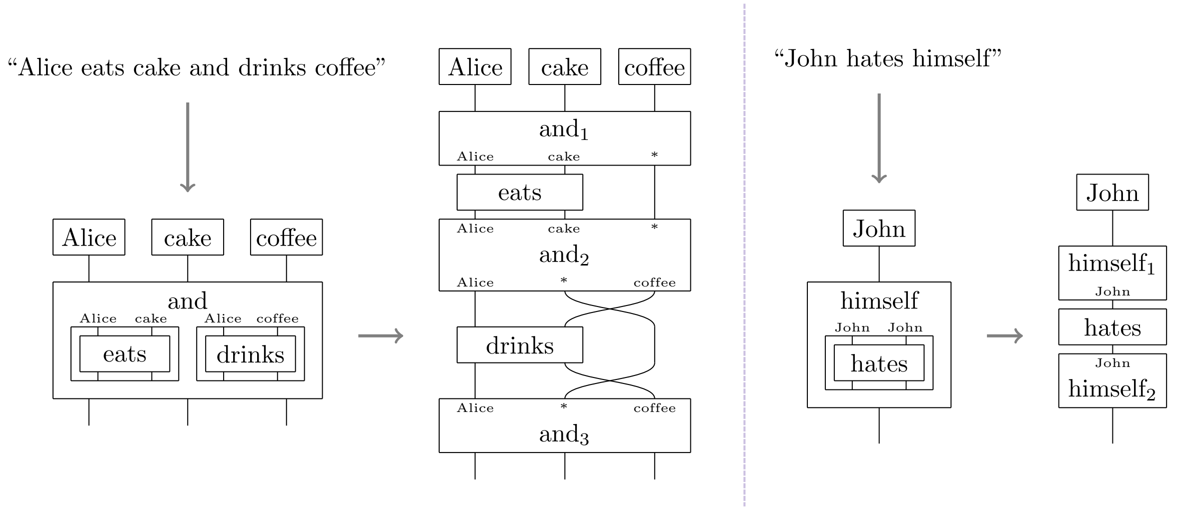
A Decomposing Frames

Our semantic functor maps states, effects, and boxes to pure quantum operations trivially. However, frames, which are mapped to quantum supermaps, in general, need some special treatment. Some frames in text circuits may contain more wires for one of their arguments than are input and output to the frame – this arises in many cases, for instance in reflexive structures. We cannot represent these extra wires as auxiliary qubits because these would then be subsequently discarded, making the state mixed. In general, quantum supermaps can be written as pairs of unitaries conjugating their arguments, along with a side channel consisting of some auxiliary qubits. However, this conflicts with our no-ancilla requirement.

Therefore, we will instead *constrain* our representation of frames as follows:

- We lay the frame out vertically, with unitaries before the first argument, between every argument, and after the last argument.
- For every wire in each argument, we map it to one of the wires of the unitary: if that wire corresponds to an unused noun wire that is input to the frame, we map it to that wire. Otherwise, we map it to any unused noun wire. If there are none, then we delete the wire.
- After this mapping, we substitute the argument between the unitaries of the frame, and any unused wires are connected directly between the unitaries of the frame to serve as side-channels.

In order to resolve which wires correspond to which nouns (and thus where to assign each wire of the frame arguments), we rely on coindexing information provided by the parser, which within sentences is provided by finding the grammatical 'head' of each phrase, and between sentences is provided by a coreference tool [35]. To minimize the number of wires that are deleted, we perform this mapping for each argument independently, not respecting the wire assignments in previous arguments. This may create boxes that are ungrammatical, but we assign them unitaries in the same way as other boxes, so this is not an issue.



This illustrates two examples of the sandwich construction. In each, the wires inside the unitaries making up the frame are labeled with which nouns they are representing, and wires that are being used as side channels are labeled with '*'. In the first example, we see that swaps may be introduced to map wires to the correct locations. This is an artifact of the notation and does not appear in the compiled quantum circuits. In the second example we see a wire being deleted since there are no unused wires to assign it to, and in the process this creates an ungrammatical box for 'hates', since it is a transitive verb.

B Proofs of Theorems

Theorem 2. QDisCoCirc-QA *can be solved on a quantum computer in time*

$$O\left(\frac{k\log\left(\frac{k}{\delta}\right)|V|_w(|T|+\max_i|Q_i|)}{\varepsilon^2}\right)$$

with failure probability δ, where $|V|_w$ is the maximum size of any word embedding in V.

Proof. To calculate each

$$p_j = \operatorname{tr}\left(\rho_T(\rho_{Q_j}\otimes I)\right)$$

we produce a circuit using the swap test where the probability of measuring a zero on the first qubit is $\frac{1}{2}+\frac{p_j}{2}$. Let $\mu_{j,n}$ be the estimate of p_j after n shots. By Hoeffding's inequality,

$$P(|\mu_{j,n}-p_j|\geq\varepsilon')\leq 2\exp\left(-\frac{n\varepsilon'^2}{2}\right)$$

thus if we take $n=\frac{2}{\varepsilon'^2}\log(\frac{2k}{\delta})$, then we have

$$P\left(\bigwedge_{j=1}^{k}|\mu_{j,n}-p_j|\leq\varepsilon'\right)\geq\left(1-\frac{\delta}{k}\right)^k\geq 1-\delta$$

by Bernoulli's inequality. Note that if $|\mu_{j,n}-p_j|\leq\varepsilon'$ for all j, then $|\max_i\mu_{n,i}-\max_i p_i|\leq\varepsilon'$, and hence if $j=\arg\max_i\mu_{n,i}$ then $|p_j-\max_i p_i|\leq 2\varepsilon'$. Therefore, if $\varepsilon'=\frac{\varepsilon}{2}$, then this j correctly solves the QDisCoCirc-QA instance with probability $1-\delta$. For each j, each shot of the swap test circuit takes $O\left((|T|+|Q_j|)|V|_w\right)$ time since U_T and U_{Q_j} contain $O(|T||V|_w)$ and $O(|Q_j||V|_w)$ gates respectively, and the width of U_{Q_j} must be at most $2|Q_j||V|_w$. Therefore, we have an overall time complexity of:

$$O\left(\sum_i(|T|+|Q_i|)|V|_w n\right)=O\left(\frac{k\log\left(\frac{k}{\delta}\right)|V|_w(|T|+\max_i|Q_i|)}{\varepsilon^2}\right)$$

□

Definition 3 (Approx-QCircuit-Prob)**.** *Suppose we are given a quantum circuit C on n qubits with m 2-local gates, where C is taken from a uniform family such that m is polynomial in n, along with the promise that either*

- *measuring the first qubit $C|0\rangle$ has probability at least $\frac{2}{3}$, or*
- *measuring the first qubit $C|0\rangle$ has probability at most $\frac{1}{3}$.*

Then the Approx-QCircuit-Prob *problem is to distinguish between these two cases. This is BQP-hard.*

Theorem 3. *Suppose that a set of word embeddings V satisfies the following:*

1. *The operations of V use one qubit for each input wire,*
2. *V contains arbitrarily many proper nouns,*
3. *V contains at least two adjectives that generate a dense subset of $SU(2)$,*

4. V contains at least one transitive verb that is entangling

then for any fixed $\varepsilon < \frac{1}{7}$, QDisCoCirc-QA *is BQP-hard.*

Proof. Let U_1 and U_2 be the adjectives defined by condition three, and U_3 be the transitive verb identified by condition four. Then, by the inverse-free Solovay-Kitaev algorithm [8], any single-qubit unitary U can be approximated to precision ε' by a sequence of U_1 and U_2 operations of length $O(\log^{8.62}(\frac{1}{\varepsilon'}))$. From a result of Zhang et al [55], any entangling gate combined with arbitrary single-qubit gates can be used to implement any two-qubit gate exactly in a constant number of gates (where the constant depends only on the definition of the entangling gate). Hence, any two-qubit gate can also be approximated arbitrarily well by $O(\log^{8.62}(\frac{1}{\varepsilon'}))$ applications of U_1, U_2, and U_3.

Now we can reduce from the Approx-QCircuit-Prob problem to QDisCoCirc-QA, by constructing a QDisCoCirc circuit corresponding to the given circuit C. First, assign each qubit a proper noun N_i with corresponding unitary U_{N_i} in V. To each noun apply the unitary $U_{N_i}^\dagger$. Each $U_{N_i}^\dagger$ is approximated as defined above. Then, for each gate in C, we approximate it as discussed above and apply it to the corresponding wires. This yields a QDisCoCirc circuit U_T which approximates the original circuit C.

From this, we can generate T as follows: for each gate in the circuit, we generate a sentence. If the gate acts on a single qubit i, we write "$\{N_i\}$ is $\{U_k\}$." where U_k is the adjective corresponding to the gate. If the gate acts on two qubits i and j, we write "$\{N_i\}$ $\{U_3\}$ $\{N_j\}$." To produce Q_1 and Q_2, we first assemble quantum circuits $U_{Q_1} = U_{N_1}^\dagger U_{N_1}$ and $U_{Q_2} = U_X U_{N_1}^\dagger U_{N_1}$ where U_X is an approximation of the X gate, and $U_{N_1}^\dagger$ is also approximated as above. Then Q_1 and Q_2 can be generated in the same way as T.

Let $\varepsilon' \geq \frac{1}{42}$ be such that $\varepsilon' + \varepsilon < \frac{1}{6}$, and let $P_U(i)$ be the probability of measuring $|i\rangle$ on the first qubit of $U|0\rangle$. Suppose we solve the QDisCoCirc-QA instance associated with V, T, $\{Q_1, Q_2\}$ to precision ε. Note that since $U_{Q_1}|0\rangle \approx |0\rangle$, $\mathrm{tr}(\rho_T(\rho_{Q_1} \otimes I)) \approx P_{U_T}(0)$. Likewise, since $U_{Q_2}|0\rangle \approx |1\rangle$, $\mathrm{tr}(\rho_T(\rho_{Q_2} \otimes I)) \approx P_{U_T}(1)$. As $U_T \approx C$, if we find $j = 1$, then we can conclude that $P_C(0) > P_C(1) = 1 - P_C(0) \implies P_C(0) > \frac{1}{2}$ up to some error $\varepsilon + \varepsilon' < \frac{1}{6}$. However, in the definition of Approx-QCircuit-Prob, we are guaranteed that $P_C(0) > \frac{2}{3}$ or $P_C(0) < \frac{1}{3}$, which are separated from $\frac{1}{2}$ by $\frac{1}{6}$, so by checking whether $j = 1$ or $j = 2$, we can distinguish between these cases. Therefore, QDisCoCirc-QA is BQP-hard. □

Theorem 4. *Given a set of word embeddings V, suppose that operations in V are independent Haar-random unitaries. Then conditions three and four of Theorem 3 are almost surely satisfied for all word embeddings containing at least two adjectives and one transitive verb.*

Proof. Let U_1 and U_2 be the adjectives defined by condition three of Theorem 1, and U_3 be the transitive verb identified by condition four. Consider U_3 as a point in $SU(4)$. If U_3 is not entangling it is either separable or locally equivalent to the SWAP gate. The spaces formed in both of these cases are isomorphic to $SU(2) \times SU(2)$ which is a 9-dimensional subspace of the 15-dimensional $SU(4)$. Hence condition four is satisfied almost everywhere in $SU(4)$. Therefore, for Haar random U_3 it is almost surely satisfied.

Consider arbitrary U_1 and U_2. If U_1 and U_2 are non-commuting elements of infinite order in $SU(2)$, then they generate a dense subset of $SU(2)$. This is because the only closed positive-dimension Lie subgroup of $SU(2)$ is itself, so it must be equal to the closure of the group generated by U_1 and U_2. Consider U_1 and U_2 as a point in $SU(2) \times SU(2)$. Since elements of $SU(2)$ can also be characterized as rotations of the Bloch sphere, then commuting elements correspond to rotations around the same axis. Thus the subspace of commuting U_1 and U_2 is isomorphic to $SU(2) \times S$. Likewise, elements of finite order in $SU(2)$ correspond to rotations about any axis by a rational multiple of π, and the subspace is isomorphic to $S^2 \times \mathbb{Q}$, which has measure zero. Thus almost everywhere in $SU(2) \times SU(2)$, U_1 and U_2 generate a dense subset of $SU(2)$. Hence, for Haar random U_1 and U_2, this is almost surely the case. □

Theorem 5. *Given a specific set of word embeddings V it is possible to check numerically that the conditions of Theorem 3 are satisfied.*

Proof. Let U_1 and U_2 be the adjectives defined by condition three of Theorem 3, and U_3 be the transitive verb identified by condition four. Consider the state $|U_3\rangle$ obtained via the Choi-Jamiołkowski isomorphism. If we compute the reduced density matrix $\hat{\rho} = \mathrm{tr}_2\left(|U_3\rangle\langle U_3^\dagger|\right)$ (where the partial trace is taken over both qubits corresponding to the second qubit of U_3), then this is pure (i.e has rank one) if and only if U_3 is separable. Therefore, we can check that U_3 is entangling by performing this test on both U_3 and $U_3 \cdot SWAP$ (to check that it is not locally equivalent to a SWAP).

To check that U_1 and U_2 generate a dense subset of $SU(2)$, first check that they do not commute. We wish to check that the closure of the subgroup generated by U_1 and U_2 is $SU(2)$ itself. The only infinite closed nonabelian proper subgroup of $SU(2)$ is isomorphic to $S \rtimes_\phi C_2$ (where $\phi_0(x) = x$, $\phi_1(x) = -x$). In this subgroup for all x and y, x^2 and y^2 commute, so to rule out this case it suffices to find a non-commuting pair. This can be done numerically by generating elements of the group by taking products and checking randomly until an example is found.

The only finite closed nonabelian subgroups of $SU(2)$ are the binary dicylic groups, and three groups of order 24, 48 (the single-qubit Cliffords), and 120, by the ADE classification. The binary dicylic groups are subgroups of $S \rtimes_\phi C_2$, so we have already ruled these out. Therefore, it remains to check that the order of the group generated by U_1 and U_2 is more than 120. This can be done numerically by testing products of group elements until 121 unique elements are generated. □

Theorem 6. *There exists a polynomial $f(m,n)$, such that given an oracle to solve* QDisCoCirc-QA *instances with arbitrary word embeddings and text circuits drawn from $\mathscr{D}_{f(m,n)}$, we can perform arbitrary quantum computations with m 2-local gates on n qubits with high probability in polynomial time. That is,* QDisCoCirc-QA *for worst-case word embeddings is average-case BQP-hard over texts.*

Proof. We will reduce from the problem Approx-QCircuit-Prob. We are given an arbitrary quantum circuit on n qubits with m two-qubit gates given by arbitrary unitaries, and we assume that any single-qubit gates is combined into an adjacent two-qubit gate – the only time this is not possible is when a qubit is unentangled, so it can be disregarded (if it is the measurement qubit, we can calculate the outcome immediately, and if it is not, it doesn't affect the outcome).

Suppose that we have sampled a large random text circuit from $\mathscr{D}_k$ for some $k \gg n$. We wish to calculate how many boxes are required so that there exists a subsequence which matches the given circuit. By match, we mean that each gate has a corresponding box in the circuit with a unique label, operating on the same wires as the gate. Let N_i be the gap between gates i and the previous gate (or that start of the circuit) – that is, the number of boxes between them in the circuit that are not part of the subsequence. Because all boxes in the circuit are chosen in an identical random manner, $P(N_i \geq q) = (1-p_i)^q$ where p_i is the probability of any particular box being correct for gate i. Since inputs to boxes are chosen uniformly, we have that all the $p_i = p$ are identical.

Therefore, we can bound the probability that the total sequence length required is at least some constant q:

$$P\left(\sum_i N_i \geq q\right) \leq 1 - P\left(\bigwedge_i N_i < \frac{q}{m}\right) = 1 - \left(1 - P\left(N_i \geq \frac{q}{m}\right)\right)^m = 1 - (1 - (1-p)^{\frac{q}{m}})^m$$
$$\leq 1 - (1 - m(1-p)^{\frac{q}{m}}) = m(1-p)^{\frac{q}{m}} \leq me^{-\frac{pq}{m}}$$

The second line follows from Bernoulli's inequality. Thus letting $q = \frac{m}{p}(\log(m)+1)$, we have that $P\left(\sum_i N_i \geq q\right) \leq \frac{1}{e}$. We have

$$p = P(A=2) \cdot P(\text{hapax}) \cdot \frac{2}{k(k-1)}$$

where $P(\text{hapax})$ is the probability that any particular box has a unique label, since $\frac{2}{k(k-1)}$ is the probability of selecting any particular pair of wires uniformly, assuming that $k > n$. According to Heap's law, the number of distinct words in a text grows like $\sqrt{w}$ where w is the total number of words. Furthermore from Zipf's law, a constant proportion of distinct words occur only once (*hapax legomena*), usually about half for large corpora. By assumption, this law holds exactly for circuits drawn from $\mathscr{D}_k$, so this implies that

$$P(\text{hapax}) \leq \frac{c_1\sqrt{k^\gamma}}{k^\gamma} = \frac{c_1}{\sqrt{k^\gamma}} \quad \text{and} \quad p \geq \frac{c_2}{k^{2+\frac{\gamma}{2}}}$$

for some constants $c_1, c_2 > 0$, since we assume $w \geq c_3 k^\gamma$ for some c_3. Therefore, we can set $q \leq \frac{1}{c_2} m(\log(m)+1)k^{2+\frac{\gamma}{2}}$, but we also require $c_3 k^\gamma \geq q$ to ensure the number of boxes required is less than the total in the text. This can be enforced by setting:

$$c_3 k^\gamma \geq \frac{1}{c_2} m(\log(m)+1)k^{2+\frac{\gamma}{2}} \implies k \geq \left(\frac{1}{c_2 c_3} m(\log(m)+1)\right)^{\frac{2}{\gamma-4}}$$

Finally, in order to make sure $k \geq n$, we can set $k = f(m,n) = \max(n, \lceil (cm(\log(m)+1))^{\frac{2}{\gamma-4}} \rceil)$ for some $c > 0$.

The protocol to solve Approx-QCircuit-Prob is as follows: we are given a circuit C on n qubits with m two-qubit gates (single-qubit gates are eliminated as before). Sample a text circuit from $\mathscr{D}_k$ with k as given above. With probability at least $1 - e^{-1}$, there exists a subsequence of boxes in the circuit that matches C'. We construct a set of word embeddings for the circuit by setting each word that does not appear in the subsequence to be the identity, and every word that is part of the subsequence to be the unitary of the corresponding gate in the circuit. Each state is assigned to the $|0\rangle$ quantum state. This defines the text circuit and embeddings for the context text T. Similarly, we can obtain the questions Q_1 and Q_2 by drawing from $\mathscr{D}_1$, setting all boxes to the identity, and assigning the state to $|0\rangle$ and $|1\rangle$ respectively.

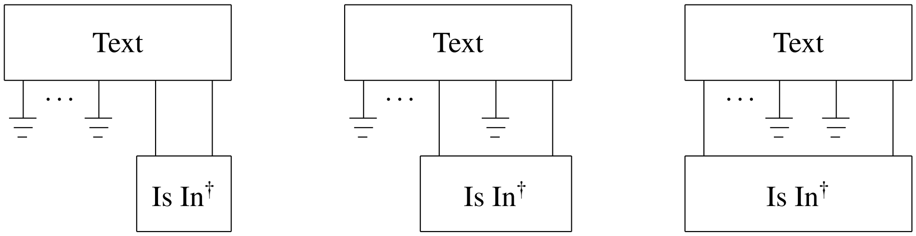
In the same way as in Theorem 3, solving this instance of QDisCoCirc-QA with $\varepsilon < \frac{1}{7}$ is sufficient to solve the corresponding Approx-QCircuit-Prob instance. $f(m,n)$ as given above grows polynomially if $\gamma > 4$, and the construction of the word embeddings is done in polynomial time. Since this procedure has fixed positive success probability, it can be amplified to any high success probability with logarithmically many iterations. □

C Oracle Construction

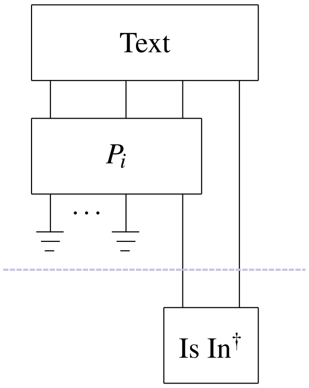
As a warmup, we will start by constructing oracles for a different task called bAbI1, first proposed in [50]. In the DisCoCirc model of the bAbI1 task, we have a context text from which we extract information by asking a series of yes/no questions. An example context text is:

> John goes to the bedroom.
> Mary walks to the hallway.
> John goes back to the bathroom.

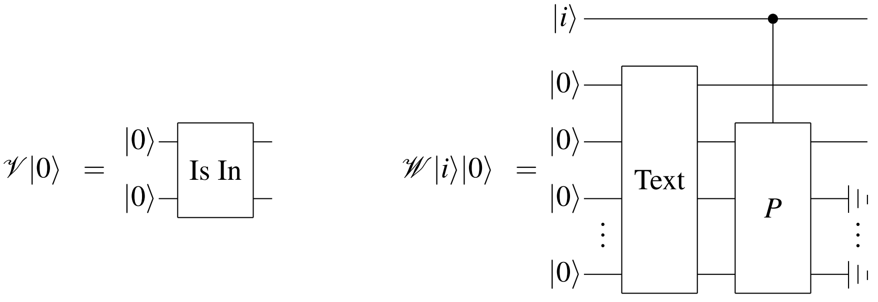
And the task then is to answer a question about the state of the world. For example: “Where is John?”. In DisCoCirc, we do this by stating a potential answer to the question as an affirmative: “John is in the bathroom.” Then we measure which potential answer (“John is in the bathroom”, “John is in the hallway”, etc) has the highest overlap with the context text, discarding the nouns that do not occur in the answer. Graphically, this means we wish to find the maximum of the following circuits:



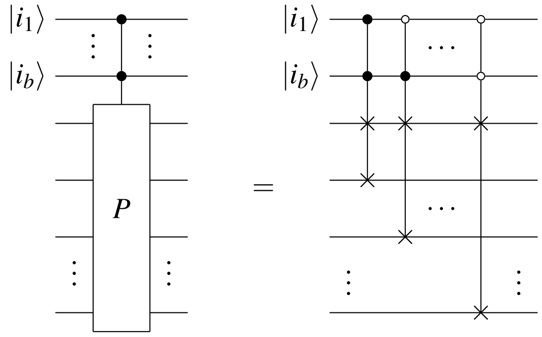
We can phrase this as a instance of the closest vector problem by noting that this is equivalent to finding P_i which maximizes the following probability. The set $\{P_1, P_2 \cdots P_i \cdots\}$ is any set of permutations such that P_j brings the jth input to the top.



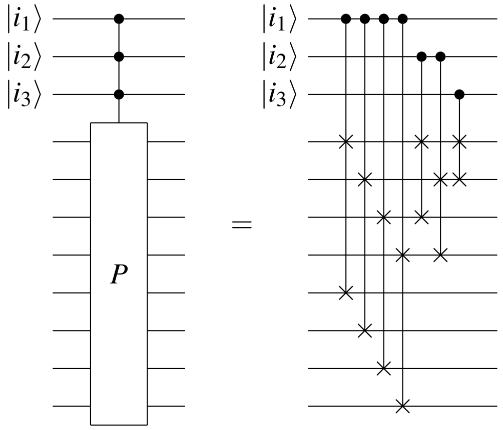
To view this as a closest vector problem, we divide the problem along the dotted line, letting the upper portion be the variable set of states ρ_i and the lower portion be $\rho_0 = |v_0\rangle\langle v_0|$. Since ρ_0 is pure, we can apply the quantum closest vector algorithm of Basheer et al [5] to find $i = \arg\max_{j\geq 1} \mathrm{F}(\rho_j, \rho_0)$. For this task the oracles $\mathscr{V}$ and $\mathscr{W}$ are given by the following circuits:



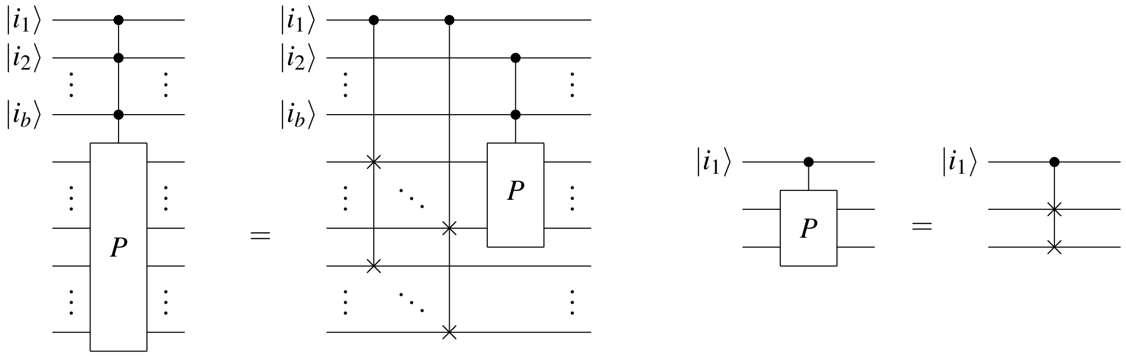
Here, P is a 1-to-n *multiplexer*, it maps each wire to the top output (leaving the rest undefined) depending on the control register $|i\rangle$. There are several ways of constructing this operation. For instance, we can do a CSWAP controlled on every possible bitstring of the control register $|i\rangle = |i_1 i_2 \cdots i_b\rangle$. Here, the CSWAPs are swapping nouns, so they can be realized with $O(\log N)$ CSWAPs acting on qubits. This looks like the following circuit:



The gate complexity of this is $O(n\log(N)\log(n))$ since we have to put a CX gate with $O(\log(n))$ controls on each of $O(n\log(N))$ qubits, and swaps the noun at the index with the top noun, leaving the rest unchanged. However, we can do better by leaving the other nouns in an unspecified order using a binary search method - first, we swap the first and second half of the nouns depending on the first bit of the index, then the first and second quarter depending on the second bit of the index, and so on, until the last bit controls whether we swap the top two nouns. This requires $n\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots\right)O(\log(N)) = O(n\log(N))$ singly-controlled Toffolis, so has an overall gate complexity of $O(n\log(N))$. For example, the circuit on eight words looks like the following:



P can be constructed recursively as:



Therefore, the oracles are given as follows in pseudocode:

```
fn V(n1, n2) {
    IsIn(n1, n2);
}

fn NounCSWAP(ctrl, n1, n2) {
    for q in 0..log(N) {
        if ctrl {
            SWAP(n1[q], n2[q]);
        }
    }
}

fn P(i, nouns) {
    if nouns.len() == 1 {
        return;
    } else if nouns.len() == 2 {
        NounCSWAP(i[0], nouns[0], nouns[1])
    } else {
        const m = nouns.len() / 2;

        for j in 0..m {
            NounCSWAP(i[0], nouns[j], nouns[m + j]);
        }

        P(i[1..], nouns[..m]);
    }
}

fn W(i, n1, n2) {
    let ancillas = InitAncillas(log(N) * (n - 2));
    let nouns = [n1, n2, ..ancillas];
    Text(nouns);
    P(i, nouns);
    Discard(ancillas);
}
```

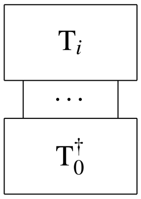
The gate complexity of the $\mathscr{W}$ oracle is thus $O\left(wA(N,d)+n\log(N)\right)$, and for the $\mathscr{V}$ oracle it is $O(A(N,2)) = O(A(N,d))$. It is reasonable to make the assumption that $A(N,d) = O(d\log(N))$. For example, this is the case if we are using standard ansatze like those from Sim et al [47]. In the closest vector algorithm for this task, we have $M = O(n)$. Therefore, for a fixed success probability and precision ε for calculating fidelity, the overall gate complexity of the algorithm is:

$$O\left(\frac{\sqrt{n}}{\varepsilon}(wd+n+\log(N))\log(N)\right) \leq O\left(\frac{\sqrt{n}wd\log^2(N)}{\varepsilon}\right)$$

The inequality here follows from $n = O(w)$. Furthermore, the number of ancilla used by the oracles is at most $O(n\log(N))$.

C.1 The Text-Text Similarity Task

The text-text similarity task involves finding a text that is most similar to a target text, by maximizing the overlap between the states representing the two texts. This is a versatile primitive that can be used both for full text-text similarity and for some variants of question answering. As discussed in Section 5, we can use the closest vector algorithm to solve this problem. Pictorially, we want to find i such that



is maximized. For this task the oracles $\mathscr{V}$ and $\mathscr{W}$ are given by:

- $\mathscr{V}|0\rangle = |T_0\rangle$
- $\mathscr{W}|i\rangle = |T_i\rangle$

Clearly, we have $\mathscr{V} = U_{T_0}$ which is the direct translation of the text circuit into a quantum circuit via QDisCoCirc. However, for $\mathscr{W}$, we need a significantly more complicated circuit. Let us define the following parameters:

- Each noun wire has dimension N,
- Each text circuit considered has at most w unitaries and n noun wires total,
- Each unitary in the text circuit acts on at most d wires,
- There are V words in the vocabulary,
- The number of gates in the ansatz for d wires of dimension N scales as $A_g(N,d)$,
- The number of parameters in the ansatz for d wires of dimension N scales as $A_p(N,d)$,
- Each parameter for the ansatz is stored with P bits of precision,
- We consider M possible texts, so that $i \leq M$.

We will construct the circuit for $\mathscr{W}$ by interleaving layers of two operations. One operation is a parameterized partial permutation, which permutes some of its inputs according to information stored in a QRAM. The second is a parameterized ansatz, which applies an ansatz of a given size to a set of qubits with the parameterized angles of the ansatz stored in a QRAM. By combining these two operations, we can reconstruct a text circuit by loading from QRAM the parameters and argument locations for each unitary.

We assume access to the following QRAM oracles:

- $\mathrm{Param}_{ij}|k\rangle|0\rangle = |k\rangle|\phi_{ijk}\rangle$ where ϕ_{ijk} is a P-bit fixed-precision binary encoding of the jth parameter for the ansatz of the ith unitary in T_k,
- $\mathrm{Width}_i|k\rangle|0\rangle = |k\rangle|W_{ik}\rangle$ where W_{ik} is a binary encoding of the width of the ith unitary in T_k,
- $\mathrm{Index}_{ij}|k\rangle|0\rangle = |k\rangle|I_{ijk}\rangle$ where I_{ijk} is an encoding of the position of the jth argument to unitary i in T_k. Note that the encoding here is not a straightforward mapping from zero-indexed position to binary, and we will elaborate more later.

Each of these oracles can be implemented using a series of calls (one for each bit) to bucket-brigade QRAMS of size M [23]. In pseudocode, each of these lookups is as follows:

```
fn QRAMLookup(dict, qidx, cidx, out) {
    const ORACLES: [[Unitary]];
    let size = out.len();
    for i in 0..size {
        ORACLES[dict][cidx * size + i](qidx, out[i]);
    }
}
```

where `dict` specifies the chosen oracle, `qidx` the quantum index $|k\rangle$ and `cidx` the classical index (i.e ij or i).

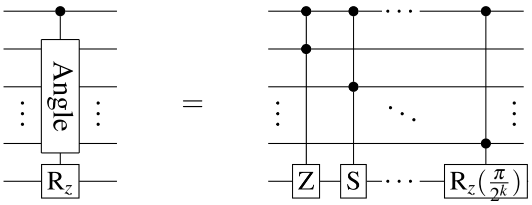
For the parameterized ansatz, we can construct a circuit out of some building blocks:

- Firstly, we have a parameterized controlled rotation gate, which implements a controlled-Z rotation based on an angle specified in auxiliary qubits.
- For a given ansatz of a specific size, we can implement a parameterized controlled circuit by replacing all gates except for parameterized rotations with their controlled versions. For the parameterized rotations, we then load the parameters from QRAM and implement each rotation using a parameterized controlled rotation circuit.
- This gives a parameterized ansatz of a specific width. To cover all possible widths, first we can load the width from QRAM, and perform the parameterized controlled ansatz of all possible widths, controlled on the width loaded from QRAM. This gives a general parameterized ansatz.

For the parameterized controlled rotation gate, we have:

```
fn PCRz(ctrl, angle, q) {
    const MAX_PREC;
    for i in 0..=MAX_PREC {
        CCRz(pi / (2^i), angle[i], ctrl, q);
    }
}
```

which is diagrammatically:

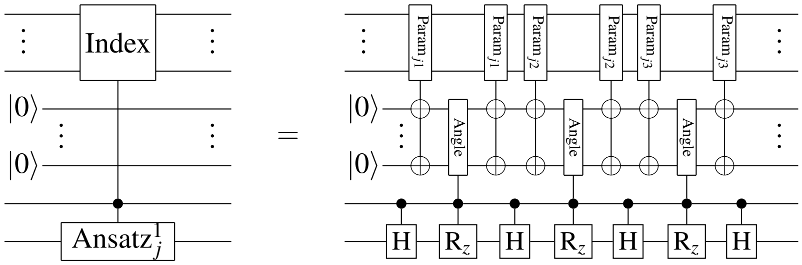


This has a gate complexity of $O(P)$.

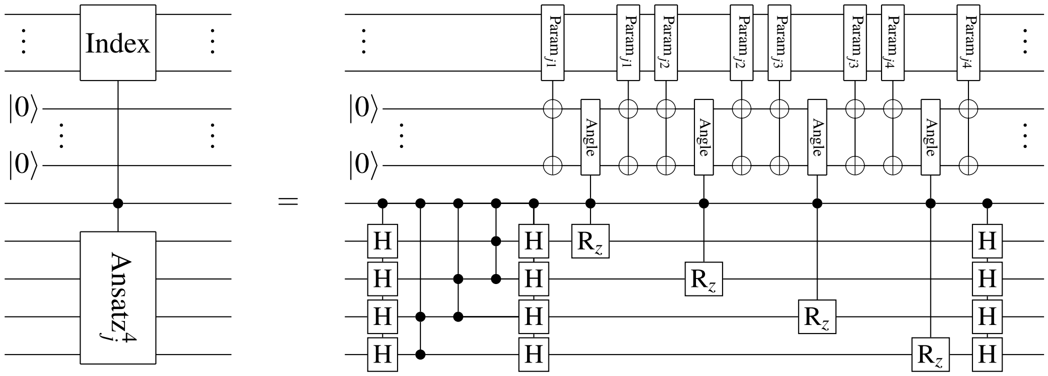
Then for a specific realization of an ansatz of a given width, we can convert it to a parameterized controlled ansatz as follows. Here `angle_base` refers to the location in QRAM of the parameters for this ansatz, `text_idx` is the quantum register specifying the text index, and `angle` is a ancilla register for storing a parameter.

```
fn PCAnsatz(ansatz, ctrl, text_idx, angle_base, angle, qs) {
    for gate in ansatz {
        match gate {
            CX(a, b) => CCX(ctrl, qs[a], qs[b]),
            RzParam(idx, a) => {
                QRAMLookup(
                    ANGLES, text_idx, angle_base + idx, angle
                );
                PCRz(ctrl, angle, qs[a]);
                QRAMLookup(
                    ANGLES, text_idx, angle_base + idx, angle
                );
            },
            RzFixed(p, a) => CRz(p, ctrl, qs[a]),
            H(a) => CH(ctrl, qs[a])
        }
    }
}
```

For example, for an ansatz of width one (parameterized Euler rotations), corresponding to $d = 1, N = 2$, we have:



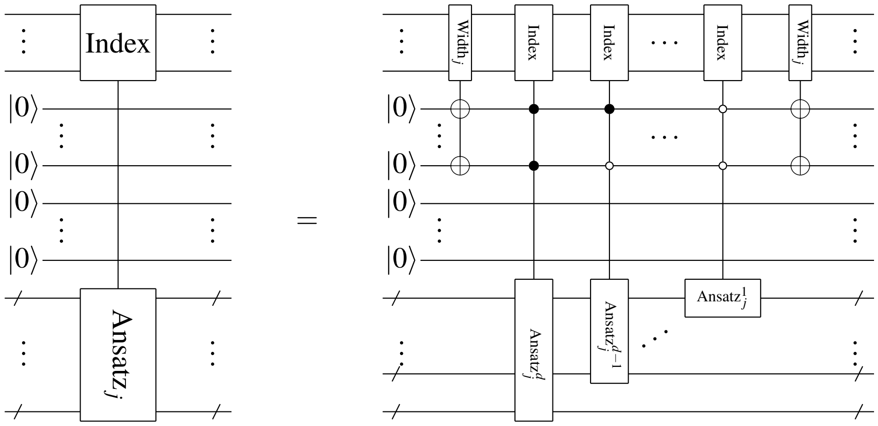
And for a one-layer of ansatz 9 from Sim et al. [47] of width four (i.e for $d = 4$ and $N = 2$ or $d = 2$ and $N = 4$) [47], we have the following:



This has a gate complexity of $O(A_g(N,d) + P\log(M)A_p(N,d))$. These ansatze of differing widths can be combined into one parameterized ansatz that controls all of these based on a width loaded from QRAM. We wish to do the following operation

```
fn PAnsatz(width, text_idx, angle_base, angle, qs) {
    const MAX_WIDTH;
    const ANSATZE = [
        AnsatzGates(w) for w in 1..=MAX_WIDTH
    ];


    QRAMLookup(WIDTHS, text_idx, i, width);
    for w in 1..=MAX_WIDTH {
        let ctrl = w == width;
        PCAnsatz(ANSATZE[w], ctrl, text_idx, angle_base, angle, qs)
        uncompute ctrl;
    }
    QRAMLookup(WIDTHS, text_idx, i, width);
}
```

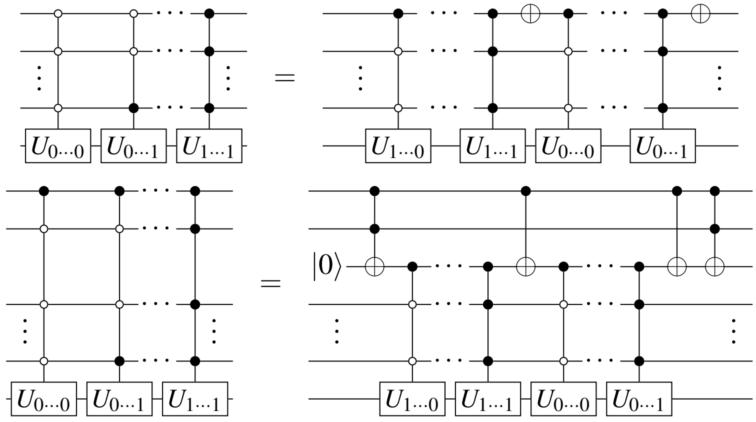
which is represented diagrammatically as:



However, compiling this circuit naively results in a gate complexity of

$$O(d\log(d)\cdot(A_g(N,d)+P\log(M)A_p(N,d)))$$

since each parameterized ansatz has $O(\log(d))$ controls that must be combined. By making use of the unary iteration method, we can compile this more efficiently with gate complexity of $O(d(A_g(N,d)+P\log(M)A_p(N,d)))$. Given a series of unitaries $U_{0\cdots 0}$ to $U_{1\cdots 1}$ which we want to apply conditionally given an index register of size $\log(d)$, this technique synthesizes a circuit using singly-controlled versions of $U_{i\cdots j}$ with only $O(d)$ gates. It is based on the following recurrences:



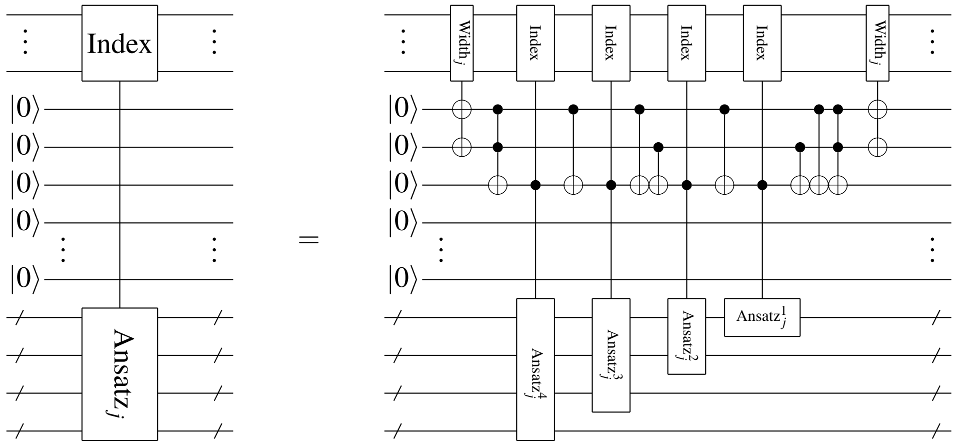
This second equation is applied until only single controls on each $U_{i\cdots j}$ remain. In pseudocode, this process is given by:

```
fn UnaryIteration(ctrls, ancillas, func) {
    X(ctrls[0]);
    CUnaryIteration(ctrls[1..], 0, ctrls[0], ancillas, func);
    X(ctrls[0]);
    CUnaryIteration(ctrls[1..], 1, ctrls[0], ancillas, func);
}

fn CUnaryIteration(ctrls, base, prev, ancillas, func) {
    if ctrls.len() > 0 {
        X(ctrls[0]);
        let fresh = ancillas[0];
        CCX(prev, ctrls[0], fresh);
        CUnaryIteration(
            ctrls[1..], 2*base + 0, fresh, ancillas[1..], func
        );
        CX(prev, fresh);
        CUnaryIteration(
            ctrls[1..], 2*base + 1, fresh, ancillas[1..], func
        );
        CCX(prev, ctrls[0], fresh);
        Discard(fresh);
    } else {
        func(base, prev);
    }
}
```

Applying this to create a parameterized ansatz, we have the following pseudocode:

```
fn PAnsatz(width, text_idx, angle_base, angle, ancillas, qs) {
    const MAX_WIDTH;
    const ANSATZE = [
        AnsatzGates(w) for w in 1..=MAX_WIDTH
    ];


    QRAMLookup(WIDTHS, text_idx, i, width);
    UnaryIteration(width, ancillas, |idx, ctrl| {
        if idx > 0 && idx <= MAX_WIDTH {
            PCAnsatz(
                ANSATZE[idx], ctrl, text_idx, angle_base, angle, qs
            );
        }
    });
    QRAMLookup(WIDTHS, text_idx, i, width);
}
```

For the case that `MAX_WIDTH = 4` and hence $\log(d) = 2$, this can be represented diagrammatically as follows (although note that we use a more optimized form of unary iteration on two index qubits).



Note that in this setup, the index $0\ldots0$ is mapped to an ansatz with width one, and in general x in binary maps to the ansatz with width $x+1$. Note that if we wish to apply the identity operator, we can apply an ansatz of width one with all angles set to zero. This has an overall gate complexity of $O(d(A_g(N,d)+P\log(M)A_p(N,d)))$.

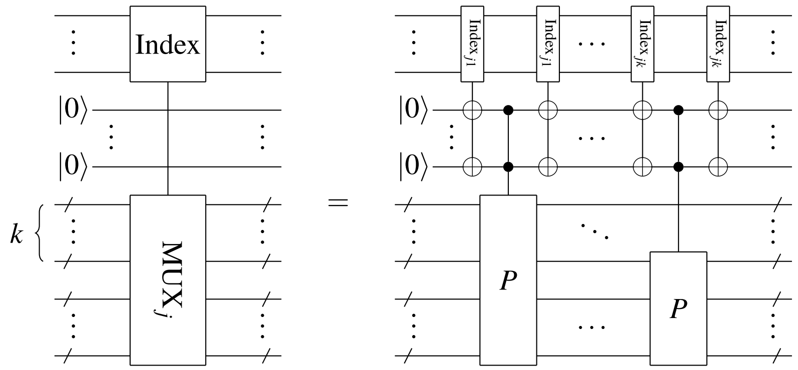
For the parameterized partial permutation, we can reuse the multiplexer given before to implement each bit of the partial permutation. This puts a bit of our choice at the top of the multiplexer and leaves the rest permuted. By stacking d of these together, we can specify which qubits should form the top d qubits that the ansatz is applied too. We can calculate the required indices by propagating the desired index forward through all previous changes to calculate the actual index required at each step. Noting that each swap in the multiplexer is actually a swap of noun wires (which consist of multiple qubits), we can represent the multiplexer in pseudocode as:

```
fn NounCSWAP(ctrl, n1, n2) {
    const N;

    for q in 0..log(N) {
        CSWAP(ctrl, n1[q], n2[q]);
    }
}

fn P(idx, nouns) {
    if nouns.len() == 1 {
        return;
    } else if nouns.len() == 2 {
        NounCSWAP(idx[0], nouns[0], nouns[1])
    } else {
        const mf = floor(nouns.len() / 2);
        const mc = ceil(nouns.len() / 2);

        for j in 0..mf {
            NounCSWAP(i[0], nouns[j], nouns[mc + j]);
        }

        P(idx[1..], nouns[..mc]);
    }
}
```

This has a gate complexity of $O(n\log(N))$. From this the parameterized partial permutation can be constructed as

```
fn NounMux(i, idx, text_idx, nouns) {
    const MAX_ARITY;

    for j in 0..MAX_ARITY {
        QRAMLookup(INDICES, text_idx, i * MAX_ARITY + j, idx);
        P(idx, nouns[j..]);
        QRAMLookup(INDICES, text_idx, i * MAX_ARITY + j, idx);
    }
}
```

and diagrammatically, this looks as follows:

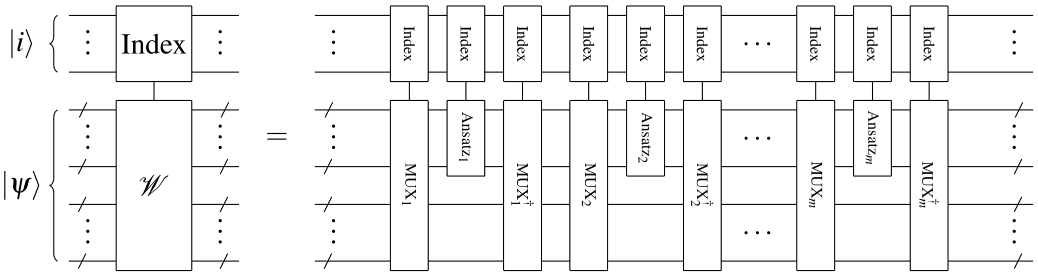


This has an overall gate complexity of $O(d(n\log(N)+\log(n)\log(M)))$.

Finally, by layering these two parts, we can construct the oracle. In this way, we first move the wires for each unitary to the top of the circuit, apply an ansatz (which applies the given unitary), and apply the inverse of the permutation. Then we repeat until no unitaries are remaining. In pseudocode, we have

```
fn Oracle(text_idx, nouns) {
    const MAX_PARAMS;
    const MAX_WORDS;
    const MAX_WIDTH;
    const MAX_NOUNS;
    const MAX_PREC;

    let idx = InitAncillas(MAX_NOUNS.log2());
    let angle = InitAncillas(MAX_PREC);
    let width = InitAncillas(MAX_WIDTH.log2());
    let ui_ancillas = InitAncillas(MAX_WIDTH.log2());
    for i in 0..MAX_WORDS {
        NounMux(i, idx, text_idx, nouns);

        PAnsatz(
            width, text_idx, i * MAX_PARAMS,
            angle, ui_ancillas, nouns[..MAX_ARITY]
        );

        NounMux'(i, idx, text_idx, nouns);
    }
    Discard(idx);
    Discard(angle);
    Discard(width);
    Discard(ui_ancillas);
}
```

and diagrammatically this looks like:



This has an overall gate complexity of

$$O(wd(n\log(N)+\log(n)\log(M)+A_g(N,d)+P\log(M)A_p(N,d)))$$

which can be simplified to:

$$O(wd\log(M)(n\log(N)+PA_g(N,d)))$$

Density Matrices for Metaphor Understanding

Jay Owers
SEMT, University of Bristol
Bristol, UK
jo16726@bristol.ac.uk

Ekaterina Shutova
ILLC, FNWI
University of Amsterdam
Amsterdam,
The Netherlands
e.shutova@uva.nl

Martha Lewis
SEMT, University of Bristol
Bristol, UK
Santa Fe Institute
Santa Fe, NM, USA
martha.lewis@bristol.ac.uk

In physics, density matrices are used to represent mixed states, i.e. probabilistic mixtures of pure states. This concept was used to model lexical ambiguity in [31]. In this paper, we consider metaphor as a type of lexical ambiguity, and examine whether metaphorical meaning can be effectively modelled using mixtures of word senses. We find that modelling metaphor is significantly more difficult than other kinds of lexical ambiguity, but that our best-performing density matrix method outperforms simple baselines as well as some neural language models.

1 Introduction

The use of vectors to model word meaning forms the basis of all modern approaches to modelling language. The idea behind this approach is called *distributional semantics* – using the distributions of words in text to build vectors that encode word meanings. When we have built vectors for words, say we have $|cat\rangle$, $|kitten\rangle$, and $|orthodontist\rangle$, the hope is that words that have similar meanings will be close together in the vector space, and that words that have dissimilar meanings are further apart, where we measure distance between words as the inner product of the normalised vectors, or equivalently as the cosine similarity, i.e., the cosine of the angle between the vectors. So, for example, we should have that:

$$\langle cat|kitten\rangle > \langle cat|orthodontist\rangle$$

However, representing words as vectors has some clear disadvantages. Firstly, words can have more than one meaning, and a naïve approach to building word vectors will try to pack all those meanings into one vector—ending up with a representation somewhere in the middle. Secondly, as well as representing word meanings we also wish to represent phrases, sentences, and paragraphs of text. However, simply representing words as vectors does not give us an obvious way of composing word vectors to produce phrase or sentence vectors.

There have been a number of approaches to representing ambiguity for word vectors. The task of discriminating different senses of a given word (word sense discrimination or WSD), is often framed as a classification task: given a sentence with a particular word, identify the sense of that word from its context [20, 36, 29]. An alternative task is to identify the sense of every word in a sentence. WSD can become a complex task, since each word must be disambiguated with respect to each other word in the sentence. One approach is simply to disambiguate word meanings in advance, and learn different vectors for different senses of a word. However, given that any word may have multiple meanings, this approach could generate a large number of possible meaning combinations that need to be disambiguated. Instead, we would like to be able to compose words together, and for a sentence to be disambiguated in the process of composition.

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 197–215, doi:10.4204/EPTCS.406.9

A key manifestation of ambiguity in language is the use of metaphor. Metaphor is pervasive in speech, with some estimates showing that we use metaphors on average every 3 sentences. As such, language models need to be able to deal with metaphor when it occurs. Many metaphorical uses are what is known as 'conventional' metaphor. This means that the metaphor has become entrenched in language, and can therefore be seen as a case of lexical ambiguity. For example, the most basic use of the word *bright* is as applied to colour. However, we often use this word to mean 'intelligent', as in *bright student*. This meaning of the word was created metaphorically from the original meaning of 'colourful', but has now become conventionalised in language.

The field of *compositional distributional semantics* [3, 10, 27] looks at systematic ways to compose word vectors together that are guided by our knowledge of grammatical composition. This paper works within the framework proposed by [10], which has fundamental links to quantum theory. Within [10], the structure of a sentence, together with the meanings of the words in the sentence, can be viewed as a tensor network creating the meaning of the sentence as a whole. This is further developed and indeed implemented on quantum computers in [23].

In the current work, we look at the ability of density matrices to represent the ambiguity represented by conventional metaphor. [31] show how density matrices and completely positive maps can be integrated with the compositional distributional semantics proposed by [10], and these methods are extended in [24]. The methods developed by [31] and applied in [24] are very successful, beating state of the art language models. We investigate whether density matrix methods are as effective in modelling conventional metaphor as they are in standard cases of lexical ambiguity.

In this paper, we firstly (section 2) give an overview of how density matrices have been used in NLP to model lexical entailment and semantics and syntactic ambiguity. In section 3 we review the quantum-inspired approach to NLP originally proposed by [10] which extends distributional semantics models to include composition, including how density matrices can be used in this compositional framework, and in section 4 how density matrices can be built automatically from text. We introduce a new dataset to test how well our implementations are able to model metaphor (Section 5), and finally, report results on this new dataset (Section 6), finding that metaphor interpretation is a difficult task for all models tested, although some density matrix methods perform above baseline.

2 Related Work: Density Matrices in Natural Language Processing

Two key uses of density matrices in NLP are 1) to model hyponymy and ambiguity, and 2) to model lexical ambiguity. We summarize research in these areas.

Density Matrices for Lexical Entailment Vectors are not well suited for representing hyponymy (is-a) relations between words. However, since density matrices can be ordered using a variant of the Löwner ordering [40], they are a candidate for representing these relations. This was investigated in [35], where density matrices were used to model hyponymy between words and phrases. This work models the strength of a hyponymy relation between two words as a function of the KL-divergence between the two matrices of the words. This measure has the nice property that hyponymy relationships between individual words lift to an entailment relationship between the two sentences. For example, given that *clarify* is a hyponym of *explain* and *rule* is a hyponym of *process*, we would expect that the phrase *clarify rule* entails *explain process*. In similar work, [2] model the relationship of hyponymy as the Löwner order between two matrices. The Löwner order states that $A \leqslant B$ iff $0 \leqslant B - A$. [2] provide a measure of graded hyponymy between two matrices which again lifts to entailment at the sentence level. [21] proposes a method for building density matrices using information from WordNet together with off-the shelf word vectors such as word2vec or GloVe. Density matrices for words can be composed to form density matrices for phrases and sentences by a variety of operators such as addition, pointwise multiplication, or more complex operators, such as BMult/Phaser [8, 21, 31] and KMult/Fuzz [8, 21]. These representations and composition operators work well on a simple entailment task from [17].

The use of density matrices to model logical and conversational entailment is developed further in [22, 34, 38], who develop a notion of negation for density matrices. [34, 38] extend these ideas to include the notion of conversational negation [18]. This can be thought of as modelling the acceptability of a sentence such as 'That's not a dog, it's a wolf' vs 'That's not a dog, it's a rainbow'. If we view negation as purely logical, then these sentences should be viewed as equally acceptable. In contrast, conversational negation provides a set of alternatives. [34, 38] model this by narrowing the subspace spanned by the logical negation to give a set of relevant alternatives. They furthermore provide alternative models of negation that utilize the Moore-Penrose pseudo-inverse of a matrix. In [13], the monotonicity of composition operators with respect to the Löwner order is investigated.

Another approach to building density matrices for entailment was developed by [5]. In this work, the authors model language as a set of sequences S from a given vocabulary, together with a probability distribution over this set. They form the free vector space over the set of sequences S, and then form a rank-1 density operator which encodes the probability distribution over sequences that models the language. Reduced density operators may be formed by tracing out over particular subspaces, and they show that in doing so, the hierarchy of subsequences is encoded by the Löwner order over these density matrices. This means that they can, for example, encode the fact that *black cats* are a subclass of *cats* - something that is not done by the approach outlined in [35] or [2].

Density Matrices for Ambiguity Using density matrices to model ambiguity in natural language is very natural. Density matrices were introduced in quantum physics to encode the notion of a mixed state: the case where the state of a system is not known. In this case, the system is encoded by taking a *mixture* of the possible states that it could be in. In the case of language, if we consider a word on its own, it may have multiple senses, and without context we cannot tell what the meaning of the word is. For example, *table* on its own can mean a piece of furniture, a structure for storing data, the act of presenting a topic, and so on. If we can store multiple different senses of a word in one representation, this can help to represent language.

One of the first uses of density matrices to represent ambiguity in text was [4]. This work aims to build representations of words that can encode multiple different kinds of word usage within one representation. For example, consider the word *table*. This word is semantically ambiguous: *table* means something different in 'Your dinner is on the table' vs. 'The data is in the table'. It is also syntactically ambiguous, since we can use it as a verb: 'table a motion'. The standard way of building word vectors forms a superposition of all these senses. [4] firstly build a space that takes into account the grammatical role of words, as encoded by dependency relations. They then form density matrices for individual words, based on the grammatical relations that word can participate in. The representations they learn are effective at the word level, but they do not give any methods for composing words together. [12] also use density matrix representations to model syntactic ambiguity. They show to to model alternative syntactic structures within one whole, and show how word representations can be composed.

[31] provide a thorough and elegant theoretical grounding that describes how to use the categorical compositional methods of [10] with density matrix representations of words. This will be described fully in subsequent sections. The main idea behind this is that words are represented as probabilistic mixtures

of their senses, and that when words are composed to make phrases, the phrase should disambiguate the meaning of the ambiguous word. The amount of ambiguity in a word can be represented as von Neumann entropy. [31] carry out corpus-based experiments, and show that the von Neumann entropy of word representations reduces in composition, indicating that the words have been disambiguated.

[24] extend the work of [31] to provide a means of building density matrices automatically from large scale text corpora. We describe this method in detail later in the paper. [24] test their density matrix representations on a range of datasets designed to test the ability of compositional models to disambiguate word meanings. Their representations outperform compositional baselines as well as state of the art large neural models.

3 Categorical Compositional Distributional Semantics

We work in the framework of categorical compositional distributional semantics [10]. In brief, words are represented as vectors inhabiting vector spaces that match their grammatical type. Setting a vector space N to be the noun type, and another space S to be the sentence type, we model nouns as vectors, adjectives as linear maps $adj : N \to N$ and verbs as multilinear maps from copies of N to S.

3.1 Pregroup Grammars

In order to describe grammatical structure we use Lambek's pregroup grammars [19]. A pregroup $(P, \leq, \cdot, 1, (-)^l, (-)^r)$ is a partially ordered monoid $(P, \leq, \cdot, 1)$ where each element $p \in P$ has a left adjoint p^l and a right adjoint p^r, such that the following inequalities hold:

$$p^l \cdot p \leq 1 \leq p \cdot p^l \quad \text{and} \quad p \cdot p^r \leq 1 \leq p^r \cdot p \tag{1}$$

We think of the elements of a pregroup as linguistic types. Concretely, we will use an alphabet $\mathscr{B} = \{n, s\}$. We use the type s to denote a declarative sentence and n to denote a noun. A transitive verb can then be denoted $n^r s n^l$. If a string of words and their types reduces to the type s, the sentence is judged grammatical. The sentence *Junpa loves cats* is typed $n\ (n^r s n^l)\ n$, and can be reduced to s as follows:

$$n\ (n^r s n^l)\ n \leq 1 \cdot s n^l n \leq 1 \cdot s \cdot 1 \leq s$$

3.2 Compositional Distributional Models

We interpret a pregroup grammar as a compact closed category, a structure shared by the category of finite dimensional Hilbert spaces. We briefly describe here the structure of a compact closed category, but for more details, please see [10, 32] and the introduction to relevant category theory given in [9].

Distributional vector space models live in the category **FHilb** of finite dimensional real Hilbert spaces and linear maps. **FHilb** is compact closed. Each object V is its own dual and the left and right unit and counit morphisms coincide. Given a fixed basis $\{|v_i\rangle\}_i$ of V, the unit η and counit ε are defined as:

$$\eta : \mathbb{R} \to V \otimes V :: 1 \mapsto \sum_i |v_i\rangle \otimes |v_i\rangle \qquad \varepsilon : V \otimes V \to \mathbb{R} :: \sum_{ij} c_{ij} |v_i\rangle \otimes |v_j\rangle \mapsto \sum_i c_{ii}$$

3.3 Grammatical Reductions in Vector Spaces

Following [32], reductions of the pregroup grammar are mapped into the category **FHilb** of finite dimensional Hilbert spaces and linear maps using a strong monoidal functor Q which preserves the compact closed structure:

$$\mathsf{Q} : \mathbf{Preg} \to \mathbf{FHilb}$$

We map noun and sentence types to appropriate finite dimensional vector spaces $\mathsf{Q}(n) = N$ $\mathsf{Q}(s) = S$, and concatenation in **Preg** is mapped to the tensor product in **FHilb**. Each type reduction α in the pregroup is mapped to a linear map in **FHilb**. Given a grammatical reduction $\alpha : p_1, p_2, ...p_n \to s$ and word vectors $|w_i\rangle$ with types p_i, a vector representation of the sentence $w_1w_2...w_n$ is given by:

$$|w_1w_2...w_n\rangle = \mathsf{Q}(\alpha)(|w_1\rangle \otimes |w_2\rangle \otimes ... \otimes |w_n\rangle)$$

We use the inner product to compare meanings of sentences by computing the cosine distance between sentence vectors. So, if sentence s has vector representation $|s\rangle$ and sentence s' has representation $|s'\rangle$, their degree of synonymy is given by:

$$\frac{\langle s|s'\rangle}{\sqrt{\langle s|s\rangle\, \langle s'|s'\rangle}}$$

3.4 Density Matrices in Categorical Compositional Distributional Semantics

Categorical compositional distributional semantics was extended in [31] to model nouns as density matrices in $N \otimes N$ and adjective and verbs as completely positive maps.

In distributional models of meaning, density matrices have been used in a variety of ways. We consider the meaning of a word w to be given by a collection of unit vectors $\{|w_i\rangle\}_i$. Each $|w_i\rangle$ is weighted by $p_i \in [0,1]$, such that $\sum_i p_i = 1$. Then the density operator:

$$[\![w]\!] = \sum_i p_i\, |w_i\rangle\, \langle w_i|$$

represents the word w. In [31], the vectors $\{|w_i\rangle\}_i$ are interpreted as senses of a given word, and we will use this interpretation later in the paper. In [2, 1, 21], the vectors $\{|w_i\rangle\}_i$ are interpreted as exemplars of a concept, and in [4] the $\{|w_i\rangle\}_i$ are interpreted as instances of use of a word.

3.5 The CPM Construction

Applying Selinger's CPM construction [37] to **FHilb** produces a new compact closed category in which the states are positive operators. This construction has previously been used in a linguistic setting in [16, 31, 1, 2, 21]. Throughout this section $\mathscr{C}$ denotes an arbitrary †-compact closed category.

Definition 1 (Completely positive morphism [37]) *A $\mathscr{C}$-morphism $\varphi : A^* \otimes A \to B^* \otimes B$ is said to be completely positive if there exists $C \in \mathsf{Ob}(\mathscr{C})$ and $k \in \mathscr{C}(C \otimes A, B)$, such that φ can be written in the form:*

$$(k_* \otimes k) \circ (1_{A^*} \otimes \eta_C \otimes 1_A)$$

Identity morphisms are completely positive, and completely positive morphisms are closed under composition in $\mathscr{C}$, leading to the following:

Definition 2 (CPM($\mathscr{C}$) [37]) *If $\mathscr{C}$ is a †-compact closed category then* **CPM**($\mathscr{C}$) *is a category with the same objects as $\mathscr{C}$ and its morphisms are the completely positive morphisms.*

The †-compact structure required for interpreting language in our setting lifts to **CPM**($\mathscr{C}$):

Theorem 1 (Compact Closure [37]) **CPM**($\mathscr{C}$) *is also a †-compact closed category. There is a functor:*

$$\mathsf{E} : \mathscr{C} \to \mathbf{CPM}(\mathscr{C})$$
$$k \mapsto k_* \otimes k$$

This functor preserves the †-compact closed structure, and is faithful "up to a global phase".

3.5.1 Sentence Meaning in the category $\mathbf{CPM}(\mathbf{FHilb})$

In the vector space model of distributional models of meaning the movement from syntax to semantics was achieved via a strong monoidal functor $\mathsf{Q} : \mathbf{Preg} \to \mathbf{FHilb}$. Language can be assigned semantics in $\mathbf{CPM}(\mathbf{FHilb})$ in an entirely analogous way via a strong monoidal functor:

$$\mathsf{S} : \mathbf{Preg} \to \mathbf{CPM}(\mathbf{FHilb})$$

If $w_1, w_2...w_n$ is a string of words with corresponding grammatical types t_i in $\mathbf{Preg}_{\mathscr{B}}$. and the type reduction is given by $t_1,...t_n \xrightarrow{r} x$ for some $x \in \mathsf{Ob}(\mathbf{Preg}_{\mathscr{B}}$, where $[\![w_i]\!]$ is the meaning of word w_i in $\mathbf{CPM}(\mathbf{FHilb})$, i.e. a density matrix ρ_i. Then the meaning of $w_1 w_2...w_n$ is given by:

$$[\![w_1 w_2...w_n]\!] = \mathsf{S}(r)([\![w_1]\!] \otimes ... \otimes [\![w_n]\!])$$

We render semantic similarity of representations as the generalised inner product, i.e. $\text{Tr}([\![w_1]\!]^\dagger [\![w_2]\!])$, as do [31]. This notion of semantic similarity is modulated by the extent of the ambiguity of a representation. If a representation is maximally ambiguous, that is, $[\![w_1]\!] = \mathbb{I}/n$, then the similarity of $[\![w_1]\!]$ with itself is only $1/n$. We choose to interpret this as reflecting the fact that the meaning of of w_1 is undetermined without further context, and hence that the two instances of w_1 being compared could in fact have different senses. On the other hand, if $[\![w_1]\!]$ is pure, then self-similarity is 1, as expected.

We now go on to describe how density matrices can be learnt directly from text corpora, and composed to form sentence representations. We will assess these representations in a setting requiring metaphor interpretation.

We now have all the ingredients to derive sentence meanings in $\mathbf{CPM}(\mathbf{FHilb})$.

Example 1 We firstly show that the results from $\mathbf{FHilb}$ lift to $\mathbf{CPM}(\mathbf{FHilb})$. Let the noun space N be a real Hilbert space with basis vectors given by $\{|n_i\rangle\}_i$, where for some i, $|n_i\rangle = |shoulders\rangle$ Let the sentence space be another space S with basis $\{|s_i\rangle\}_i$. The verb $|slouch\rangle$ is given by:

$$|slouch\rangle = \sum_{pq} C_{pq} |n_p\rangle \otimes |s_q\rangle$$

The density matrix for the noun *shoulders* is in fact a pure state given by:

$$[\![shoulders]\!] = |n_i\rangle \langle n_i|$$

and similarly, $[\![slouch]\!]$ in $\mathbf{CPM}(\mathbf{FHilb})$ is:

$$[\![slouch]\!] = \sum_{pqtu} C_{pq} C_{tu} |n_p\rangle \langle n_t| \otimes |s_q\rangle \langle s_u|$$

The meaning of the composite sentence is simply $(\varepsilon_N \otimes 1_S)$ applied to $([\![shoulders]\!] \otimes [\![slouch]\!])$. This corresponds to:

$$\begin{aligned}[\![shoulders\ slouch]\!] &= \varphi([\![shoulders]\!] \otimes [\![slouch]\!]) \\ &= \sum_{qu} C_{iq} C_{iu} |s_q\rangle \langle s_u|\end{aligned}$$

This is a pure state corresponding to the vector $\sum_q C_{iq} |s_q\rangle$.

We can also deal with mixed states.

Example 2 Let the noun space N be a real Hilbert space with basis vectors given by $\{|n_i\rangle\}_i$. Consider two senses of the word *shoulder* meaning 1) a part of your body and 2) the edge of a road. Let:

$$|shoulder_{body}\rangle = \sum_i a_i |n_i\rangle\,,\ |shoulder_{road}\rangle = \sum_i b_i |n_i\rangle$$

and with the sentence space S, consider the word *slump* with the senses *slouch* and *decline* we define:

$$|slump_{slouch}\rangle = \sum_{pqr} C_{pqr} |n_p\rangle \otimes |s_q\rangle$$
$$|slump_{decline}\rangle = \sum_{pqr} D_{pqr} |n_p\rangle \otimes |s_q\rangle$$

We set:

$$[\![shoulder]\!] = \frac{1}{2}(|shoulder_{body}\rangle \langle shoulder_{body}| + |shoulder_{road}\rangle \langle shoulder_{road}|)$$
$$[\![slump]\!] = \frac{1}{2}(|slump_{slouch}\rangle \langle slump_{slouch}| + |slump_{decline}\rangle \langle slump_{decline}|)$$

Then, the meaning of the sentence:

$$s = Shoulders\ slump$$

is given by:

$$[\![s]\!] = (\varepsilon_N \otimes 1_S \otimes \varepsilon_N)([\![shoulders]\!] \otimes [\![slump]\!]$$

In the example above, we have a two word sentence where each word has two interpretations. There are therefore 4 possible assignments of senses to the words. However, only one assignment of words makes sense in context. The aim is for the correct senses of each word to be picked out in composition.

4 Methods

4.1 Learning Density Matrices from Text

[24] introduce a method for learning density matrices from text called Multi-sense Word2DM. This is an extension of the word2vec skipgram with negative sampling (SGNS) [25] to density matrices. word2vec learns vectors for words by running through a large corpus of text and updating word vectors according to the following objective function with regard to model parameters θ:

$$J(\theta) = \log \sigma(\langle v_t|v_c\rangle) + \sum_{k=1}^{K} \log \sigma(-\langle v_t|v_k\rangle) \tag{2}$$

where v_t is the embedding of target word, v_c is the embedding of the context word, $v_1, v_2, ..., v_K$ are the embeddings of K negative samples, and σ is the logistic function. Maximising equation 2 adjusts the embeddings of words occurring in the same context to be more similar and adjusts the embeddings of words that don't occur together to be less similar. Multi-sense Word2DM is a modification of SGNS for density matrices. Instead of learning one vector per word, multiple sense embeddings are learnt and then combined together to form a density matrix. Each sense of a word has its own n-dimensional embedding. A density matrix can be expressed in terms of the sense embeddings as

$$A = BB^{\dagger} = \sum_{i=1}^{m} |b_i\rangle \langle b_i| \tag{3}$$

where $|b_1\rangle, ..., |b_m\rangle$ are the columns of B corresponding to different senses. Each word is also associated with a single vector v_w, which represents it as a context word. The following objective function is maximised:

$$J(\theta) = \log\sigma(\langle b_t|c_t\rangle) + \sum_{k=1}^{K} \log\sigma(-\langle b_t|v_{w_k}\rangle) \tag{4}$$

where $|c_t\rangle$ is the sum of context vectors for all words surrounding the target word and $|b_t\rangle$ is the the embedding for the relevant sense of the target word. We select $|b_t\rangle$ by finding the column of B_t most similar to $|c_t\rangle$ (measured by cosine similarity). Multi-sense Word2DM explicitly models ambiguity by letting the columns of the intermediary matrix represent the different senses of a word. During training the column closest to the context embedding is selected as the relevant sense embedding and only this column is updated.

4.2 Composition Methods

The methods used by [24] to build density matrices only build matrices that inhabit a single space $W \otimes W$, rather than the larger spaces needed for verbs and adjectives. Because of this, [24] use a set of composition methods that can be seen as 'lifting' a given word representation to the type needed for composition. These are based on methods in [21, 8], and we will use these in section 6. We view relational words such as adjectives or verbs as maps that takes nouns as arguments. The composition methods are as follows, using the example of an adjective modifying a noun:

- Add: $[\![adj]\!] + [\![noun]\!]$
- Mult: $[\![adj]\!] \odot [\![noun]\!]$
- Fuzz: $\sum_i p_i P_i [\![noun]\!] P_i$, where $\sum_i p_i P_i$ is the spectral decomposition of $[\![adj]\!]$
- Phaser: $[\![adj]\!]^{1/2} [\![noun]\!] [\![adj]\!]^{1/2}$

More complex phrases are combined according to their parse. So, a transitive sentence modified with an adjective is composed as (subj(verb(adj obj))). Composing e.g. the sentence *Junpa likes stripy cats* would consist of the following steps:

$$[\![stripy\ cats]\!] = f([\![stripy]\!], [\![cats]\!]) \tag{5}$$

$$[\![likes\ stripy\ cats]\!] = f([\![likes]\!], [\![stripy\ cats]\!]) \tag{6}$$

$$[\![Junpa\ likes\ stripy\ cats]\!] = f([\![Junpa]\!], [\![likes\ stripy\ cats]\!]) \tag{7}$$

where the composer f can be substituted by any of the composition methods listed above.

5 Implementation

5.1 Datasets and Tasks

We build a novel dataset to test disambiguation in a metaphorical context. The basic structure of the dataset is as follows. Given a target sentence that uses a metaphorical word, we minimally alter the target sentence to provide an apt literal paraphrase and an inapt paraphrase of the sentence. We attempt to replace only the single metaphorically used word, although this is not always possible.

Example

- Target sentence: He showered her with presents
- Apt paraphrase: He gave her presents
- Inapt paraphrase: He sprinkled her with presents

The expectation is that the apt paraphrase is semantically closer to the target sentence than the inapt sentence is. So, given representations $[\![target]\!]$, $[\![apt]\!]$, $[\![inapt]\!]$, $\mathrm{Tr}([\![target]\!]^\dagger [\![apt]\!]) > \mathrm{Tr}([\![target]\!]^\dagger [\![inapt]\!])$.

We use the metaphorical sentences and their literal paraphrases from [28], and generate inapt paraphrases as follows. Each target sentence includes one metaphorically used verb. We use WordNet [26] to determine the most commonly used sense of the verb. WordNet is a resource listing words, their different senses (called ‘synsets’), synonyms within each synset, and hyponym, hypernym and other relations between words. The synsets are listed in order of frequency, so that the first synset is the most commonly used sense of a word. We expect that the most commonly used sense of a word will be a literal sense, and that taking a synonym from this sense will form an inapt paraphrase, as in the example above. We manually inspect each candidate inapt paraphrase, and if the paraphrase is not inapt, we generate a new paraphrase by looking at synonyms of the hypernym of the metaphorically used word. We generated inapt paraphrases for a total of 171 metaphorical sentences.

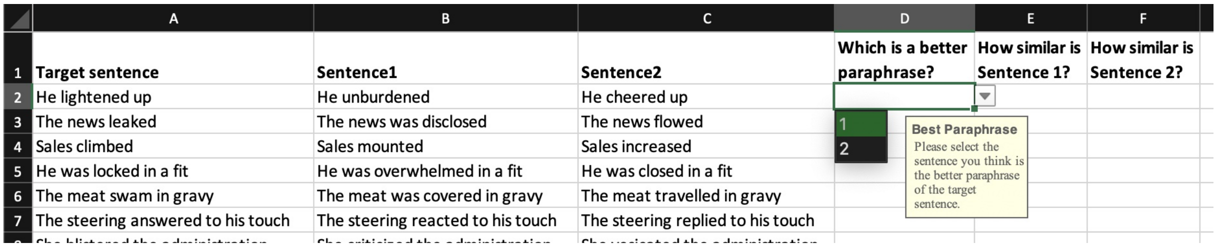
We then simplified the sentences to consist of just subject-verb (SV), verb-object (VO) or subject-verb-object (SVO) fragments, and subsequently padded these fragments with pronouns and determiners to generate simplified versions of the full sentences. This usually just involved a shortening of the sentence. We test our models on two versions of the dataset. The dataset in the format of SV/VO/SVO fragments we call the **short-form dataset**, and the dataset with full but simplifed sentences we call the **long-form dataset**.

Simplification Procedure

- Original sentence: He **wasted** his inheritance on his insincere friends.
- VO format: waste inheritance
- Simplified sentence: She wasted her inheritance

Human Annotation We solicited annotations from humans as follows. We gave them triples of target sentence, apt paraphrase, and inapt paraphrase. The apt and inapt paraphrases were presented in a random order: sometimes the apt was presented first, and sometimes the inapt. Participants were asked to state which was the best paraphrase of the target sentence, and then to rate the similarity of each paraphrase to the target sentence. All participants were asked to annotate all 171 triples, resulting in 10 annotations per triple. Triples were presented in a random order for each participant. Mean inter-annotator agreement, calculated using Spearman’s rho over all pairs of annotators, was 0.589, which is a reasonable level. Examples of the annotation task are shown in figures 1 and 2

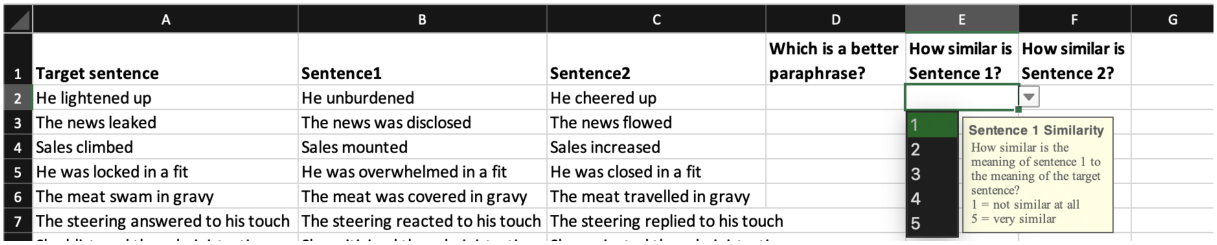
Model Task We evaluate a range of density matrix models, neural models, and baselines on this disambiguation task. The evaluation runs as follows. For each triple of sentences, we generate vectors or density matrices for each sentence. We then compute the similarity of the sentence representations using either cosine similarity (for vectors) or the generalized inner product (for density matrices. We assess the extent to which models agree with the mean of the human judgements using Spearman’s rho.



	A	B	C	D	E	F
1	Target sentence	Sentence1	Sentence2	Which is a better paraphrase?	How similar is Sentence 1?	How similar is Sentence 2?
2	He lightened up	He unburdened	He cheered up			
3	The news leaked	The news was disclosed	The news flowed			
4	Sales climbed	Sales mounted	Sales increased			
5	He was locked in a fit	He was overwhelmed in a fit	He was closed in a fit			
6	The meat swam in gravy	The meat was covered in gravy	The meat travelled in gravy			
7	The steering answered to his touch	The steering reacted to his touch	The steering replied to his touch			
8	She blistered the administration	She criticized the administration	She vesicated the administration			

Best Paraphrase
Please select the sentence you think is the better paraphrase of the target sentence.

Figure 1: Participants are asked to choose which of paraphrase 1 or 2 is the best.



	A	B	C	D	E	F	G
1	Target sentence	Sentence1	Sentence2	Which is a better paraphrase?	How similar is Sentence 1?	How similar is Sentence 2?	
2	He lightened up	He unburdened	He cheered up				
3	The news leaked	The news was disclosed	The news flowed				
4	Sales climbed	Sales mounted	Sales increased				
5	He was locked in a fit	He was overwhelmed in a fit	He was closed in a fit				
6	The meat swam in gravy	The meat was covered in gravy	The meat travelled in gravy				
7	The steering answered to his touch	The steering reacted to his touch	The steering replied to his touch				
8	She blistered the administration	She criticized the administration	She vesicated the administration				

Sentence 1 Similarity
How similar is the meaning of sentence 1 to the meaning of the target sentence?
1 = not similar at all
5 = very similar

Figure 2: Participants are asked to rate the similarity of each paraphrase to the target sentence

5.2 Models

We use the density matrices computed in [24]. We test 4 variants of **Multi-sense Word2DM** (ms-Word2DM). One parameter varies the number of senses computed for each matrix, either 5 or 10, and one varies the means by which the closest sense is chosen for updating, choosing the closest via cosine similarity (c) or by Euclidean distance (d).[24] also propose two other key ways of generating density matrices. One, called **Context2DM**, extends the methods proposed in [31, 36]. Firstly, a set of word vectors is trained with the gensim implementation[1] of Word2Vec on the combined ukWaC+Wackypedia corpus. For each target word, the set of words whose context they appear in is collected. The vectors of these context words are clustered using hierarchical agglomerative clustering to between 2 and 10 clusters. The centroids of each of these clusters are taken to form the sense vectors for the target word, and subsequently these sense vectors are combined into a density matrix.

The second alternative method, **Bert2DM** uses BERT [14] to generate sense vectors. BERT is a Transformer-based architecture [39] which produces contextual embeddings, meaning that the vector produced for each word is dependent on the context of the sentence. A small corpus is fed through BERT. Vectors for each word are extracted, and dimensionality reduction applied (either PCA or SVD). Vectors for each word are then combined to form density matrices.

We compare performance with two neural sentence encoders, **SBERT** [33] and **InferSent** [11]. Both of these models are explicitly trained to predict when one sentence can be inferred from another. Since synonymy is a simple kind of entailment relation, these sentence encoders should perform well.

We compare with two baseline models, **word2vec** [25] and **GloVe** [30]. These are two methods for producing static word vectors with the key property that semantically similar words should be close together in the vector space.

[1] https://radimrehurek.com/gensim/models/word2vec

6 Experiments and Results

For each model described in the previous section, we generate sentence embeddings for triples in the dataset. For the density matrix methods, we apply the composition methods outlined in 4.2, that is, Add, Mult, Fuzz, and Phaser. For neural methods, we simple take the sentence embeddings produced by the models. For the static word vector baselines, we apply Add and Mult. We also use a baseline 'composition' method of simply taking the verb as sentence representation (Verb only)

We compute $sim(\llbracket target \rrbracket, \llbracket apt \rrbracket)$ and $sim(\llbracket target \rrbracket, \llbracket inapt \rrbracket)$, where sim is cosine similarity for vectors and generalized inner product for density matrices. We compare the similarity scores generated by the models to the similarity scores generated by humans and compute the correlation between these scores using Spearman's rho. Spearman's rho ranges between -1 and 1, with 1 indicating that scores are perfectly correlated and -1 indicating that they are perfectly anti-correlated.

For each model tested, there were a number of occurrences of words in the dataset that were not in the lexicon of the trained model. Where this occurred, the sentence pairs containing these words were removed from the tests for the given model. The error occurred more frequently with the long-form dataset due to words not being in their lemmatised form. For Context2DM and BERT2DM with the long-form dataset, the results were deemed unreliable and not included due to having only 8 usable sentence pairs. Table 6 in the Appendix shows the number of sentence pairs used in the tests, out of the total of 342 sentence pairs, as a result of these errors being removed.

6.1 Results

Results are shown in tables 1, 2 and 3. Table 1 shows firstly that the neural sentence encoders (left hand table) were not able to correctly interpret the metaphorically used word in context, with correlation close to 0, i.e. indicating chance level. Secondly, on the right hand side, we see that in the compositional settings (Add and Mult), static word vectors also perform poorly, although very slightly better than the sentence encoders. For the Verb-only setting, we see a stronger negative correlation, i.e., similarity ratings are more consistently in the wrong direction. This is expected, as the verb alone does not provide the context necessary for disambiguation of the metaphor, and we would expect that the most common sense of a word would dominate the learnt representations.

SBERT	short	-0.0317
	long	-0.0146
Infersent1	short	0.0085
	long	-0.0027
Infersent2	short	0.0113
	long	-0.0411

		Verb	Add	Mult
Word2Vec	short	-0.2719	-0.0536	-0.1092
	long		-0.0010	-0.0642
GloVe	short	-0.1529	-0.0272	0.0738
	long		0.0236	0.0579

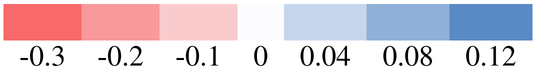


Table 1: Spearman's rho for sentence encoders and vector baselines. Note that all sentence encoders performed poorly, while Glove with mult performed better.

In Table 2, we see a similar pattern. The verb-only baseline produces similarity ratings that are negatively correlated with those of humans. In general, the BERT2DM and Context2DM models produce slightly stronger results. The highest correlation is produced by Multi-sense Word2DM with 10 senses, using Euclidean distance to choose which sense to update, and using Mult as the composition operator. Overall, correlation is low. Table 3 gives performance of the density matrix models when using Fuzz or Phaser as the composition operator. When applying Fuzz and Phaser, there is a choice of whether to use the verb as the operator or the noun as the operator. The linguistically motivated choice is to use the verb as the operator. However, as seen in table 3, using the noun as the operator produces a better correlation with human judgements. We speculate that when using the verb as operator, the most common sense of the verb dominates the composition. In the case of a metaphorical verb, we require that the noun modifies the meaning of the verb to obtain the correct interpretation.

		Verb	Add	Mult
ms-Word2DM-c5	short	-0.2062	-0.0587	0.0201
	long		0.0373	-0.0764
ms-Word2DM-c10	short	-0.2094	-0.0552	-0.0365
	long		0.0450	-0.0577
ms-Word2DM-d5	short	-0.2519	-0.0999	-0.0204
	long		0.0136	0.0385
ms-Word2DM-d10	short	-0.2102	-0.0784	-0.0609
	long		0.0432	0.1061
Word2DM	short	-0.1772	-0.1063	-0.0004
	long		-0.0057	0.0263
bert2dm-pca-cls	short	0.0120	0.0749	0.0262
bert2dm-svd-cls	short	-0.0296	0.0172	0.0698
context2dm	short	0.0240	0.0483	0.0885

🟥	🟥	🟥	⬜	🟦	🟦	🟦
-0.3	-0.2	-0.1	0	0.04	0.08	0.12

Table 2: Spearman's rho for density matrix models with simple composition. Most models with verb-only composition had negative correlation. ms-Word2DM-d10 with mult-long had the best performance.

All models except for BERT2DM-svd-cls and Context2DM consistently show an increase in rho after composition, supporting the idea that interpretation of metaphor is easier when provided with context.

6.2 Analysis

To understand more about which sentences were being scored correctly, we examined the responses of ms-word2dm-d10. Each instance of a metaphorical sentence and its two paraphrases was marked correct if the apt paraphrase scored higher on cosine similarity than the inapt paraphrase, and incorrect if the inapt paraphrase scored highest. These results were then compiled for different composition methods and compared with the verb baseline. The number of instances correct and incorrect with verb and composition are shown in figure 3. We see that overall, the majority of similarity judgements are incorrect for verb-only models and for compositional models (yellow bar). We also see that for many models, both the verb-only and the compositional models are correct (light green bar). For Fuzz and Phaser, models where the verb is the operator showed fewer instances where the result changes after composition. This means that in more cases, the result remained either correct or incorrect according to the result of the

	Fuzz verb	Fuzz noun	Phaser verb	Phaser noun
ms-Word2DM-c5	-0.1732	0.0378	-0.0251	0.0148
ms-Word2DM-c10	-0.1681	-0.0146	-0.0949	-0.0160
ms-Word2DM-d5	-0.1783	0.0576	-0.1132	0.0121
ms-Word2DM-d10	-0.1997	0.0402	-0.1552	-0.0125
Word2DM	-0.1042	-0.0008	-0.1311	-0.0029
bert2dm-pca-cls	0.0217	0.0274	0.0337	0.0186
bert2dm-svd-cls	-0.0280	0.0377	-0.0451	0.0358
context2dm	0.0177	0.0448	-0.0088	0.0350

-0.3 -0.2 -0.1 0 0.04 0.08 0.12

Table 3: Spearman's rho for density matrix models with Fuzz and Phaser composition on the short-form dataset. Verb operator models generally had poor performance while noun operator models, especially with Fuzz, performed better.

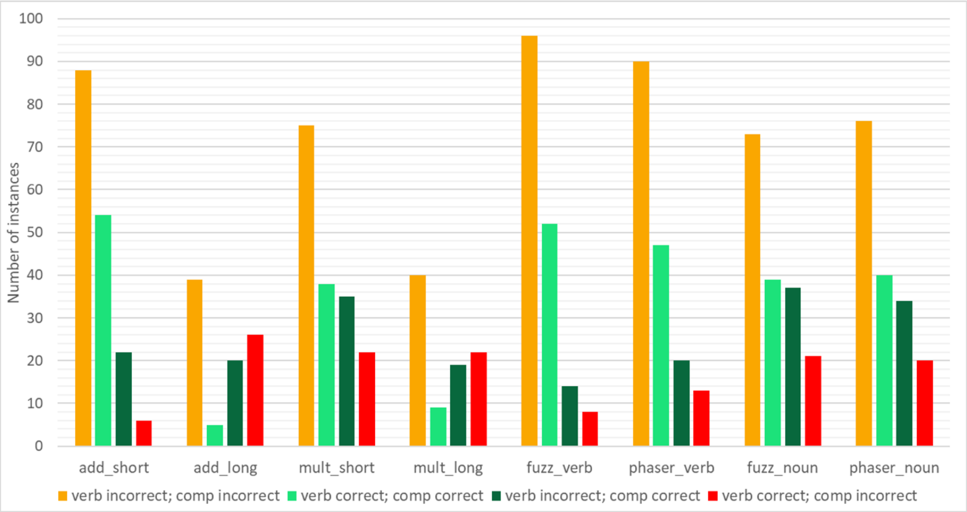


Figure 3: Analysis of which sentences were scored correctly by ms-Word2DM-d10. Note that verb operator models scored a lot of sentences the same as the verb, meaning composition had little effect.

verb alone. In contrast, the noun operator models had more instances where the verb-only model was incorrect and the composition was correct (dark green bar). This also indicates that the result is being heavily influenced by the operator matrix. Finally, there are a lot of instances where the verb-only model produces the correct answer but this is ‘undone’ by the composition (red bar). For Add and Mult, the long-form sentences showed a greater proportion of the correct sentences using the verb that became incorrect after composition, suggesting that the extra words in the long sentences may be distracting from the relevant information.

To analyse how each composition method affects the modelled ambiguity, we calculated the von Neumann entropy of each of the sentences for each composition method including the verb baseline, as shown in tables 4 and 5. The relative difference in entropy between the verb and the sentence embedding indicates the model’s change in ambiguity upon composition of the sentence. This is shown by the colour of the cells, red meaning an increase in entropy and blue meaning a decrease. The entropy results are consistent with the results for the similarity task; Phaser resulted in a greater reduction in entropy than Fuzz, and also produced stronger Spearman correlation. BERT2DM-svd-cls mult-short also had a high rho and a strong reduction in entropy.

		Verb	Add	Mult
ms-Word2DM-c5	short	1.1464	1.9298	1.6927
	long	1.1464	1.7579	1.8508
ms-Word2DM-c10	short	1.3230	2.2118	1.6821
	long	1.3230	1.9797	1.9587
ms-Word2DM-d5	short	1.0590	1.9834	1.4786
	long	1.0590	1.7171	1.6482
ms-Word2DM-d10	short	1.4508	2.1462	1.6160
	long	1.4508	2.0085	1.8854
Word2DM	short	0.1799	1.5876	0.9849
	long	0.1799	1.0555	0.4431
bert2dm-pca-cls	short	0.4647	1.0575	0.4467
bert2dm-svd-cls	short	0.2346	0.4943	0.0339
context2dm	short	0.2877	0.4546	0.3097

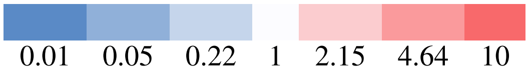


Table 4: Mean von Neumann entropy produced by density matrix models with simple composition. Colour indicates the relative change in entropy with composition. Bert2dm-svd-cls with mult is the only model to significantly decrease entropy.

7 Discussion and Conclusions

Interpreting metaphor correctly is an essential part of language understanding. Previous work showed that density matrix approaches to modelling language meaning could effectively capture ambiguity. The aim of this paper was to test whether these representations could capture ambiguity in the case of conventional metaphor. We built a new dataset to test our models, and tested a range of density matrix word

	Verb	Fuzz verb	Fuzz noun	Phaser verb	Phaser noun
ms-Word2DM-c5	1.1464	0.6781	0.6731	0.3661	0.3550
ms-Word2DM-c10	1.3230	0.8769	0.9141	0.4971	0.4860
ms-Word2DM-d5	1.0590	0.7257	0.6873	0.3046	0.2798
ms-Word2DM-d10	1.4508	0.8580	0.8412	0.4034	0.3918
Word2DM	0.1799	0.2896	0.2519	0.0511	0.0413
bert2dm-pca-cls	0.4647	0.3001	0.3182	0.1245	0.1269
bert2dm-svd-cls	0.2346	0.0444	0.0546	0.0103	0.0103
context2dm	0.2877	0.0912	0.0389	0.0041	0.0041

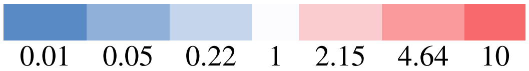


Table 5: Mean von Neumann entropy produced by density matrix models with Fuzz and Phaser on short-form dataset. Colour indicates the relative change in entropy with composition. Note that Phaser caused a greater entropy reduction than Fuzz.

representations and composition methods, as well as neural sentence encoders. All models found this dataset extremely difficult. However, we do see some insights into how metaphor can be interpreted. Firstly, in almost all compositional methods, we see an increase in performance over a simple verb-only baseline. We also see some increases in performance over simple vector-based models. We find that using the non-metaphorical nouns as operators rather than the metaphorical verbs gives improved performance, indicating that perhaps we need to view the metaphorically used words as being updated by their context. We also see that in the case of Fuzz and Phaser composition, the composition reduces the entropy of representations, as previously seen in [31, 24]

Further Work We have tested our dataset on SBERT and InferSent, however, recently, a wide range of language models have become available, and we plan to test those models on this difficult dataset. The work we have presented in this paper relates only to conventional metaphor, which can be seen fairly straightforwardly as a case of lexical ambiguity. Work is ongoing to extend these methods to novel metaphor, where new meanings must be created on-the-fly. Another line of enquiry is to link the interpretation of density matrices as modelling ambiguity with the interpretation as modelling hyponymy. The categorisation theory of metaphor proposed in [15] argues that in noun-noun metaphors (e.g. 'My lawyer is a shark'), an ad-hoc class (e.g. 'vicious things') is created that abstracts both the metaphor and its target. We can therefore use the entailment interpretation of density matrices to discover these ad-hoc classes by finding something like a greatest lower bound for downward entailment – with the caveat that such bounds are not always unique for density matrices. There is potential to integrate these methods with the DiscoCirc formalism [7], in which a sense of narrative is included, and the level of modelling is at the full text rather than sentence level. Finally, implementing quantum-inspired models using real quantum computers is a burgeoning new line of research [23] and our methods have already been investigated in simulation in [6]. Pushing these ideas further will be an important step in this process.

References

[1] Esma Balkır, Mehrnoosh Sadrzadeh & Bob Coecke (2016): *Distributional Sentence Entailment Using Density Matrices*. In Mohammad T. Hajiaghayi & Mohammad R. Mousavi, editors: *Topics in Theoretical Computer Science*, *Lecture Notes in Computer Science* 9541, Springer, Cham, pp. 1–22. Available at `https://doi.org/10.1007/978-3-319-28678-5_1`.

[2] Dea Bankova, Bob Coecke, Martha Lewis & Dan Marsden (2019): *Graded Hyponymy for Compositional Distributional Semantics*. *Journal of Language Modelling* 6(2), p. 225, doi:10.15398/jlm.v6i2.230. Available at `http://jlm.ipipan.waw.pl/index.php/JLM/article/view/230`.

[3] Marco Baroni & Roberto Zamparelli (2010): *Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space*. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 1183–1193. Available at `https://dl.acm.org/doi/abs/10.5555/1870658.1870773`.

[4] William Blacoe, Elham Kashefi & Mirella Lapata (2013): *A Quantum-Theoretic Approach to Distributional Semantics*. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Atlanta, Georgia, pp. 847–857. Available at `https://aclanthology.org/N13-1105`.

[5] Tai-Danae Bradley & Yiannis Vlassopoulos (2021): *Language Modeling with Reduced Densities*. *Compositionality* 3, p. 4. Available at `https://doi.org/10.32408/compositionality-3-4`.

[6] Saskia Bruhn (2021): *Density Matrix Methods in Quantum Natural Language Processing*. Ph.D. thesis, Master's thesis, Universität Osnabrück, 2022. doi: 10.48693/111.

[7] Bob Coecke (2021): *The mathematics of text structure*. *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pp. 181–217. Available at `https://doi.org/10.48550/arXiv.1904.03478`.

[8] Bob Coecke & Konstantinos Meichanetzidis (2020): *Meaning Updating of Density Matrices*. *Journal of Applied Logics* 2631(5), p. 745. Available at `https://doi.org/10.48550/arXiv.2001.00862`.

[9] Bob Coecke & Éric Oliver Paquette (2011): *Categories for the practising physicist*. In: *New Structures for Physics*, Springer, pp. 173–286. Available at `https://doi.org/10.1007/978-3-642-12821-9_3`.

[10] Bob Coecke, Mehrnoosh Sadrzadeh & Stephen J Clark (2010): *Mathematical Foundations for a Compositional Distributional Model of Meaning*. *Linguistic Analysis* 36(1), pp. 345–384. Available at `https://doi.org/10.48550/arXiv.1003.4394`.

[11] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault & Antoine Bordes (2017): *Supervised Learning of Universal Sentence Representations from Natural Language Inference Data*. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, doi:10.18653/v1/D17-1070.

[12] Adriana D. Correia, Michael Moortgat & Henk T. C. Stoof (2020): *Density Matrices with Metric for Derivational Ambiguity*, doi:10.48550/arXiv.1908.07347. Available at `http://arxiv.org/abs/1908.07347`.

[13] Gemma De las Cuevas, Andreas Klingler, Martha Lewis & Tim Netzer (2020): *Cats Climb Entails Mammals Move: Preserving Hyponymy in Compositional Distributional Semantics*. *arXiv:2005.14134 [cs, math]*. Available at `https://doi.org/10.48550/arXiv.2005.14134`.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee & Kristina Toutanova (2019): *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In Jill Burstein, Christy Doran & Thamar Solorio, editors: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186, doi:10.18653/v1/N19-1423. Available at `https://aclanthology.org/N19-1423`.

[15] Sam Glucksberg & Boaz Keysar (1990): *Understanding metaphorical comparisons: Beyond similarity*. *Psychological review* 97(1), p. 3. Available at `https://doi.org/10.1037/0033-295X.97.1.3`.

[16] Dimitri Kartsaklis (2015): *Compositional Distributional Semantics with Compact Closed Categories and Frobenius Algebras*. Ph.D. thesis, University of Oxford. Available at https://arxiv.org/abs/1505.00138.

[17] Dimitri Kartsaklis, Matthew Purver & Mehrnoosh Sadrzadeh (2016): *Verb Phrase Ellipsis using Frobenius Algebras in Categorical Compositional Distributional Semantics*. In: DSALT Workshop, European Summer School on Logic, Language and Information, pp. 1–2. Available at https://www.eecs.qmul.ac.uk/~mpurver/papers/kartsaklis-et-al16dsalt.pdf.

[18] Germán Kruszewski, Denis Paperno, Raffaella Bernardi & Marco Baroni (2016): *There Is No Logical Negation Here, But There Are Alternatives: Modeling Conversational Negation with Distributional Semantics*. Computational Linguistics 42(4), pp. 637–660, doi:10.1162/COLI_a_00262. Available at https://aclanthology.org/J16-4003.

[19] Joachim Lambek (1997): *Type Grammar Revisited*. In Alain Lecomte, François Lamarche & Guy Perrier, editors: Logical Aspects of Computational Linguistics, Second International Conference, LACL '97, Nancy, France, September 22-24, 1997, Selected Papers, Lecture Notes in Computer Science 1582, Springer, pp. 1–27. Available at https://doi.org/10.1007/3-540-48975-4_1.

[20] Michael Lesk (1986): *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*. In: Proceedings of the 5th annual international conference on Systems documentation, SIGDOC '86, Association for Computing Machinery, New York, NY, USA, pp. 24–26, doi:10.1145/318723.318728. Available at https://dl.acm.org/doi/10.1145/318723.318728.

[21] Martha Lewis (2019): *Compositional Hyponymy with Positive Operators*. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), INCOMA Ltd., Varna, Bulgaria, pp. 638–647, doi:10.26615/978-954-452-056-4_075. Available at https://www.aclweb.org/anthology/R19-1075.

[22] Martha Lewis (2020): *Towards Logical Negation for Compositional Distributional Semantics*. IfCoLog Journal of Applied Logics 7(5), pp. 771–794. Available at https://doi.org/10.48550/arXiv.2005.04929.

[23] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis & Bob Coecke (2023): *QNLP in practice: Running compositional models of meaning on a quantum computer*. Journal of Artificial Intelligence Research 76, pp. 1305–1342, doi:10.1613/jair.1.14329.

[24] Francois Meyer & Martha Lewis (2020): *Modelling Lexical Ambiguity with Density Matrices*. In: Proceedings of the 24th Conference on Computational Natural Language Learning, Association for Computational Linguistics, Online, pp. 276–290, doi:10.18653/v1/2020.conll-1.21. Available at https://www.aclweb.org/anthology/2020.conll-1.21.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado & Jeff Dean (2013): *Distributed representations of words and phrases and their compositionality*. Advances in neural information processing systems 26. Available at https://doi.org/10.48550/arXiv.1310.4546.

[26] George A. Miller (1995): *WordNet: A Lexical Database for English*. Communinications of the ACM 38(11), pp. 39–41. Available at http://doi.acm.org/10.1145/219717.219748.

[27] Jeff Mitchell & Mirella Lapata (2010): *Composition in Distributional Models of Semantics*. Cognitive Science 34(8), pp. 1388–1429, doi:10.1111/j.1551-6709.2010.01106.x. Available at https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1551-6709.2010.01106.x.

[28] Saif Mohammad, Ekaterina Shutova & Peter Turney (2016): *Metaphor as a medium for emotion: An empirical study*. In: Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, pp. 23–33, doi:10.18653/v1/S16-2003.

[29] Roberto Navigli (2009): *Word sense disambiguation: A survey*. ACM computing surveys (CSUR) 41(2), pp. 1–69. Available at https://doi.org/10.1145/1459352.1459355.

[30] Jeffrey Pennington, Richard Socher & Christopher D Manning (2014): *Glove: Global vectors for word representation*. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, doi:10.3115/v1/D14-1162.

[31] Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke & Mehrnoosh Sadrzadeh (2015): *Open System Categorical Quantum Semantics in Natural Language Processing*. In Lawrence S. Moss & Pawel Soboci'nski, editors: *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24-26, 2015, Nijmegen, The Netherlands*, *LIPIcs* 35, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 270–289. Available at https://doi.org/10.4230/LIPIcs.CALCO.2015.270.

[32] Anne Preller & Mehrnoosh Sadrzadeh (2011): *Bell States and Negative Sentences in the Distributed Model of Meaning*. *Electronic Notes in Theoretical Computer Science* 270(2), pp. 141 – 153. Available at https://doi.org/10.1016/j.entcs.2011.01.028. Proceedings of the 6th International Workshop on Quantum Physics and Logic (QPL 2009).

[33] Nils Reimers & Iryna Gurevych (2019): *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, doi:10.18653/v1/D19-1410.

[34] Benjamin Rodatz, Razin Shaikh & Lia Yeh (2021): *Conversational Negation Using Worldly Context in Compositional Distributional Semantics*. In: *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*, Association for Computational Linguistics, Groningen, The Netherlands, pp. 53–65. Available at https://doi.org/10.48550/arXiv.2105.05748.

[35] Mehrnoosh Sadrzadeh, Dimitri Kartsaklis & Esma Balkır (2018): *Sentence Entailment in Compositional Distributional Semantics*. *Annals of Mathematics and Artificial Intelligence* 82(4), pp. 189–218. Available at https://doi.org/10.1007/s10472-017-9570-x.

[36] Hinrich Schütze (1998): *Automatic Word Sense Discrimination*. *Comput. Linguist.* 24(1), pp. 97–123. Available at https://dl.acm.org/doi/10.5555/972719.972724.

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

[38] Razin A. Shaikh, Lia Yeh, Benjamin Rodatz & Bob Coecke (2022): *Composing Conversational Negation*. *Electronic Proceedings in Theoretical Computer Science* 372, p. 352–367. Available at http://dx.doi.org/10.4204/EPTCS.372.25.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser & Illia Polosukhin (2017): *Attention is all you need*. *Advances in neural information processing systems* 30. Available at https://doi.org/10.48550/arXiv.1706.03762.

[40] John van de Wetering (2017): *Ordering information on distributions*. arXiv:1701.06924.

A Numbers of Sentence Pairs

		Cosine Similarity	Von Neumann Entropy
GloVe	verb-only	331	
	short-form	269	
	long-form	63	
Word2Vec	verb-only	331	
	short-form	269	
	long-form	63	
BERT2DM models	verb-only	300	336
	short-form	204	252
	long-form	8	10
Context2DM	verb-only	310	338
	short-form	212	256
	long-form	8	10
Word2DM models	verb-only	332	342
	short-form	335	342
	long-form	172	220

Table 6: Number of sentence pairs used in each test after removing out-of-vocabulary words. SBERT and Infersent covered all words.