

EPTCS 394

Proceedings of the
**19th International Conference on
Quantum Physics and Logic**

Wolfson College, Oxford, UK, 27 June - 1 July 2022

Edited by: Stefano Gogioso and Matty Hoban

Published: 16th November 2023
DOI: 10.4204/EPTCS.394
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Table of Contents	i
Preface	iii
<i>Stefano Gogioso and Matty Hoban</i>	
Logical Characterization of Contextual Hidden-Variable Theories based on Quantum Set Theory ...	1
<i>Masanao Ozawa</i>	
Generators and Relations for 2-Qubit Clifford+T Operators	13
<i>Xiaoning Bian and Peter Selinger</i>	
Reducing 2-QuBit Gate Count for ZX-Calculus based Quantum Circuit Optimization	29
<i>Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann and Wolfgang Gehrke</i>	
Building Qutrit Diagonal Gates from Phase Gadgets	46
<i>John van de Wetering and Lia Yeh</i>	
Complete Flow-Preserving Rewrite Rules for MBQC Patterns with Pauli Measurements	66
<i>Tommy McElvanney and Miriam Backens</i>	
Quantum Linear Optics via String Diagrams	83
<i>Giovanni de Felice and Bob Coecke</i>	
Identification of Causal Influences in Quantum Processes	101
<i>Isaac Friend and Aleks Kissinger</i>	
Architecture-Aware Synthesis of Phase Polynomials for NISQ Devices	116
<i>Arianne Meijer-van de Griend and Ross Duncan</i>	
Encoding High-level Quantum Programs as SZX-diagrams	141
<i>Augustin Borgna and Rafael Romero</i>	
Q# as a Quantum Algorithmic Language	170
<i>Kartik Singhal, Kesha Hietala, Sarah Marshall and Robert Rand</i>	
Universal Properties of Partial Quantum Maps	192
<i>Pablo Andrés-Martínez, Chris Heunen and Robin Kaarsgaard</i>	
The Causal Structure of Semantic Ambiguities	208
<i>Daphne Wang and Mehrnoosh Sadrzadeh</i>	
Tunable Quantum Neural Networks in the QPAC-Learning Framework	221
<i>Viet Pham Ngoc, David Tuckey and Herbert Wiklicky</i>	

How to Sum and Exponentiate Hamiltonians in ZXW Calculus	236
<i>Razin A. Shaikh, Quanlong Wang and Richie Yeung</i>	
Diagrammatic Analysis for Parameterized Quantum Circuits	262
<i>Tobias Stollenwerk and Stuart Hadfield</i>	
A Biset-Enriched Categorical Model for Proto-Quipper with Dynamic Lifting	302
<i>Peng Fu, Kohei Kishida, Neil J. Ross and Peter Selinger</i>	
Symbolic Synthesis of Clifford Circuits and Beyond	343
<i>Matthew Amy, Owen Bennett-Gibbs and Neil J. Ross</i>	
Dynamic Qubit Routing with CNOT Circuit Synthesis for Quantum Compilation	363
<i>Arianne Meijer-van de Griend and Sarah Meng Li</i>	
Quantum de Finetti Theorems as Categorical Limits, and Limits of State Spaces of C*-algebras . . .	400
<i>Sam Staton and Ned Summers</i>	
Annealing Optimisation of Mixed ZX Phase Circuits	415
<i>Stefano Gogioso and Richie Yeung</i>	
Finite-dimensional Quantum Observables are the Special Symmetric Dagger-Frobenius Algebras of CP Maps	432
<i>Stefano Gogioso</i>	

Preface

This volume contains the proceedings of the 19th International Conference on Quantum Physics and Logic (QPL 2022). The conference was held from 27 June to 1 July 2022 at the University of Oxford, UK.

Quantum Physics and Logic is an annual conference that brings together academic and industry researchers working on mathematical foundations of quantum computation, quantum physics, and related areas. The main focus is on the use of algebraic and categorical structures, formal languages, semantic methods, as well as other mathematical and computer scientific 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 2022 conference solicited four different kinds of submissions: proceedings submissions, non-proceedings submissions, poster submissions, and programming tool submissions.

Proceedings submissions were papers that were 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 2022. No other kinds 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-proceeding submissions were allowed 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 one of the poster sessions of the conference. Programming tool submissions consisted of three page descriptions of programming tools or frameworks. Authors of accepted programming tool submissions were given an opportunity to present their software during a dedicated "Software Session".

These proceedings contain 20 contributed papers that were selected for publication by the QPL 2022 Program Committee. An additional contributed paper is included, that was selected for publication by the QPL 2020 Program Committee, but not included in the QPL 2020 proceedings in error. Papers submitted to QPL undergo a review process that is managed by members of the PC. Almost all submissions received at least three reviews. The selection of accepted papers was done through the use of the EasyChair conference management system following consideration of the submitted reviews and following (where necessary) discussion among the PC. The review process was single-blind: the identity of the authors is revealed to the reviewers, but not vice-versa. PC members were allowed to invite external experts to serve as sub-reviewers and to participate in the discussion of those submissions which they were invited to review.

A record 132 submissions (excluding withdrawals and retractions) were considered for review by the PC. QPL 2023 had 49 accepted submissions in the non-proceedings track and 25 accepted submissions in the

proceedings track. Most of the talks were presented during parallel sessions, but a selection of talks were presented during plenary sessions in the mornings. The program had a poster session, a session dedicated to showcasing accepted programming tool submissions, as well as an industry session, where industrial sponsors of QPL 2022 were given an opportunity to present their companies. The industry session consisted of seven talks, from four sponsors: Quantinuum, Quandela, Huawei and Hashberg.

The official website of the conference is <https://www.qplconference.org/> and it contains a lot of relevant information about QPL 2022, as well as links to previous editions.

The Program Committee consisted of 40 members who were: Antonio Acin, Miriam Backens, Jonathan Barrett, Dan Browne, Caslav Brukner, Giulio Chiribella, Bob Coecke, Alejandro Diaz-Caro, Ross Duncan, Yuan Feng, Stefano Gogioso (co-chair), Amar Hadzihasanovic, Teiko Heinosaari, Chris Heunen, Matty Hoban (co-chair), Martti Karvonen, Kohei Kishida, Aleks Kissinger, Ravi Kunjwal, Martha Lewis, Shane Mansfield, Konstantinos Meichanetzidis, David Moore, Mio Murao, Simon Perdrix, Julien Ross, Mehrnoosh Sadrzadeh, Ana Belén Sainz, Carlo Maria Scandolo, John Selby, Rui Soares Barbosa, Pawel Sobocinski, Rob Spekkens, Isar Stubbe, Benoit Valiron, John van de Wetering, Quanlong Wang, Alexander Wilce, Vladimir Zamdzhiev, and Margherita Zorzi.

The Organising Committee consisted of 6 members who were: Bob Coecke, Aleks Kissinger, Stefano Gogioso, Konstantinos Meichanetzidis, Matty Hoban, and Destiny Chen.

The QPL Steering Committee consisted of Bob Coecke, Prakash Panangaden, and Peter Selinger.

We wish to thank all the members of the PC for their work in selecting the program of QPL 2022. We thank all external sub-reviewers for their help, and the authors for their submissions. 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, the student helpers who volunteered to assist us, as well as the staff of Wolfson College, Oxford, who helped us with the organisation of the conference. Finally, we thank the QPL steering committee for their support and we thank all people who have contributed to the success of QPL 2022.

QPL 2022 received (financial) support from Quantinuum, Quandela, the US Air Force, MindSpore (Huawei), and Hashberg.

Nov 2023, Stefano Gogioso and Matty Hoban

Logical Characterization of Contextual Hidden-Variable Theories Based on Quantum Set Theory

Masanao Ozawa*

Center for Mathematical Science and Artificial Intelligence, Academy of Emerging Sciences, Chubu University,
1200 Matsumoto-cho, Kasugai 487-8501, Japan

ozawa@isc.chubu.ac.jp

Graduate School of Informatics, Nagoya University, Chikusa-ku, Nagoya 464-8601, Japan

ozawa@is.nagoya-u.ac.jp

While non-contextual hidden-variable theories are proved to be impossible, contextual ones are possible. In a contextual hidden-variable theory, an observable is called a beable if the hidden-variable assigns its value in a given measurement context specified by a state and a preferred observable. Halvorson and Clifton characterized the algebraic structure of beables as a von Neumann subalgebra, called a beable subalgebra, of the full observable algebra such that the probability distribution of every observable affiliated therewith admits the ignorance interpretation. On the other hand, we have shown that for every von Neumann algebra there is a unique set theoretical universe such that the internal “real numbers” bijectively correspond to the observables affiliated with the given von Neumann algebra. Here, we show that a set theoretical universe is associated with a beable subalgebra if and only if it is ZFC-satisfiable, namely, every theorem of ZFC set theory holds with probability equal to unity. Moreover, we show that there is a unique maximal ZFC-satisfiable subuniverse “implicitly definable”, in the sense of Malament and others, by the given measurement context. The set theoretical language for the ZFC-satisfiable universe, characterized by the present work, rigorously reconstructs Bohr’s notion of the “classical language” to describe the beables in a given measurement context.

1 Introduction

In 1935, Einstein, Podolsky, and Rosen [14] argued that the quantum mechanical description of physical reality is incomplete. Bohr [6] immediately responded to rebut their conclusion. As to which claim is correct, the majority view has become in favor of Bohr, based on no-go theorems against non-contextual hidden-variable theories by von Neumann [25], Gleason [16], Kochen-Specker [22]. From the above debate it has been concluded that non-contextual hidden-variable theories for quantum mechanics are impossible. Nevertheless, contextual hidden-variable theories are possible as Bohr’s complementarity interpretation [5] and the Bohmian mechanics [4].

In accordance with the above view, the modal interpretation of quantum mechanics has been studied extensively, used for no-collapse interpretation to solve the measurement problem [9], and successfully articulated Bohr’s otherwise obscure complementarity interpretation of quantum mechanics [9,18,33]. In modal interpretation, an observable possessing a well defined value is called a “beable”, and it is attempted to define the class of beables as broad as possible depending on the given measurement context. The observable to be measured is naturally considered to possess its value to be revealed by the measurement even in a superposition of its eigenstates. This avoids the measurement problem arising from state

*Supported by JSPS KAKENHI Grant Numbers JP22K03424, JP21K11764, JP19H04066.

collapsing with the Dirac-von Neumann interpretation [12,25] that an observable has a value only in its eigenstates or mixtures of them [9].

Halvorson and Clifton [17] algebraically characterized a contextual hidden-variable theory for a state ψ in the Hilbert space \mathcal{H} of a quantum system \mathbf{S} as a von Neumann subalgebra \mathcal{B} , called a beable subalgebra, of the full observable algebra $\mathcal{L}(\mathcal{H})$ such that the probability distribution of every observable therein admits the ignorance interpretation. Their celebrated uniqueness theorem determines the unique maximal beable subalgebra $\mathcal{B}(\psi, A)$ “implicitly definable”, in the sense of Malament [23] and others, by the given measurement context (ψ, A) . However, the algebraic structure of beables does not directly treat the logical structure of observational propositions nor the structure of the language speaking of beables, so that we do not have a formal framework to treat, for instance, Bohr’s original notion of the “classical language” to describe beables in a given measurement context, or Hardy’s logical formulation of non-locality.¹

Here, we introduce a new approach based on quantum set theory to provide a logical framework for modal interpretations. Quantum set theory was introduced by Takeuti [37] for constructing mathematics based on quantum logic and developed by the present author [26–32]; the relationship with topos quantum theory was studied by Eva [15] and Döring *et al.* [13]. In the preceding study, we have shown that for any von Neumann subalgebra \mathcal{M} of the full observable algebra $\mathcal{L}(\mathcal{H})$ on a Hilbert space \mathcal{H} , we can construct the unique mathematical, or more specifically, set theoretical, universe $V[\mathcal{M}]$ based on the logic represented by the projection lattice $\mathcal{P}(\mathcal{M})$ in \mathcal{M} such that the internal “real numbers” in the universe $V[\mathcal{M}]$ coincides with the self-adjoint operators (or observables) affiliated with the von Neumann algebra \mathcal{M} .

In this work, we shall logically characterize a contextual hidden-variable theory by showing that a von Neumann algebra \mathcal{B} is a beable subalgebra of $\mathcal{L}(\mathcal{H})$ for a state $\psi \in \mathcal{H}$ if and only if the set theoretical universe $V[\mathcal{B}]$ based on the logic $\mathcal{P}(\mathcal{B})$ is ZFC-satisfiable in ψ , in the sense that every theorem of ZFC set theory in the language $\mathbf{L}(\in, V[\mathcal{B}])$ of set theory augmented by the names of elements of $V[\mathcal{B}]$ holds in $V[\mathcal{B}]$ with probability equal to unity in the state ψ . In this case, the set of beables represented by self-adjoint operators affiliated with \mathcal{B} coincides with the set of the internal “real numbers” in the universe $V[\mathcal{B}]$. Moreover, we uniquely determine the maximal ZFC-satisfiable subuniverse $V[\mathcal{B}(\psi, A)]$ among those implicitly definable by the given measurement context (ψ, A) . Thus, we can identify Bohr’s notion of “classical language” describing the beables in the given measurement context (ψ, A) with the language $\mathbf{L}(\in, V[\mathcal{B}(\psi, A)])$.²

2 Algebraic approach to beables

In this paper, we consider a quantum system \mathbf{S} described by a separable Hilbert space \mathcal{H} , called the *state space* of \mathbf{S} , with inner product (ξ, η) for all $\xi, \eta \in \mathcal{H}$, linear in η and conjugate linear in ξ . Observables of \mathbf{S} are bijectively represented by self-adjoint operators (densely defined) on \mathcal{H} and every unit vector in \mathcal{H} represents a (pure) state of \mathbf{S} . If an observable (represented by a self-adjoint operator) X is measured in a state (represented by a unit vector) $\phi \in \mathcal{H}$, the outcome \mathbf{x} of the measurement satisfies the Born statistical formula

$$\Pr\{\mathbf{x} \leq x \mid \phi\} = (\phi, E^X(x)\phi) \quad (1)$$

¹We focus on the former topic in this paper and we will discuss the latter elsewhere.

²Bohr called the notion of beables in the given measurement context in several different ways, e.g., as the observables definable for the given measurement arrangement, or the elements of physical reality determined by the measurement of a preferred observable in the given state.

for every real number $x \in \mathbb{R}$, where $E^X(x)$ is the resolution of the identity belonging to the self-adjoint operator X [25, p. 119]. We denote by μ_ϕ^X the Borel probability measure on \mathbb{R} uniquely determined by the relation $\mu_\phi^X((-\infty, x]) = (\phi, E^X(x)\phi)$, and call it the (*Born*) *probability distribution* of the observable X in the state ϕ . From the above, for any (real-valued) Borel function f the observable $f(X)$ defined by $f(X) = \int_{\mathbb{R}} f(x)dE^X(x)$ has the expectation value

$$\text{Ex}\{f(X)|\phi\} = (\phi, f(X)\phi) \quad (2)$$

if $\phi \in \text{dom}(f(X))$.

Now we consider a situation, called the *measurement context* (ψ, A) , in which an observable A is to be measured in a state ψ , and we take the ignorance interpretation for the Born probability distribution μ_ψ^A , as a typical reading of Bohr's complementarity interpretation [9,17,18,33], that just before the measurement of the observable A in the state ψ , the observable A possesses its value with the probability distribution μ_ψ^A , and that the measurement faithfully reveals the value possessed by A . We would call an observable that is considered to possess its value in the measurement context (ψ, A) as a *beable* [2, p. 41] in that measurement context. Obviously, the observable A itself should be a beable together with its functions $f(A)$ for all Borel functions f .

The objective of modal interpretations is to determine the set of beables in the context (ψ, A) as broad as possible. We would call it the *maximal beable set*. From the impossibility theorem of non-contextual hidden-variable theories by von Neumann [25] and others,³ the maximal beable set cannot be the whole set of observables.

It is natural to assume that the maximal beable set is closed under appropriate algebraic operations (i.e., addition, Jordan product, and Lie product) and closed under appropriate convergences, so that we assume that the maximal beable set is the set of observables (or self-adjoint operators) affiliated with a von Neumann algebra \mathcal{B} on \mathcal{H} .⁴

The first requirement for such a von Neumann algebra \mathcal{B} concerns the state ψ requires that the Born probability distribution of every beable admit the ignorance interpretation. This requirement is mathematically formulated as follows. We call any normalized positive linear functional on \mathcal{B} a *state* on \mathcal{B} . A state ω on \mathcal{B} is said to be *dispersion-free* iff $\omega(X^*X) = |\omega(X)|^2$ for any $X \in \mathcal{B}$. We say that a von Neumann algebra \mathcal{B} is a *beable subalgebra for a state vector* $\psi \in \mathcal{H}$ iff there is a probability measure μ on the space $\mathcal{D}(\mathcal{B})$ of dispersion-free states of \mathcal{B} satisfying

$$(\psi, X\psi) = \int_{\mathcal{D}(\mathcal{B})} \omega(X)d\mu(\omega) \quad (3)$$

for every $X \in \mathcal{B}$. The second requirement is, of course, that the observable A be affiliated with \mathcal{B} as a "privileged" observable, from which it follows that $f(A)$ be affiliated with \mathcal{B} for all Borel functions f by the Borel function calculus in von Neumann algebras. Thus, we call a von Neumann algebra \mathcal{B} on \mathcal{H} a *beable subalgebra for the measurement context* (ψ, A) iff it satisfies following conditions:

- (i) (Beable) \mathcal{B} is a beable subalgebra for ψ .
- (ii) (A-Priv) A is affiliated with \mathcal{B} .

³The first general proof of the impossibility theorem for non-contextual hidden-variable theories was given by von Neumann [25]; see [1,10,11,24] for the recent debate on the status of von Neumann's impossibility proof. Later, Kochen and Specker [22] proved the theorem for the Hilbert space with the dimension greater than 2 under the sole requirement that hidden-variables satisfy functional relations for observables; a similar result can be derived as a corollary of Gleason's theorem [16].

⁴An observable X is affiliated with \mathcal{B} if and only if $E^X(x) \in \mathcal{B}$ for all $x \in \mathbb{R}$.

According to the above formulation, we are tempted to identify the maximal beable set with the set of observables affiliated with a maximal von Neumann algebra among those satisfying the requirements (A-Priv) and (Beable) above. However, such a choice is not unique.

To see this, consider the measurement context (ψ, A) for the composite system \mathbf{S} of two spin 1/2 particles with the state space $\mathcal{H} = \mathbf{C}^2 \otimes \mathbf{C}^2$, consisting of the singlet state $\psi = 2^{-1/2}(|+_z\rangle|-_z\rangle - |-_z\rangle|+_z\rangle)$ and the z -component $A = \sigma_z \otimes I$ of Pauli spin operators of the first particle. In this case, we have many beable subalgebras $\mathcal{B}_\theta = W^*(\sigma_z) \otimes W^*(\sigma_\theta)$, where $\sigma_\theta = \cos \theta \sigma_z + \sin \theta \sigma_x$, for $0 \leq \theta < \pi$.⁵ Yet, there is no common maximal subalgebra \mathcal{B}_{\max} , since if \mathcal{B}_{\max} were to include \mathcal{B}_θ and $\mathcal{B}_{\theta'}$ with $\theta \neq \theta'$, there would be no dispersion-free state on \mathcal{B}_{\max} .

To consider which beable subalgebra we should choose, recall the debate between EPR [14] and Bohr [6] around the “reality criterion” proposed by EPR.

If, without in any way disturbing a system, we can predict with certainty (i.e., with probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity. [14, p.777]

EPR argued, in accordance with this criterion, that in the EPR state the position and momentum of the second particle have simultaneous reality, because the measurement on the first particle measures the second particle without disturbing it [36, p. 334]. Yet, the position and momentum cannot have simultaneous reality in any states by the uncertainty principle, so that EPR concluded that quantum-mechanical description of physical reality is incomplete. Bohr immediately responded to EPR. Bohr claimed that although the measurement on the first particle does not mechanically disturb the second particle, the measurement on the first particle influences the condition that defines elements of reality for the second particle, and he rejected EPR’s conclusion.

[T]here is in a case like that just considered no question of a mechanical disturbance of the system under investigation during the last critical stage of the measuring procedure. But even at this stage there is essentially the question of *an influence on the very conditions which define the possible types of predictions regarding the future behavior of the system*. ... [W]e see that the argumentation of the mentioned authors [EPR] does not justify their conclusion that quantum-mechanical description is essentially incomplete.” [6, p. 700]

Following the reality criterion posed by EPR [14], but “contextualized” to the particular measurement context (ψ, A) as suggested by Bohr [6] above, we should choose $B_0 = I \otimes \sigma_z$ to be a beable in this measurement context but not $B_\theta = I \otimes \sigma_\theta$ with $\theta \neq 0$, because only the value of B_0 can be inferred from the value of $A = \sigma_z \otimes I$ in the measurement context (ψ, A) without disturbing the second particle. If the observer were to measure the observable $A' = \sigma_x \otimes I$ instead of $A = \sigma_z \otimes I$, then from the value of A' the observer could infer the value of $B_{\pi/2} = I \otimes \sigma_x$ without disturbing the second particle. EPR might have concluded that both B_0 and $B_{\pi/2}$ are beables, or elements of reality. However, Bohr [6] pointed out that the status of being beable depends on the inference from the value of A to the value of B_0 or the inference from the value of A' to the value of $B_{\pi/2}$, but each inference is justified only in the respective context, in which classical logic and classical mathematics can be used in the ordinary sense, and that there is no context-free classical language that supports the above two types of inferences simultaneously. Thus, what are beables of the second particle depends on what is measured on the first particle as Bohr [6] suggested.

Halvorson and Clifton [18, pp.14–15] proposed a mathematical approach to single out appropriate beable subalgebras consistent with the above “contextualized” reality criterion, as follows. We say that

⁵For an observable X , we denote by $W^*(X)$ the von Neumann algebra generated by X if X is bounded, or the von Neumann algebra generated by $E^X(x)$ for all $x \in \mathbb{R}$.

a von Neumann algebra \mathcal{B} is *implicitly definable* by a measurement context (ψ, A) iff $U^* \mathcal{B} U = \mathcal{B}$ for any unitary $U \in \mathcal{L}(\mathcal{H})$ such that $U^* A U = A$ and $U \psi = \psi$. In other words, \mathcal{B} is implicitly definable by (ψ, A) iff the membership relation $X \in \mathcal{B}$ for any $X \in \mathcal{L}(\mathcal{H})$ is not affected (i.e., $X \in \mathcal{B}$ if and only if $\alpha(X) \in \mathcal{B}$) by any automorphism α of $\mathcal{L}(\mathcal{H})$ that leaves ψ and A invariant,⁶ which is widely used in foundational studies, for example, by Malament [23, p.297]. A beable subalgebra \mathcal{B} for (ψ, A) is called *definable* iff it further satisfies

(iii) (Def) \mathcal{B} is implicitly definable by (ψ, A) .

We call a von Neumann algebra \mathcal{B} a *maximally definable beable subalgebra for the measurement context* (ψ, A) iff \mathcal{B} satisfies (A-Priv), (Beable), and (Def) and \mathcal{B} is maximal (in the set inclusion) among all von Neumann subalgebras of $\mathcal{L}(\mathcal{H})$ satisfying those three requirements. Then, the celebrated Halvorson-Clifton uniqueness theorem [17] is stated as follows.

Theorem 2.1 (Halvorson-Clifton [17]). *For any state ψ and an observable A , there exists the unique maximally definable beable subalgebra $\mathcal{B}(\psi, A)$ for the measurement context (ψ, A) , and it is of the form*

$$\mathcal{B}(\psi, A) = W^*(A)P \oplus \mathcal{L}(P^\perp \mathcal{H}), \quad (4)$$

where P is the projection from \mathcal{H} onto the cyclic subspace $\mathcal{C}(\psi, A)$ of \mathcal{H} generated by ψ and A , i.e., $\mathcal{C}(\psi, A) = \{f(A)\psi \mid f \text{ is a bounded Borel function}\}^{\perp\perp}$.

3 Quantum logic

Bohr's view has, unfortunately, prevailed with several improper restatements, and there have been only a few serious attempts to reconstruct his reply in a rigorous analysis. In his early contribution, Howard [19,20] attempted to clarify what is the element of physical reality for Bohr. He focused on Bohr's notion of "classical description". In fact, Bohr emphasizes in several places that one should describe experimental evidence classically.⁷

Howard's view was further sharpened by Halvorson and Clifton [18] in the framework of modal interpretations as presented in the preceding section; see also Ref. [33] for mathematical refinements. While Howard [19,20] formulated Bohr's classicality requirement by the notion of "appropriate" mixture, Halvorson and Clifton reformulate it as the requirement that the Born probability distribution admit the ignorance interpretation. However, Bohr never explained his notion of "classical description" by the ignorance interpretation of the Born probability distribution. In this paper, we attempt to go a step further.

In view of Bohr's writings, it seems that "classical" means classical physics, but he also stated:

[I]t would seem that the recourse to three-valued logic, sometimes proposed as means for dealing with the paradoxical features of quantum theory, is not suited to give a clearer account of the situation, since all well-defined experimental evidence, even if it cannot be analysed in terms of classical physics, must be expressed in ordinary language making use of common logic. [7, p. 317]

⁶Any automorphism α of $\mathcal{L}(\mathcal{H})$ is of the form $\alpha(X) = U^* X U$ for some unitary $U \in \mathcal{L}(\mathcal{H})$ [34, p. 119].

⁷"[I]t is decisive to recognize that, *however far the phenomena transcend the scope of classical physical explanation, the account of all evidence must be expressed in classical terms.* The argument is simply that by the word "experiment" we refer to a situation where we can tell others what we have done and what we have learned and that, therefore, the account of the experimental arrangement and of the results of the observations must be expressed in unambiguous language with suitable application of the terminology of classical physics." [8, p.209]

According to the above, we can see that what Bohr refers to in the word “classical” is a broader concept than classical physics and it is well understood as the classical language that obeys classical logic rather than quantum logic. Therefore, if the algebraic reconstruction should be consistent with Bohr’s original view, the language that speaks of the beables should obey classical logic and theorems of classical mathematics. The purpose of this paper is to rigorously realize this interpretation of Bohr’s view in the framework of quantum set theory.

Recall that the logic of observational propositions on the system \mathbf{S} is represented by the projection lattice $\mathcal{P}(\mathcal{H})$ of all projections on \mathcal{H} . In order to treat a context-dependent part of the whole observational propositions, we consider a sublogics of $\mathcal{P}(\mathcal{H})$ represented by the projection lattice $\mathcal{P}(\mathcal{M})$ of a von Neumann subalgebra \mathcal{M} of $\mathcal{L}(\mathcal{H})$. Let \mathcal{M} be a von Neumann algebra on \mathcal{H} and denote by $\mathcal{O}(\mathcal{M})$ the set of observables affiliated with \mathcal{M} . The observational propositions on the observables affiliated with \mathcal{M} are constructed from *atomic propositions* “ $X \leq_o \lambda$ ”, where $X \in \mathcal{O}(\mathcal{M})$ and $\lambda \in \mathbb{R}$, by connecting them with \wedge (conjunction), \vee (disjunction), \rightarrow (conditional), \leftrightarrow (equivalence), and \neg (negation). In what follows, we consider only the conjunction and negation as primitive symbols and the other connectives as derived symbols by the following definitions: $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2)$, $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$. Here, we note that the conditional \rightarrow is defined as the Sasaki conditional [35]. We denote by $\mathbf{L}(\mathcal{M})$ the set of observational propositions on the observables in $\mathcal{O}(\mathcal{M})$. We also write $\mathbf{L}(\mathcal{H}) = \mathbf{L}(\mathcal{L}(\mathcal{H}))$. We have $\mathbf{L}(\mathcal{M}_1) \subseteq \mathbf{L}(\mathcal{M}_2)$ if $\mathcal{M}_1 \subseteq \mathcal{M}_2$.

Following Birkhoff and von Neumann [3], every observational proposition φ has the projection-valued truth value $[[\varphi]]_o$ determined by the following rules.

- (i) $[[X \leq_o \lambda]]_o = E^X(\lambda)$.
- (ii) $[[\varphi_1 \wedge \varphi_2]]_o = [[\varphi_1]]_o \wedge [[\varphi_2]]_o$.
- (iii) $[[\neg\varphi]]_o = [[\varphi]]_o^\perp$.

Then for every observational proposition φ we define the probability that the observational proposition φ holds in the state ψ by $\Pr\{\varphi\|\psi\} = \|[[\varphi]]_o\psi\|^2$. It is well-know that the logic of observational propositions in $\mathbf{L}(\mathcal{H})$ is non-distributive, so that it does not necessarily follow the laws of classical logic. According to Bohr’s view on “classical description” it is natural to expect that in the state ψ the language $\mathbf{L}(\mathcal{B})$ satisfies classical logic and all the mathematical theorems, if \mathcal{B} is beable for the state ψ . However, the language $\mathbf{L}(\mathcal{M})$ has only a limited power in expressing observational relations between observables, just as the propositional logic has only a limited power in expressing relations between mathematical objects. In fact, the language $\mathbf{L}(\mathcal{M})$ cannot generally express relations between observables in $\mathcal{O}(\mathcal{M})$ such as equality and order relation, so that it cannot assign the projection valued truth-value, or the probability in the state ψ , for the proposition “A and B have the same value”. In what follows we strengthen the language of observational propositions to have full power of expressing all the mathematically definable relations between observables with projection-valued truth values.

4 Quantum set theory

In this and the next sections, we shall introduce basic facts about quantum set theory. We refer the reader to Ref. [31] for detailed formulations. Let \mathcal{M} be a von Neumann algebra. The purpose of quantum set theory is to extend the universe, or a ground model, V of ZFC set theory to a sort of “generic extension” $V[\mathcal{M}]$ adding self-adjoint operators affiliated with \mathcal{M} as “generic reals” to V .

We denote by V the universe of the Zermelo-Fraenkel set theory with the axiom of choice (ZFC). Let $\mathbf{L}(\in)$ be the language for first-order theory with equality augmented by a binary relation symbol \in , bounded quantifier symbols $\forall x \in y$, $\exists x \in y$, and no constant symbols. For any class U , the language

$\mathbf{L}(\in, U)$ is the one obtained by adding a name for each element of U . We take the symbols $\neg, \wedge, \forall x \in y$, and $\forall x$ as primitive, and the symbols $\vee, \rightarrow, \leftrightarrow, \exists x \in y$, and $\exists x$ as derived symbols by defining:

- (i) $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$,
- (ii) $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2)$,
- (iii) $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$,
- (iv) $\exists x \in y \varphi(x) = \neg(\forall x \in y \neg\varphi(x))$,
- (v) $\exists x \varphi(x) = \neg(\forall x \neg\varphi(x))$.

Let \mathcal{M} be a von Neumann algebra on a Hilbert space \mathcal{H} . For each ordinal α , let

$$V_\alpha[\mathcal{M}] = \{u \mid u : \text{dom}(u) \rightarrow \mathcal{P}(\mathcal{M}) \text{ and } (\exists \beta < \alpha) \text{dom}(u) \subseteq V_\beta[\mathcal{M}]\}.^8 \quad (5)$$

The $\mathcal{P}(\mathcal{M})$ -valued universe $V[\mathcal{M}]$ is defined by

$$V[\mathcal{M}] = \bigcup_{\alpha \in \text{On}} V_\alpha[\mathcal{M}], \quad (6)$$

where On is the class of all ordinals. We shall write $V_\alpha[\mathcal{H}] = V_\alpha[\mathcal{L}(\mathcal{H})]$ and $V[\mathcal{H}] = V[\mathcal{L}(\mathcal{H})]$. For every $u \in V[\mathcal{M}]$, the *rank* of u , denoted by $\text{rank}(u)$, is defined as the least α such that $u \in V_{\alpha+1}[\mathcal{M}]$. It is easy to see that if $u \in \text{dom}(v)$ then $\text{rank}(u) < \text{rank}(v)$.

We introduce the implication operation \rightarrow and its dual conjunction operation $*$ on the lattice $\mathcal{P}(\mathcal{M})$ by $P \rightarrow Q = P^\perp \vee (P \wedge Q)$ and $P * Q = P \wedge (P^\perp \vee Q)$, or equivalently $P * Q = (P \rightarrow Q^\perp)^\perp$, for any $P, Q \in \mathcal{P}(\mathcal{M})$. The operation \rightarrow on $\mathcal{P}(\mathcal{M})$ is called the Sasaki arrow and the operation $Q \mapsto P * Q$ is called the Sasaki projection [21].

For any $u, v \in V[\mathcal{M}]$, the $\mathcal{P}(\mathcal{M})$ -valued truth values of atomic formulas $u = v$ and $u \in v$ are assigned by the following rules recursive in rank.

- (vi) $\llbracket u = v \rrbracket_{\mathcal{M}} = \inf_{u' \in \text{dom}(u)} (u(u') \rightarrow \llbracket u' \in v \rrbracket_{\mathcal{M}}) \wedge \inf_{v' \in \text{dom}(v)} (v(v') \rightarrow \llbracket v' \in u \rrbracket_{\mathcal{M}})$.
- (vii) $\llbracket u \in v \rrbracket_{\mathcal{M}} = \sup_{v' \in \text{dom}(v)} (v(v') * \llbracket u = v' \rrbracket_{\mathcal{M}})$.

To each statement φ of $\mathbf{L}(\in, V[\mathcal{M}])$ we assign the $\mathcal{P}(\mathcal{M})$ -valued truth value $\llbracket \varphi \rrbracket_{\mathcal{M}}$ by the following rules.

- (viii) $\llbracket \neg\varphi \rrbracket_{\mathcal{M}} = \llbracket \varphi \rrbracket_{\mathcal{M}}^\perp$.
- (ix) $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\mathcal{M}} = \llbracket \varphi_1 \rrbracket_{\mathcal{M}} \wedge \llbracket \varphi_2 \rrbracket_{\mathcal{M}}$.
- (x) $\llbracket (\forall x \in u) \varphi(x) \rrbracket_{\mathcal{M}} = \bigwedge_{u' \in \text{dom}(u)} (u(u') \rightarrow \llbracket \varphi(u') \rrbracket_{\mathcal{M}})$.
- (xi) $\llbracket (\forall x) \varphi(x) \rrbracket_{\mathcal{M}} = \bigwedge_{u \in V[\mathcal{M}]} \llbracket \varphi(u) \rrbracket_{\mathcal{M}}$.

We say that a statement φ of $\mathbf{L}(\in, V[\mathcal{M}])$ holds in $V[\mathcal{M}]$ if $\llbracket \varphi \rrbracket_{\mathcal{M}} = I$. A formula in $\mathbf{L}(\in, V[\mathcal{M}])$ is called a Δ_0 -formula iff it has no unbounded quantifiers $\forall x$ or $\exists x$. The following theorem holds [31, Theorem 4.3].

Theorem 4.1 (Δ_0 -Absoluteness Principle). *For any Δ_0 -formula $\varphi(x_1, \dots, x_n)$ of $\mathbf{L}(\in)$ and $u_1, \dots, u_n \in V[\mathcal{M}]$, we have*

$$\llbracket \varphi(u_1, \dots, u_n) \rrbracket_{\mathcal{M}} = \llbracket \varphi(u_1, \dots, u_n) \rrbracket_{\mathcal{L}(\mathcal{H})}. \quad (7)$$

⁸We denote by $\text{dom}(f)$ the domain of a function f . By $f : D \rightarrow R$ we mean that f is a function defined on a set D with values in a set R .

Henceforth, for any Δ_0 -formula $\varphi(x_1, \dots, x_n)$ and $u_1, \dots, u_n \in V[\mathcal{M}]$, we abbreviate $\llbracket \varphi(u_1, \dots, u_n) \rrbracket = \llbracket \varphi(u_1, \dots, u_n) \rrbracket_{\mathcal{M}}$, which is the common $\mathcal{P}(\mathcal{H})$ -valued truth value for $u_1, \dots, u_n \in V[\mathcal{H}]$.

The universe V can be embedded in $V[\mathcal{M}]$ by the following operation $\check{v} : v \mapsto \check{v}$ defined by the \in -recursion: for each $v \in V$, $\check{v} = \{\check{u} \mid u \in v\} \times \{I\}$. Note that $\check{v} \in V[\mathcal{M}]$ for any $v \in V$ and any von Neumann subalgebra $\mathcal{M} \subseteq \mathcal{L}(\mathcal{H})$. Then we have the following [31, Theorem 4.8].

Theorem 4.2 (Δ_0 -Elementary Equivalence Principle). *For any Δ_0 -formula $\varphi(x_1, \dots, x_n)$ of $\mathbf{L}(\in)$ and $u_1, \dots, u_n \in V$, we have $\langle V, \in \rangle \models \varphi(u_1, \dots, u_n)$ if and only if $\llbracket \varphi(\check{u}_1, \dots, \check{u}_n) \rrbracket = I$.*

Thus, $V[\mathcal{M}]$ includes (a copy of) the standard universe V as a Δ_0 -elementary equivalent submodel. For further detail about the universe $V[\mathcal{M}]$ we refer the reader to [31].

5 Transfer principle

For any $u \in V[\mathcal{M}]$, we define the *support* of u , denoted by $L(u)$, by transfinite recursion on the rank of u by the relation

$$L(u) = \bigcup_{x \in \text{dom}(u)} L(x) \cup \{u(x) \mid x \in \text{dom}(u)\} \cup \{0\}. \quad (8)$$

For $\mathcal{A} \subseteq V[\mathcal{M}]$ we write $L(\mathcal{A}) = \bigcup_{u \in \mathcal{A}} L(u)$ and for $u_1, \dots, u_n \in V[\mathcal{M}]$ we write $L(u_1, \dots, u_n) = L(\{u_1, \dots, u_n\})$. Let $\mathcal{A} \subseteq \mathcal{L}(\mathcal{H})$. The *commutant* of \mathcal{A} in $\mathcal{L}(\mathcal{H})$, denoted by \mathcal{A}' , is defined by

$$\mathcal{A}' = \{A \in \mathcal{L}(\mathcal{H}) \mid [A, B] = 0 \text{ for all } B \in \mathcal{A}\}, \quad (9)$$

and the *commutant* of \mathcal{A} in $\mathcal{P}(\mathcal{H})$, denoted by $\mathcal{A}^!$, is defined by $\mathcal{A}^! = \mathcal{A}' \cap \mathcal{P}(\mathcal{H})$.

Let $\mathcal{A} \subseteq \mathcal{P}(\mathcal{H})$. Takeuti [37] introduced the *commutator* of \mathcal{A} , denoted by $\text{com}(\mathcal{A})$, given by

$$\text{com}(\mathcal{A}) = \bigvee \{E \in \mathcal{A}' \cap \mathcal{A}^! \mid [P, Q]E = 0 \text{ for all } P, Q \in \mathcal{A}\}. \quad (10)$$

For any $P_1, \dots, P_n \in \mathcal{P}(\mathcal{H})$, we write $\text{com}(\{P_1, \dots, P_n\}) = \text{com}(P_1, \dots, P_n)$. We refer the reader to [28] for further properties of commutators.

Let $\mathcal{A} \subseteq V[\mathcal{M}]$. The *commutator* of \mathcal{A} , denoted by $\text{com}(\mathcal{A})$, is defined by

$$\text{com}(\mathcal{A}) = \text{com}(L(\mathcal{A})). \quad (11)$$

For any $u_1, \dots, u_n \in V[\mathcal{M}]$, we write $\text{com}(u_1, \dots, u_n) = \text{com}(\{u_1, \dots, u_n\})$.

We have the following transfer principle for bounded theorems of ZFC [31, Theorem 4.15].

Theorem 5.1 (Δ_0 -ZFC Transfer Principle). *For any Δ_0 -formula $\varphi(x_1, \dots, x_n)$ of $\mathbf{L}(\in)$ and $u_1, \dots, u_n \in V[\mathcal{M}]$, if $\varphi(x_1, \dots, x_n)$ is provable in ZFC, then*

$$\llbracket \varphi(u_1, \dots, u_n) \rrbracket \geq \text{com}(u_1, \dots, u_n). \quad (12)$$

6 Internal real numbers in quantum set theory

Let \mathbb{Q} be the set of rational numbers in V . We define the set of rational numbers in the model $V[\mathcal{M}]$ to be $\check{\mathbb{Q}}$. We define a real number in the model by a Dedekind cut of the rational numbers. More precisely,

we identify a real number with the upper segment of a Dedekind cut whose lower segment has no end point. Therefore, the formal definition of the predicate $\mathbb{R}(x)$, “ x is a real number,” is expressed by

$$\begin{aligned} \mathbb{R}(x) \quad := \quad & \forall y \in x (y \in \check{\mathbb{Q}}) \wedge \exists y \in \check{\mathbb{Q}} (y \in x) \wedge \exists y \in \check{\mathbb{Q}} (y \notin x) \\ & \wedge \forall y \in \check{\mathbb{Q}} (y \in x \leftrightarrow \forall z \in \check{\mathbb{Q}} (y < z \rightarrow z \in x)). \end{aligned} \quad (13)$$

We define $\mathbb{R}[\mathcal{M}]$ to be the interpretation of the set \mathbb{R} of real numbers in $V[\mathcal{M}]$ as follows.

$$\mathbb{R}[\mathcal{M}] = \{u \in V[\mathcal{M}] \mid \text{dom}(u) = \text{dom}(\check{\mathbb{Q}}) \text{ and } \llbracket \mathbb{R}(u) \rrbracket = I\}. \quad (14)$$

For any $u \in \mathbb{R}[\mathcal{M}]$ and $\lambda \in \mathbb{R}$, we define $E^u(\lambda)$ by

$$E^u(\lambda) = \bigwedge_{\lambda < r \in \mathbb{Q}} u(\check{r}). \quad (15)$$

Then it can be shown that $\{E^u(\lambda)\}_{\lambda \in \mathbb{R}}$ is a resolution of identity in $\mathcal{P}(\mathcal{M})$ and hence by the spectral theorem there is an observable $\hat{u} \in \mathcal{O}(\mathcal{M})$ uniquely satisfying $\hat{u} = \int_{\mathbb{R}} \lambda dE^u(\lambda)$. On the other hand, let $A \in \mathcal{O}(\mathcal{M})$. We define $\tilde{A} \in V[\mathcal{M}]$ by

$$\tilde{A} = \{(\check{r}, E^A(r)) \mid r \in \mathbb{Q}\}. \quad (16)$$

Then $\text{dom}(\tilde{A}) = \text{dom}(\check{\mathbb{Q}})$ and $\tilde{A}(\check{r}) = E^A(r)$ for all $r \in \mathbb{Q}$. It is easy to see that $\tilde{A} \in \mathbb{R}[\mathcal{M}]$ and we have $(\hat{u})^\sim = u$ for all $u \in \mathbb{R}[\mathcal{M}]$ and $(\tilde{A})^\wedge = A$ for all $A \in \mathcal{O}(\mathcal{M})$. Therefore, the correspondence between $\mathbb{R}[\mathcal{M}]$ and $\mathcal{O}(\mathcal{M})$ is bijective. We call the above correspondence the *Takeuti correspondence*. Now, we have the following.

Theorem 6.1. *The relations*

$$(i) \quad E^{\tilde{A}}(\lambda) = \bigwedge_{\lambda < r \in \mathbb{Q}} u(\check{r}) \quad \text{for all } \lambda \in \mathbb{Q},$$

$$(ii) \quad u(\check{r}) = E^{\tilde{A}}(r) \quad \text{for all } r \in \mathbb{Q},$$

where $u = \tilde{A} \in \mathbb{R}[\mathcal{M}]$ and $A = \hat{u} \in \mathcal{O}(\mathcal{M})$, sets up a bijective correspondence between $u \in \mathbb{R}[\mathcal{M}]$ and $A \in \mathcal{O}(\mathcal{M})$.

For any $r \in \mathbb{R}$, we shall write $\tilde{r} = (r1)^\sim$, where $r1$ is the scalar operator on \mathcal{H} . Then we have $\text{dom}(\tilde{r}) = \text{dom}(\check{\mathbb{Q}})$ and $\tilde{r}(\check{r}) = \llbracket \check{r} \leq \check{r} \rrbracket$, so that we have $L(\tilde{r}) = \{0, 1\}$. The order relation for $u, v \in \mathbb{R}[\mathcal{M}]$ is naturally defined by

$$u \leq v := (\forall x \in \check{\mathbb{Q}}) [x \in v \rightarrow x \in u]. \quad (17)$$

Recall that a formula $\varphi(x_1, \dots, x_n) \in \mathbf{L}(\in, V[\mathcal{M}])$ is called a Δ_0 -formula iff it contains no unbounded quantifiers $\forall x$ nor $\exists x$. In this paper, we focus on the sublanguage $\mathbf{L}_0(\in, V[\mathcal{M}])$ consisting of Δ_0 -formulas in $\mathbf{L}(\in, V[\mathcal{M}])$. Then, for every statement $\varphi \in \mathbf{L}_0(\in, V[\mathcal{M}])$ we have the $\mathcal{P}(\mathcal{M})$ -valued truth value $\llbracket \varphi \rrbracket = \llbracket \varphi \rrbracket_{\mathcal{P}(\mathcal{M})}$, and for every observable $X \in \mathcal{O}(\mathcal{M})$ we have a real number $\tilde{X} \in \mathbb{R}[\mathcal{M}]$ in $V[\mathcal{M}]$. Thus, there is an embedding of every observational proposition φ in the language $\mathbf{L}(\mathcal{M})$ into a statement $\tilde{\varphi}$ in the language $\mathbf{L}_0(\in, V[\mathcal{M}])$ defined by the following rules for any $X \in \mathcal{O}(\mathcal{M})$ and $x \in \mathbb{R}$, and observational propositions $\varphi, \varphi_1, \varphi_2$:

$$(Q1) \quad \widetilde{X \leq_o x} := \tilde{X} \leq \tilde{x}.$$

$$(Q2) \quad \widetilde{\neg \varphi} := \neg \tilde{\varphi}.$$

$$(Q3) \quad \widetilde{\varphi_1 \wedge \varphi_2} := \widetilde{\varphi_1} \wedge \widetilde{\varphi_2}.$$

Then, it is easy to see that the relation

$$[[\widetilde{\varphi}]] = [[\varphi]]_o \quad (18)$$

holds for any observational proposition φ .

Thus, all the observational propositions are embedded in a set of statements in $\mathbf{L}_0(\in, V[\mathcal{M}])$ preserving their projection-valued truth values. Quantum set theory provides a language strong enough to express all the possible mathematical relations among observables. For every mathematical relation $R(x_1, \dots, x_n)$ definable in ZFC set theory, we can assign the quantum logical truth value $[[R(\tilde{X}_1, \dots, \tilde{X}_n)]] \in \mathcal{P}(\mathcal{M})$ of the relation $R(\tilde{X}_1, \dots, \tilde{X}_n)$ for any observables $X_1, \dots, X_n \in \mathcal{O}(\mathcal{H})$. For example, we can generally define the projection valued truth values for the order relation and the equality relation for observables so that the following relation hold:

$$[[\tilde{X} \leq \tilde{Y}]] = [[(\forall x \in \tilde{\mathcal{Q}}) [\tilde{Y} \leq x \rightarrow \tilde{X} \leq x]]], \quad (19)$$

$$[[\tilde{X} = \tilde{Y}]] = [[\tilde{X} \leq \tilde{Y} \wedge \tilde{Y} \leq \tilde{X}]]. \quad (20)$$

7 Beable subuniverses

Let $\psi \in \mathcal{H}$ be a state. We say that a statement $\varphi(u_1, \dots, u_n) \in \mathbf{L}_0(\in, V[\mathcal{M}])$ holds in ψ , and write $\psi \Vdash \varphi(u_1, \dots, u_n)$, iff $\Pr\{\varphi(u_1, \dots, u_n) \mid \psi\} = I$. For any von Neumann algebra $\mathcal{M} \subseteq \mathcal{L}(\mathcal{H})$, the subuniverse $V[\mathcal{M}] \subseteq V[\mathcal{H}]$ is said to be *ZFC-satisfiable in ψ* iff

$$\psi \Vdash \phi(u_1, \dots, u_n) \quad (21)$$

for any $u_1, \dots, u_n \in V[\mathcal{M}]$ and any formula $\varphi(x_1, \dots, x_n) \in \mathbf{L}_0(\in)$ provable in ZFC.

The following theorem holds.

Theorem 7.1. *Let \mathcal{M} be a von Neumann algebra on \mathcal{H} . Then the following conditions are all equivalent.*

- (i) $V[\mathcal{M}]$ is ZFC-satisfiable in ψ .
- (ii) \mathcal{M} is beable for ψ .
- (iii) $[X, Y]\psi = 0$ for any $X, Y \in \mathcal{M}$.
- (iv) $|\psi\rangle\langle\psi| \leq \text{com}(u_1, \dots, u_n)$ for any $u_1, \dots, u_n \in V[\mathcal{M}]$.

8 Context-definable beable subuniverses

Let (ψ, A) be a measurement context. For any unitary operator U on \mathcal{H} , we define $\alpha_U : V[\mathcal{H}] \rightarrow V[\mathcal{H}]$ by transfinite recursion on the rank of $u \in V[\mathcal{H}]$ as

$$\alpha_U(u) = \{\langle \alpha_U(x), U^*u(x)U \rangle \mid x \in \text{dom}(u)\}. \quad (22)$$

A subclass $\mathcal{U} \subseteq V[\mathcal{M}]$ is called *definable* by (ψ, A) iff

- (i) $\tilde{A} \in \mathcal{U}$,
- (ii) $\alpha_U(\mathcal{U}) \subseteq \mathcal{U}$ for any unitary operator U on \mathcal{H} satisfying $[A, U] = 0$ and $U\psi = \psi$.

A subuniverse $V[\mathcal{M}] \subseteq V[\mathcal{H}]$ is called a *maximally definable ZFC-satisfiable subuniverse* for the measurement context (ψ, A) iff $V[\mathcal{M}]$ is a ZFC-satisfiable subuniverse in ψ definable by (ψ, A) and there is no ZFC-satisfiable subuniverse $V[\mathcal{M}_0]$ definable by (ψ, A) that properly includes $V[\mathcal{M}]$. Then, we have the following.

Theorem 8.1. (i) A subuniverse $V[\mathcal{M}]$ is a ZFC-satisfiable subuniverse of $V[\mathcal{H}]$ definable by (ψ, A) if and only if \mathcal{M} is a beable subalgebra of $\mathcal{L}(\mathcal{H})$ definable by (ψ, A) .

(ii) There uniquely exists a maximally definable ZFC-satisfiable subuniverse $V[\mathcal{M}] \subseteq V[\mathcal{H}]$ for any measurement context (ψ, A) . In this case, \mathcal{M} is of the form $\mathcal{M} = \mathcal{B}(\psi, A)$, i.e.,

$$\mathcal{M} = W^*(A)P \oplus \mathcal{L}(P^\perp \mathcal{H}), \quad (23)$$

where P is the projection from \mathcal{H} onto the cyclic subspace $\mathcal{C}(\psi, A)$ of \mathcal{H} generated by ψ and A .

References

- [1] P. Acuña (2021): *Von Neumann's Theorem Revisited*. *Found. Phys.* 51, pp. 73/1–73/29, doi:10.1007/s10701-021-00474-5.
- [2] J. S. Bell (2004): *Subject and object*. In: *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*, 2nd edition, Cambridge UP, pp. 40–44, doi:10.1017/CBO9780511815676.007.
- [3] G. Birkhoff & J. von Neumann (1936): *The Logic of Quantum Mechanics*. *Ann. Math.* 37, pp. 823–843, doi:10.2307/1968621.
- [4] D. Bohm (1952): *A Suggested Interpretation of the Quantum Theory in Terms of "Hidden Variables," I, II*. *Phys. Rev.* 85, pp. 166–179, 180–193, doi:10.1103/PhysRev.85.166, 10.1103/PhysRev.85.180.
- [5] N. Bohr (1928): *The Quantum Postulate and the Recent Development of Atomic Theory*. *Nature* 121, pp. 580–590, doi:10.1038/121580a0.
- [6] N. Bohr (1935): *Can Quantum-Mechanical Description of Physical Reality be Considered Complete?* *Phys. Rev.* 48, pp. 696–702, doi:10.1103/PhysRev.48.696.
- [7] N. Bohr (1948): *On the Notions of Causality and Complementarity*. *Dialectica* 2, pp. 312–319, doi:10.1111/j.1746-8361.1948.tb00703.x.
- [8] N. Bohr (1949): *Discussion with Einstein on epistemological problems in atomic physics*. In P. A. Shilpp, editor: *Albert Einstein: Philosopher-Scientist, The Library of Living Philosophers VII*, Northwestern University, Evanston, pp. 200–241, doi:10.1016/S1876-0503(08)70379-7.
- [9] J. Bub (1997): *Interpreting the Quantum World*. Cambridge UP, Cambridge.
- [10] J. Bub (2010): *Von Neumann's 'No Hidden Variables' Proof: A Re-Appraisal*. *Found. Phys.* 40, pp. 1333–1340, doi:10.1007/s10701-010-9480-9.
- [11] D. Dieks (2017): *Von Neumann's impossibility proof: Mathematics in the service of rhetorics*. *Stud. Hist. Philos. Sci. B* 60, pp. 136–148, doi:10.1016/j.shpsb.2017.01.008.
- [12] P. A. M. Dirac (1958): *The Principles of Quantum Mechanics*, 4th edition. Oxford UP, Oxford, doi:10.1063/1.3062610.
- [13] A. Döring, B. Eva & M. Ozawa (2021): *A Bridge Between Q-Worlds*. *Rev. Symb. Log.* 14(2), pp. 447–486, doi:10.1017/S1755020319000492.
- [14] A. Einstein, B. Podolsky & N. Rosen (1935): *Can Quantum-Mechanical Description of Physical Reality be Considered Complete?* *Phys. Rev.* 47, pp. 777–780, doi:10.1103/PhysRev.47.777.
- [15] B. Eva (2015): *Towards a Paraconsistent Quantum Set Theory*. *Electronic Proceedings in Theoretical Computer Science* 195, pp. 158–169, doi:10.4204/EPTCS.195.12.
- [16] A. M. Gleason (1957): *Measures on the Closed Subspaces of a Hilbert Space*. *J. Math. Mech.* 6, pp. 885–893, doi:10.1512/iumj.1957.6.56050.
- [17] H. Halvorson & R. Clifton (1999): *Maximal Beable Subalgebras of Quantum Mechanical Observables*. *Int. J. Theor. Phys.* 38, pp. 2441–2484, doi:10.1023/A:1026628407645.

- [18] H. Halvorson & R. Clifton (2002): *Reconsidering Bohr's reply to EPR*. In T. Placek & J. Butterfield, editors: *Non-locality and Modality*, Kluwer, Dordrecht, pp. 3–18, doi:10.1007/978-94-010-0385-8_1.
- [19] D. Howard (1994): *What makes a classical concept classical?* In Jan Faye & Henry J. Folse, editors: *Niels Bohr and Contemporary Philosophy*, Kluwer, Dordrecht, pp. 201–229, doi:10.1007/978-94-015-8106-6_9.
- [20] D. A. Howard (1979): *Complementarity and Ontology: Niels Bohr and the Problem of Scientific Realism in Quantum Physics*. Ph.D. thesis, Boston University.
- [21] G. Kalmbach (1983): *Orthomodular Lattices*. Academic, London.
- [22] S. Kochen & E. P. Specker (1967): *The Problem of Hidden Variables in Quantum Mechanics*. *J. Math. Mech.* 17, pp. 59–87, doi:10.1512/iumj.1968.17.17004.
- [23] D. Malament (1977): *Causal Theories of Time and the Conventionality of Simultaneity*. *Noûs* 11, pp. 293–300, doi:10.2307/2214766.
- [24] N. D. Mermin & R. Schack (2018): *Homer Nodded: Von Neumann's Surprising Oversight*. *Found. Phys.* 48, pp. 1007–1020, doi:10.1007/s10701-018-0197-5.
- [25] J. von Neumann (1955): *Mathematical Foundations of Quantum Mechanics*. Princeton UP, Princeton, NJ. [Originally published: *Mathematische Grundlagen der Quantenmechanik* (Springer, Berlin, 1932)].
- [26] M. Ozawa (2007): *Transfer Principle in Quantum Set Theory*. *J. Symb. Log.* 72, pp. 625–648, doi:10.2178/jsl/1185803627. arXiv:math/0604349.
- [27] M. Ozawa (2014): *Quantum Set Theory Extending the Standard Probabilistic Interpretation of Quantum Theory (Extended Abstract)*. *Electronic Proceedings in Theoretical Computer Science (EPTCS)* 172, pp. 15–26, doi:10.4204/EPTCS.172.2.
- [28] M. Ozawa (2016): *Quantum Set Theory Extending the Standard Probabilistic Interpretation of Quantum Theory*. *New Generat. Comput.* 34, pp. 125–152, doi:10.1007/s00354-016-0205-2. arXiv:1504.06838.
- [29] M. Ozawa (2017): *Operational Meanings of Orders of Observables Defined through Quantum Set Theories with Different Conditionals*. *Electronic Proceedings in Theoretical Computer Science (EPTCS)* 236, pp. 127–144, doi:10.4204/EPTCS.236.9.
- [30] M. Ozawa (2017): *Orthomodular-Valued Models for Quantum Set Theory*. *Rev. Symb. Log.* 10, pp. 782–807, doi:10.1017/S1755020317000120. arXiv:0908.0367.
- [31] M. Ozawa (2021): *Quantum set theory: Transfer Principle and De Morgan's Laws*. *Ann. Pure Appl. Log.* 172, pp. 102938/1–102938/42, doi:10.1016/j.apal.2020.102938. arXiv:2002.06692.
- [32] M. Ozawa (2021): *Reforming Takeuti's Quantum Set Theory to Satisfy de Morgan's Laws*. In Arai, T. et al., editor: *Advances in Mathematical Logic*, Springer Singapore, Singapore, pp. 143–159, doi:10.1007/978-981-16-4173-2_7. arXiv:2012.02928.
- [33] M. Ozawa & Y. Kitajima (2012): *Reconstructing Bohr's Reply to EPR in Algebraic Quantum Theory*. *Found. Phys.* 42, pp. 475–487, doi:10.1007/s10701-011-9615-7. arXiv:1107.0737.
- [34] S. Sakai (1971): *C*-Algebras and W*-Algebras*. Springer, Berlin.
- [35] U. Sasaki (1954): *Orthocomplemented Lattices Satisfying the Exchange Axiom*. *J. Sci. Hiroshima Univ. A* 17, pp. 293–302, doi:10.32917/hmj/1557281141.
- [36] E. Schrödinger (1935): *Die gegenwärtige Situation in der Quantenmechanik*. *Naturwissenschaften* 23, pp. 807–812, 823–828, 844–849, doi:10.1007/BF01491891, 10.1007/BF01491914, 10.1007/BF01491987. [English translation by J. D. Trimmer, *Proc. Am. Philos. Soc.* 124, 323–338 (1980)].
- [37] G. Takeuti (1981): *Quantum Set Theory*. In E. Beltrametti & B. C. van Frassen, editors: *Current Issues in Quantum Logic*, Plenum, New York, pp. 303–322, doi:10.1007/978-1-4613-3228-2_19.

Generators and Relations for 2-Qubit Clifford+ T Operators

Xiaoning Bian and Peter Selinger

Dalhousie University

We give a presentation by generators and relations of the group of Clifford+ T operators on two qubits. The proof relies on an application of the Reidemeister-Schreier theorem to an earlier result of Greylyn, and has been formally verified in the proof assistant Agda.

1 Introduction

The simplification of Clifford+ T circuits is a topic of current interest in quantum computing [4, 5, 6, 15, 16, 17]. The Clifford+ T gate set is both universal [18] and convenient for quantum error correction [9], and is therefore the preferred gate set for fault-tolerant quantum computing. Generally, in a fault-tolerant regime, applying a Clifford gate is some orders of magnitude cheaper than applying a T -gate, and therefore, it is sensible to try to simplify circuits so as to minimize the T -count [3]. Many methods for doing so have been proposed in the recent literature, including methods based on matroid partitioning [2], Reed-Muller codes [4], and ZX calculus [5, 6, 16]. Regardless of which method is used, the objective is to replace a Clifford+ T circuit by a simpler, but equivalent circuit. This requires being able to tell when two circuits are equivalent. Surprisingly, no complete set of relations for ancilla-free Clifford+ T circuits is currently known, i.e., there is no known set of relations by which any two equivalent Clifford+ T circuits can be transformed into each other.

In this paper, we give such a complete set of relations for the case of 2-qubit Clifford+ T circuits. We do this in several steps. First, a presentation of the group $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$ of all unitary 4×4 -matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ is known due to the work of Greylyn [13]. Second, it is known that the group of 2-qubit Clifford+ T circuits is exactly the subgroup of this group consisting of matrices whose determinant is in $\{\pm 1, \pm i\}$ [10]. Third, there is a theorem in group theory called the Reidemeister-Schreier theorem, by which a complete set of relations for a subgroup can be derived from a complete set of relations for the supergroup. Fourth, since the resulting relations are very long and complicated, we simplify them.

The last two steps of this procedure (applying the Reidemeister-Schreier theorem and simplifying the resulting relations) require a large amount of algebraic manipulations. Our longest equational proof has 480 steps, each of which in turns requires a lemma or rewrite procedure whose proof itself requires many equational steps. Such proofs would be impossible to verify by hand, and even verifying them by software is error-prone since it is hard to guarantee that no unwarranted assumptions were used. For this reason, we encoded our proof in machine-checkable form, using the proof assistant Agda [1].

The rest of this paper is organized as follows. In Section 2, we state our main result. Section 3 gives a brief overview of the proof. In Section 4, we present the required background material, including Greylyn's presentation of $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$, the Reidemeister-Schreier theorem, and the Pauli rotation representation, which is an important tool for manipulating Clifford+ T circuits. We also briefly describe our reasons for formalizing our proof in a proof assistant. Section 5 describes our formal proof of the main result. In Section 6, we briefly discuss the meaning of the Clifford+ T relations, and especially of the three "non-obvious" relations. Section 7 contains some concluding remarks and ideas for future work.

2 Statement of the main result

Recall that the set of Clifford operators is generated by the operators

$$\omega = e^{i\pi/4}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

and is closed under multiplication and tensor product. Every such operator U is of size $2^n \times 2^n$ for some natural number n , and as usual, we say that U is an operator on n qubits. We write $\mathcal{C}(n)$ for the group of n -qubit Clifford operators. It is well-known that this group is finite for any given n [21], and therefore not universal for quantum computing. We obtain a universal gate set by also adding the T -gate as a generator.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix},$$

The resulting operators are called the Clifford+ T operators, and we write $\mathcal{C}\mathcal{T}(n)$ for the n -qubit Clifford+ T group.

In this paper, we focus on the case $n = 2$. Our goal is to give a complete presentation of the 2-qubit Clifford+ T group in terms of generators and relations. To ensure that all of our generators are 4×4 -matrices, we introduce the following notation: we write $T_0 = T \otimes I$ and $T_1 = I \otimes T$, and similarly for H_0 , H_1 , S_0 , and S_1 . We also identify the scalar ω with the 4×4 -matrix ωI . Our main result is the following:

Theorem 2.1. *The 2-qubit Clifford+ T group is presented by (\mathcal{X}, Γ) , where the set of generators is*

$$\mathcal{X} = \{\omega, H_0, H_1, S_0, S_1, T_0, T_1, CZ\},$$

and the set of relations Γ is shown in Figure 1.

In Figure 1, we have used circuit notation to express some of the relations; for example, we have written

$$\begin{array}{c} \boxed{T} \\ \hline \end{array}, \quad \begin{array}{c} \hline \boxed{T} \\ \hline \end{array}, \quad \text{and} \quad \begin{array}{c} \bullet \\ | \\ \bullet \end{array}$$

for T_0 , T_1 , and CZ , respectively. Note that the qubits are numbered from top to bottom. We write circuits in the same order as matrix multiplication. Moreover, in relations (C18)–(C20), we have used a number of abbreviations; these are defined in Figure 2. The empty word is denoted ε .

3 Proof outline

In a nutshell, the proof can be described in a few sentences. It proceeds as follows. Let $R = \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ be the smallest subring of the complex numbers containing $\frac{1}{\sqrt{2}}$ and i , and let $G = U_4(R)$ be the group of unitary 4×4 -matrices with entries in R . Then it is clear that $\mathcal{C}\mathcal{T}(2)$ is a subgroup of G , because all of its generators belong to G . Moreover, from [10], it is known that $\mathcal{C}\mathcal{T}(2)$ is precisely equal to the subgroup of G consisting of matrices whose determinant is a power of i . A presentation of G by generators and relations was given by Greylyn [13]. There is a general procedure, called the Reidemeister-Schreier procedure [19, 20], for finding generators and relations of a subgroup, given generators and relations of the supergroup. Applying this procedure therefore yields a complete set of relations for $\mathcal{C}\mathcal{T}(2)$.

While in principle, the above proof outline suffices to prove Theorem 2.1, in practice there is a large amount of non-trivial work involved in generating and simplifying the actual relations. For this reason, we have formalized Theorem 2.1 and its proof in the proof assistant Agda. This allows the proof to be independently checked without too much manual work.

(a) Monoidal relations:

$$\omega A = A\omega, \quad \text{where } A \in \{H_i, S_i, T_i, CZ\} \quad (C1)$$

$$A_0 B_1 = B_1 A_0, \quad \text{where } A, B \in \{H, S, T\} \quad (C2)$$

(b) Order of Clifford group elements:

$$\omega^8 = \varepsilon \quad (C3)$$

$$H_i^2 = \varepsilon \quad (C4)$$

$$S_i^4 = \varepsilon \quad (C5)$$

$$(S_i H_i)^3 = \omega \quad (C6)$$

$$CZ^2 = \varepsilon \quad (C7)$$

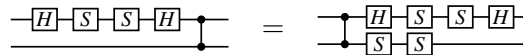
(c) Remaining Clifford relations:



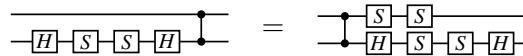
$$\quad (C8)$$



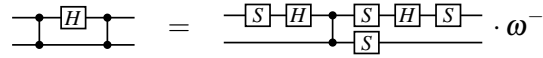
$$\quad (C9)$$



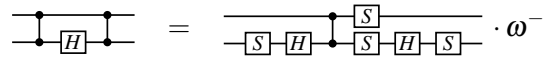
$$\quad (C10)$$



$$\quad (C11)$$



$$\quad (C12)$$



$$\quad (C13)$$

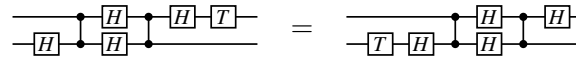
(d) “Obvious” relations involving T :

$$T_i^2 = S_i \quad (C14)$$

$$(T_i H_i S_i S_i H_i)^2 = \omega \quad (C15)$$

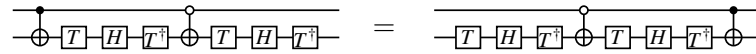


$$\quad (C16)$$

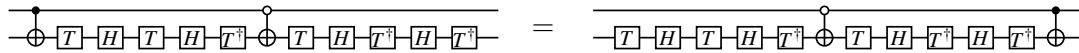


$$\quad (C17)$$

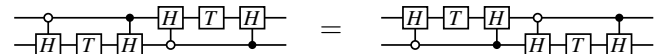
(e) “Non-obvious” relations involving T :



$$\quad (C18)$$



$$\quad (C19)$$



$$\quad (C20)$$

Figure 1: Relations for 2-qubit Clifford+ T operators. Here $i \in \{0, 1\}$.

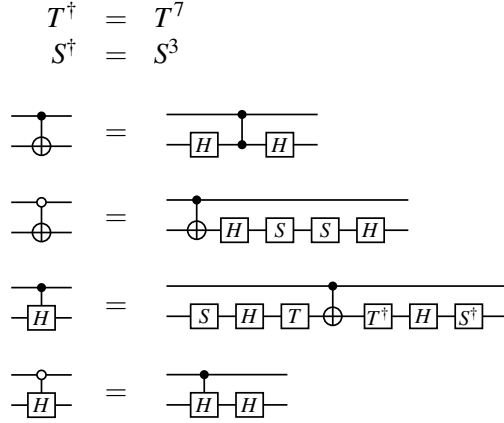


Figure 2: Abbreviations used in circuit notations

4 Background

4.1 Presentation of $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$

As usual, \mathbb{Z} is the ring of integers. Let $R = \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ be the smallest subring of the complex numbers containing $\frac{1}{\sqrt{2}}$ and i . Let $\omega = e^{i\pi/4}$ be an 8th root of unity, and note that $\omega = \frac{1+i}{\sqrt{2}} \in R$. As before, $U_4(R)$ is the group of unitary 4×4 -matrices with entries in R .

Greylyn [13] gave a presentation of $U_4(R)$ by generators and relations. His generators are $\omega_{[j]}$, $X_{[j,k]}$, and $H_{[j,k]}$, where $j, k \in \{0, \dots, 3\}$ and $j < k$. The relations are shown in Figure 3. The intended interpretation of the generators is as 1- and 2-level matrices; specifically, $\omega_{[j]}$ is like the identity matrix, except with ω in the j th row and column, and $X_{[j,k]}$ and $H_{[j,k]}$ are like identity matrices, except with the entries of X , respectively H , in the j th and k th rows and columns, like this:

$$\omega_{[j]} = \begin{matrix} & \dots & j & \dots \\ \vdots & \begin{bmatrix} I & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & I \end{bmatrix} & \\ j & & \\ \vdots & & \end{matrix}, \quad X_{[j,k]} = \begin{matrix} & \dots & j & \dots & k & \dots \\ \vdots & \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} & \\ j & & \\ \vdots & & \\ k & & \\ \vdots & & \end{matrix}, \quad H_{[j,k]} = \begin{matrix} & \dots & j & \dots & k & \dots \\ \vdots & \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} & \\ j & & \\ \vdots & & \\ k & & \\ \vdots & & \end{matrix}.$$

Note that we index rows and columns of matrices starting from 0, whereas Greylyn indexed them starting from 1. Greylyn's result is the following:

Theorem 4.1 (Greylyn [13]). *A presentation of the group $U_4(R)$ is given by (\mathcal{Y}, Δ) , where the set of generators is $\mathcal{Y} = \{\omega_{[j]}, X_{[j,k]}, H_{[j,k]} \mid j, k \in \{1, \dots, 4\} \text{ and } j < k\}$, and the set of relations Δ is shown in Figure 3.*

4.2 The Reidemeister-Schreier theorem for monoids

The Reidemeister-Schreier theorem is a theorem in group theory that allows one to derive a complete set of relations for a subgroup from a complete set of relations for the supergroup, given enough information about the cosets. We will use a version of the Reidemeister-Schreier theorem that works for monoids,

(a) Order of generators:

$$\omega_{[j]}^8 = \varepsilon \quad (\text{G1})$$

$$H_{[j,k]}^2 = \varepsilon \quad (\text{G2})$$

$$X_{[j,k]}^2 = \varepsilon \quad (\text{G3})$$

(b) Disjoint generators commute:

$$\omega_{[j]} \omega_{[k]} = \omega_{[k]} \omega_{[j]}, \quad \text{where } j \neq k \quad (\text{G4})$$

$$\omega_{[\ell]} H_{[j,k]} = H_{[j,k]} \omega_{[\ell]}, \quad \text{where } \ell \neq j, k \quad (\text{G5})$$

$$\omega_{[\ell]} X_{[j,k]} = X_{[j,k]} \omega_{[\ell]}, \quad \text{where } \ell \neq j, k \quad (\text{G6})$$

$$H_{[j,k]} H_{[\ell,t]} = H_{[\ell,t]} H_{[j,k]}, \quad \text{where } \{\ell, t\} \cap \{j, k\} = \emptyset \quad (\text{G7})$$

$$H_{[j,k]} X_{[\ell,t]} = X_{[\ell,t]} H_{[j,k]}, \quad \text{where } \{\ell, t\} \cap \{j, k\} = \emptyset \quad (\text{G8})$$

$$X_{[j,k]} X_{[\ell,t]} = X_{[\ell,t]} X_{[j,k]}, \quad \text{where } \{\ell, t\} \cap \{j, k\} = \emptyset \quad (\text{G9})$$

(c) X permutes indices:

$$X_{[j,k]} \omega_{[k]} = \omega_{[j]} X_{[j,k]} \quad (\text{G10})$$

$$X_{[j,k]} \omega_{[j]} = \omega_{[k]} X_{[j,k]} \quad (\text{G11})$$

$$X_{[j,k]} X_{[j,\ell]} = X_{[k,\ell]} X_{[j,k]} \quad (\text{G12})$$

$$X_{[j,k]} X_{[\ell,j]} = X_{[\ell,k]} X_{[j,k]} \quad (\text{G13})$$

$$X_{[j,k]} H_{[j,\ell]} = H_{[k,\ell]} X_{[j,k]} \quad (\text{G14})$$

$$X_{[j,k]} H_{[\ell,j]} = H_{[\ell,k]} X_{[j,k]} \quad (\text{G15})$$

(d) $\omega_{[j]} \omega_{[k]}$ is diagonal:

$$\omega_{[j]} \omega_{[k]} X_{[j,k]} = X_{[j,k]} \omega_{[j]} \omega_{[k]} \quad (\text{G16})$$

$$\omega_{[j]} \omega_{[k]} H_{[j,k]} = H_{[j,k]} \omega_{[j]} \omega_{[k]} \quad (\text{G17})$$

(e) Relations for H :

$$H_{[j,k]} X_{[j,k]} = \omega_{[k]}^4 H_{[j,k]} \quad (\text{G18})$$

$$H_{[j,k]} \omega_{[j]}^2 H_{[j,k]} = \omega_{[j]}^6 H_{[j,k]} \omega_{[j]}^3 \omega_{[k]}^5 \quad (\text{G19})$$

$$H_{[j,k]} H_{[\ell,t]} H_{[j,\ell]} H_{[k,t]} = H_{[j,\ell]} H_{[k,t]} H_{[j,k]} H_{[\ell,t]}, \quad \text{where } k < \ell \quad (\text{G20})$$

Figure 3: Greylyn's relations for $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$. Whenever we use a generator $X_{[j,k]}$ or $H_{[j,k]}$, we implicitly assume that $j < k$.

which we now describe. To our knowledge, this monoid formulation of the Reidemeister-Schreier theorem does not appear in the literature.

If X is a set, let us write X^* for the set of finite sequences of elements of X , which we also call *words* over the alphabet X . We write $w \cdot v$ or simply wv for the concatenation of words, making X^* into a monoid. The unit of this monoid is the empty word ε . As usual, we identify X with the set of one-letter words.

Let G be a monoid and let $X \subseteq G$ be a subset of G . We write $\langle X \rangle$ for the smallest submonoid of G containing X , and we say that X *generates* G if $\langle X \rangle = G$. Given any word $w \in X^*$, we write $[w]_G \in G$ for the canonical interpretation of w in G , i.e., $[-]_G : X^* \rightarrow G$ is the unique monoid homomorphism such that $[x]_G = x$ for all $x \in X$.

A *relation* over X is an element of $X^* \times X^*$, i.e., an ordered pair of words. We say that a relation (w, v) is *valid* in G if $[w]_G = [v]_G$. If Γ is a set of relations over X , we write \sim_Γ for the smallest congruence relation on X^* containing Γ . Here, as usual, a congruence relation is an equivalence relation that is compatible with the monoid operation, i.e., such that $w \sim v$ and $w' \sim v'$ implies $ww' \sim vv'$. Given a set X of generators for a monoid G and a set Γ of valid relations, we say that Γ is *complete* if for all $w, v \in X^*$, $[w]_G = [v]_G$ implies $w \sim_\Gamma v$. In that case, we also say that (X, Γ) is a *presentation by generators and relations* (or simply *presentation*) of G .

Definition 4.2. Given sets X, Y and a function $f : X \rightarrow Y^*$, let $f^* : X^* \rightarrow Y^*$ be the unique monoid homomorphism extending f . Concretely, f^* is given by $f^*(x_1 \dots x_n) = f(x_1) \dots f(x_n)$.

More generally, given sets C, X, Y and a function $f : C \times X \rightarrow Y^* \times C$, let $f^{**} : C \times X^* \rightarrow Y^* \times C$ be the function defined by $f^{**}(c_0, x_1 \dots x_n) = (w_1 \dots w_n, c_n)$, where $f(c_{i-1}, x_i) = (w_i, c_i)$ for all $i = 1, \dots, n$.

Note that in case C is a singleton, the functions f^* and f^{**} are essentially the same. In general, the difference is that f^{**} also keeps a “state” in the form of an element of C .

Theorem 4.3 (Reidemeister-Schreier theorem for monoids). *Let X and Y be sets, and let Γ and Δ be sets of relations over X and Y , respectively. Suppose that the following additional data is given:*

- a set C with a distinguished element $I \in C$,
- a function $f : X \rightarrow Y^*$,
- a function $h : C \times Y \rightarrow X^* \times C$,

subject to the following conditions:

- (a) *For all $x \in X$, if $h^{**}(I, f(x)) = (v, c)$, then $v \sim_\Gamma x$ and $c = I$.*
- (b) *For all $c \in C$ and $w, w' \in Y^*$ with $(w, w') \in \Delta$, if $h^{**}(c, w) = (v, c')$ and $h^{**}(c, w') = (v', c'')$ then $v \sim_\Gamma v'$ and $c' = c''$.*

Then for all $v, v' \in X^$, $f^*(v) \sim_\Delta f^*(v')$ implies $v \sim_\Gamma v'$.*

To better understand the utility of this theorem, let us briefly provide some context. First, we note that we will be using this theorem in the case where G is a monoid, H is a submonoid of G , (Y, Δ) is a presentation of G , X is a set of generators for H , and we wish to show that some proposed set of relations Γ is complete for H . Assuming that all hypotheses of Theorem 4.3 are satisfied, and further assuming that f represents the inclusion function of H into G , i.e., that for all $x \in X$, $[f(x)]_G = [x]_H$, the completeness of Γ then follows. Namely, $[v]_H = [v']_H$ implies $[f^*(v)]_G = [f^*(v')]_G$, which implies $f^*(v) \sim_\Delta f^*(v')$ by completeness of Δ , which implies $v \sim_\Gamma v'$ by Theorem 4.3.

To see how the theorem works, it is useful to further concentrate on the case where G and H are groups, although the theorem itself does not require this. In the case of groups, one would typically

consider the set $H \backslash G = \{Hc \mid c \in G\}$ of right cosets of H in G , and one would let C be a set of chosen coset representatives. The function f is then chosen to assign to each $x \in X$ some word $w \in Y^*$ such that $[x]_H = [w]_G$. The function h is chosen to assign to each pair of a coset representative $c \in C$ and generator $y \in Y$ the unique coset representative $c' \in C$ and some word $v \in X^*$ such that $c[y]_G = [v]_H c'$. Conditions (a) and (b) are then sufficient for the set of relations Γ to be complete. In the more general case of monoids, G is not necessarily partitioned into cosets, but the method works anyway, provided that appropriate C , f , and h can be chosen.

Proof of Theorem 4.3. Let us say that a word $w \in Y^*$ is *special* if $h^{**}(I, w) = (v, I)$ for some $v \in X^*$. Let Y_s^* be the set of special words. By definition of h^{**} , the empty word is special and special words are closed under concatenation, so Y_s^* is a submonoid of Y^* . Moreover, the image of f is special by property (a), and therefore the image of f^* is also special. Finally, there is a translation back from special words in Y to words in X : define $g : Y_s^* \rightarrow X^*$ by letting $g(w) = v$ where $h^{**}(I, w) = (v, I)$. Clearly, g is a monoid homomorphism.

Claim A: for all $v \in X^*$, we have $v \sim_{\Gamma} g(f^*(v))$. Proof: Since both g and f^* are monoid homomorphisms and \sim_{Γ} is a congruence, it suffices to show this in the case when $v \in X$ is a generator. But in that case, it holds by assumption (a).

Claim B: for all $w, w' \in Y^*$ and $c \in C$, if $w \sim_{\Delta} w'$ and $h^{**}(c, w) = (v, d)$ and $h^{**}(c, w') = (v', d')$, then $v \sim_{\Gamma} v'$ and $d = d'$. Proof: define a relation \sim on Y^* by $w \sim w'$ if for all $c \in C$, $h^{**}(c, w) = (v, d)$ and $h^{**}(c, w') = (v', d')$ implies $v \sim_{\Gamma} v'$ and $d = d'$. We must show that $w \sim_{\Delta} w'$ implies $w \sim w'$. Since \sim_{Δ} is, by definition, the smallest congruence containing Δ , it suffices to show that \sim is a congruence containing Δ . The fact that \sim is reflexive, symmetric, and transitive is obvious from its definition. The fact that it is a congruence follows from the definition of h^{**} and the fact that \sim_{Γ} is a congruence. Finally, \sim contains Δ by assumption (b).

Note that, as a special case of claim B, we also have the following: if $w, w' \in Y_s^*$ are special words, then $w \sim_{\Delta} w'$ implies $g(w) \sim_{\Gamma} g(w')$. This follows directly from the definition of g .

To finish the proof of the Reidemeister-Schreier theorem, let $v, v' \in X^*$ and assume that $f^*(v) \sim_{\Delta} f^*(v')$. Then we have:

$$v \sim_{\Gamma} g(f^*(v)) \sim_{\Gamma} g(f^*(v')) \sim_{\Gamma} v',$$

where the first and last congruence holds by claim A, and the middle one holds by the special case of claim B. Therefore, $v \sim_{\Gamma} v'$ as claimed. \square

Corollary 4.4. *Let G be a monoid with presentation (Y, Δ) , where $Y \subseteq G$. Suppose $H \subseteq G$ is a submonoid and X is a set of generators for H . Let Γ be a set of valid relations for H . Assume a set C and functions f and h are given, satisfying the hypotheses of Theorem 4.3, and assume that f represents the inclusion function of H into G , i.e., that $x \in X$, $[f(x)]_G = [x]_H$. Then Γ is a complete set of relations for H . \square*

4.3 Pauli rotation representation

One of the problems we face in applying the Reidemeister-Schreier theorem is that we must show that a large number of (computer-generated) Clifford+ T relations follow from the relations in Figure 1. It would be very useful if this task could be automated. Ideally, the relations in Figure 1 could be turned into a set of rewrite rules with the property that every Clifford+ T circuit can be rewritten to a unique *normal form*; in that case, to show that a given relation follows from the ones in Figure 1, it would be sufficient to reduce the left-hand and right-hand sides to normal form and check that they are equal.

Unfortunately, no such rewrite system or normal form is known. Instead, the best we can do is a semi-automated process in which words are rewritten to something that is “almost” a normal form, i.e., not quite unique, but close enough so that many relations can be proved automatically, and the rest are more easily solvable by hand.

For this, the *Pauli rotation representation* of Clifford+T operators turns out to be useful. This representation was first described in [12, Section 3]. We start by noting that the T -gate is a linear combination of the identity I and the Pauli operator Z . Specifically:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} = \frac{1+\omega}{2}I + \frac{1-\omega}{2}Z. \quad (1)$$

Therefore, an operator A commutes with T if and only if it commutes with Z . More generally, given any n -qubit Pauli operator P , define

$$R_P = \frac{1+\omega}{2}I + \frac{1-\omega}{2}P. \quad (2)$$

Note that $R_Z = T$. We refer to the operators R_P as (*45 degree*) *Pauli rotations*. Note that R_P is not a Pauli operator; we call it a Pauli rotation because it is a rotation about a Pauli axis. By (2), it is again obvious that an operator A commutes with R_P if and only if it commutes with P . Moreover, from (2), we get the following fundamental property of Pauli rotations:

$$CPC^{-1} = Q \quad \text{if and only if} \quad CR_PC^{-1} = R_Q. \quad (3)$$

Let $Z_{(i)} = I \otimes \dots \otimes I \otimes Z \otimes I \otimes \dots \otimes I$ be the n -qubit Pauli operator with Z acting on the i th qubit, and similarly $T_{(i)} = I \otimes \dots \otimes I \otimes T \otimes I \otimes \dots \otimes I = R_{Z_{(i)}}$. Since the Clifford operators act transitively on the set of non-trivial self-adjoint Pauli operators by conjugation, for every such n -qubit Pauli operator P , there exists a (non-unique) Clifford operator C such that $CZ_{(1)}C^{-1} = P$, and therefore $CT_{(1)}C^{-1} = R_P$. We therefore see that all of the Pauli rotations are Clifford conjugates of the $T_{(1)}$ -gate.

Next, we note that every Clifford+T operator can be written as a product of Pauli rotations followed by a single Clifford operator. Specifically, by definition, every Clifford+T operator can be written as

$$C_1T_{(i_1)}C_2T_{(i_2)}C_3 \cdots C_nT_{(i_n)}C_{n+1}.$$

For all k , let $D_k = C_1C_2 \cdots C_k$, so that $C_k = D_{k-1}^{-1}D_k$. Then the above can be rewritten as

$$\begin{aligned} C_1T_{(i_1)}C_2T_{(i_2)}C_3 \cdots C_nT_{(i_n)}C_{n+1} &= C_1R_{Z_{(i_1)}}C_2R_{Z_{(i_2)}}C_3 \cdots C_nR_{Z_{(i_n)}}C_{n+1} \\ &= D_1R_{Z_{(i_1)}}D_1^{-1}D_2R_{Z_{(i_2)}}D_2^{-1}D_3 \cdots D_{n-1}R_{Z_{(i_n)}}D_{n-1}^{-1}D_nR_{Z_{(i_n)}}D_n^{-1}D_{n+1} \\ &= R_{D_1Z_{(i_1)}D_1^{-1}}R_{D_2Z_{(i_2)}D_2^{-1}} \cdots R_{D_nZ_{(i_n)}D_n^{-1}}D_{n+1} \\ &= R_{P_1}R_{P_2} \cdots R_{P_n}D_{n+1}, \end{aligned}$$

where $P_k = D_kZ_{(i_k)}D_k^{-1}$. Therefore, every Clifford+T operator can be written as a product of Pauli rotations followed by a single Clifford operator, as claimed. It also shows that the number of required Pauli rotations is at most equal to the T -count of the original circuit. In fact, since every Pauli rotation has T -count 1, it is clear that every product of n Pauli rotations can be converted to a circuit of T -count n , and vice versa. In particular, the minimal T -count of a circuit is equal to the minimal number of Pauli rotations required to express it.

The Pauli rotation representation is not unique. There are some obvious relations:

- (a) R_P and R_Q commute if and only if P and Q commute. This follows from (2).

- (b) For any P , the operator R_P^2 is Clifford, and therefore can be eliminated, resulting in a shorter word. To see why, recall that there exists a Clifford operator C such that $R_P = CT_{(1)}C^{-1}$; therefore $R_P^2 = CT_{(1)}^2C^{-1}$. Since $T_{(1)}^2 = S_{(1)}$ is a Clifford gate, it follows that R_P^2 is Clifford.
- (c) For any P , there exists a Clifford operator D such that $R_{(-P)} = R_P D$. Indeed, let C be a Clifford operator such that $P = CZ_{(1)}C^{-1}$. Then $-P = C(-Z_{(1)})C^{-1} = CX_{(1)}Z_{(1)}X_{(1)}C^{-1}$. Therefore $R_{(-P)} = CX_{(1)}T_{(1)}X_{(1)}C^{-1}$. Using the relation $XTX = TS^\dagger\omega$, we have $R_{(-P)} = CT_{(1)}S_{(1)}^\dagger\omega C^{-1} = CT_{(1)}C^{-1}CS_{(1)}^\dagger\omega C^{-1} = R_P CS_{(1)}^\dagger\omega C^{-1}$. Thus, the claim holds with $D = CS_{(1)}^\dagger\omega C^{-1}$.

It is relatively easy to standardize the Pauli rotation representation modulo the above three relations: First, we eliminate any generators of the form $R_{(-P)}$. This can be done from left to right, using relations from (c); the resulting Clifford operator can be shifted all the way to the end of the word using relations of the form $DR_P = R_Q D$, where $Q = DPD^{-1}$, see (3). Next, we use relations from (a) to swap adjacent generators when possible, for example arriving at the lexicographically smallest word that is equal to the given word up to such commuting permutations. Next, we use relations from (b) to remove any duplicates. Should there be any such duplicates, the resulting word will need to be standardized again, but since it uses fewer Pauli rotations, the process eventually terminates.

However, even when the Pauli rotation representation is standardized modulo the relations (a), (b), and (c), it is still not unique. Indeed, there are some “non-obvious” relations. In a sense, the contribution of this paper is to state exactly what these non-obvious relations are. They turn out to be the following. Here, for brevity, we have omitted the tensor symbol \otimes , i.e., we wrote R_{IX} instead of $R_{I\otimes X}$.

$$\begin{aligned} R_{IX}R_{IZ}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{ZZ}R_{IX}, \\ R_{IX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{IX}, \\ R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ} &= R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ}R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}. \end{aligned}$$

These turn out to be equivalent to relations (C18), (C19), and (C20) in Figure 1, respectively. We will address the question of what these relations might “mean” (i.e., how one might be able to see that they are true without computing the matrices) in Section 6.

4.4 Proof assistants

As outlined in Section 3, once we are armed with the Reidemeister-Schreier theorem, in theory there is a mechanical way to obtain a complete set of relations for $\mathcal{C}\mathcal{T}(2)$, given that $\mathcal{C}\mathcal{T}(2)$ is a subgroup of $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$ and we already have a complete set of relations for the latter due to Greylyn [13]. However, when applied in practice, this method yields a large number of very large relations, all of which must be shown to follow from the relations in Figure 1. Although Figure 3 appears to contain only 20 relations, they are actually parameterized by indices such as j, k , etc. After accounting for these indices, there are 123 distinct relations. Since there are two cosets of $\mathcal{C}\mathcal{T}(2)$ in $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$, under part (b) of the Reidemeister-Schreier theorem, each of these 123 relations yields two Clifford+ T relations, plus another 8 relations (one for each generator) from part (a), giving a total of 254 Clifford+ T relations that must be verified. This task is too daunting to do “by hand”.

Given the mechanical and repetitive nature of these calculations, we initially wrote a computer program to generate and verify the relations. However, this raised another issue: our program was large and complicated and used a variety of tactics to show that the given relations follow from the ones in Figure 1. We could not claim with mathematical certainty that our program was free of bugs, nor that it didn’t use some hidden assumptions that weren’t actually consequences of Figure 1. Moreover, it would have been unreasonable for any referee to verify our calculations.

For this reason, we decided to go one step further and formalize the soundness and completeness proofs in a *proof assistant*. A proof assistant is a piece of software in which one can write definitions, theorems, and proofs, and the software will check the correctness of the proofs. Purists might object that the proof assistant is itself a piece of software that might be buggy. But, as has been argued eloquently by [11, 14], current proof assistants can be scrutinized at many levels and are many orders of magnitude more reliable than the traditional way of checking paper-and-pencil proofs. The particular proof assistant we used in this work is Agda [1].

5 Proof of the main result

5.1 Soundness and completeness

Our goal is to prove that Theorem 4.1 implies Theorem 2.1. Recall that Greylyn’s set of generators for $U_4(\mathbb{R})$ is $\mathcal{Y} = \{\omega_{[j]}, X_{[j,k]}, H_{[j,k]} \mid j, k \in \{1, \dots, 4\} \text{ and } j < k\}$. Also recall that our target set of generators for $\mathcal{C}\mathcal{T}(2)$ is $\mathcal{X} = \{\omega, H_0, H_1, S_0, S_1, T_0, T_1, CZ\}$. We fix a translation from \mathcal{X} to \mathcal{Y}^* as follows:

$$\begin{aligned} f(\omega) &= \omega_{[0]} \omega_{[1]} \omega_{[2]} \omega_{[3]}, \\ f(H_0) &= H_{[1,3]} H_{[0,2]}, \\ f(H_1) &= H_{[2,3]} H_{[0,1]}, \\ f(S_0) &= \omega_{[2]}^2 \omega_{[3]}^2, \\ f(S_1) &= \omega_{[1]}^2 \omega_{[3]}^2, \\ f(T_0) &= \omega_{[2]} \omega_{[3]}, \\ f(T_1) &= \omega_{[1]} \omega_{[3]}, \\ f(CZ) &= \omega_{[3]}^4. \end{aligned}$$

We prove the following soundness and completeness theorems for this translation:

Theorem 5.1 (Soundness). *For all $w, v \in \mathcal{X}^*$, $w \sim_{\Gamma} v$ implies $f^*(w) \sim_{\Delta} f^*(v)$.*

Theorem 5.2 (Completeness). *For all $w, v \in \mathcal{X}^*$, $f^*(w) \sim_{\Delta} f^*(v)$ implies $w \sim_{\Gamma} v$.*

As already noted in Section 4.2, these two theorems, together with Theorem 4.1, immediately imply Theorem 2.1. Specifically, we have $w \sim_{\Gamma} v$ if and only if $f^*(w) \sim_{\Delta} f^*(v)$ if and only if $[f^*(w)] = [f^*(v)]$ if and only if $[w] = [v]$, where the first equivalence follows from Theorems 5.1 and 5.2, the second equivalence follows from Theorem 4.1, and the last equivalence holds because the function f respects the interpretation.

5.2 The formal proof

Soundness and completeness are formally proved in the Agda code accompanying this paper [8]. We organized the code to make it hopefully as easy as possible to verify the result. The code consists of 67 files that are listed in Figure 4, and which we now briefly describe.

(a) Background. The eight files in the “background” section contain general-purpose definitions of the kind that are usually found in the Agda standard library, i.e., basic properties of booleans, integers, equality, propositional connectives, etc. The reason we did not use the actual Agda standard library is that it is very large and changes frequently. We felt that it is better for our code to be self-contained rather than depending on a particular library version.

(b) Statement of the result. In these two files, we give a minimal set of definitions that allows us to *state* the soundness and completeness theorems. The file `Word.agda` defines what it means to be a word over a set of generators, as well as the inference rules we use for deriving relations from a set of axioms (such as reflexivity, symmetry, transitivity, congruence, associativity, and the left and right unit laws). Note that in the Agda code, we define a word as a term in the language of monoids, rather than as a sequence of generators. In other words, associativity and the unit laws are treated as laws, rather than being built into the definition. The file `Word.agda` also defines the f^* operation used in the statement of the soundness and completeness theorems. The file `Generator.agda` defines the Clifford+ T generators and the relations from Figure 1, Greylyn’s generators and the relations from Figure 3, and the translation function f from Section 5.1. It also contains the statement of the soundness and completeness theorems, but not their proofs. The reason we state these theorems separately from their proofs is to make sure that Agda (and a human reviewer) can verify that the statement of these theorems only depends on the relatively small number of definitions given so far, and not on the much larger number of definitions and tactics used in the proof.

(c) Details of the proof. The proof of the soundness and completeness theorems relies on a large number of auxiliary definitions and lemmas, and comprises the bulk of our code with 56 files. This includes a formal proof of the Reidemeister-Schreier theorem; several tactics for automating steps in certain equational proofs; a simplified presentation of Greylyn’s generators and relations, using only 5 generators and 19 relations (instead of Greylyn’s original 16 generators and 123 relations), along with the proof of its completeness; a formalization of Pauli rotations and their relevant properties; as well as 46 step-by-step proofs of individual relations. These details are primarily intended to be machine-readable, and can safely be skipped by readers who trust Agda and merely want to check the proof rather than reading it. However, all of the files are documented and human-readable.

The relations in the files `Equation1.agda` to `Equation46.agda` are at the heart of the completeness proof. These are the relations that must be proved to satisfy the hypotheses of the Reidemeister-Schreier theorem. Some of these relations are trivial, such as `Equation13.agda`. Others are highly non-trivial and require almost a thousand proof steps, such as `Equation44.agda`. In particular, the proofs that require relations (C18)–(C20) from Figure 1 tend to be non-obvious; in fact, this is how we discovered relations (C18)–(C20) in the first place. We did not write these equational proofs by hand; instead, we used a semi-automated process where most of the proofs were generated by a separate Haskell program and output in a format that is convenient and efficient for Agda to check. Originally, we also attempted to write Agda tactics that would allow Agda to derive these relations fully automatically; however, this failed due to performance issues with Agda.

(d) Proof witness. Finally, the file `Proof.agda` contains nothing but a witness of the fact that the soundness and completeness theorems have been formally proven. A reader who wants to skip the details of the formal proof only needs to check two things: the statement of the main result in `Generator.agda` (to make sure the statement correctly captures what we said it does), and the fact that the Agda proof checker accepts `Proof.agda`.

6 Discussion of the axioms

Here, we give some further perspectives on what the axioms of Figure 1 might “mean”, and in particular, how one might convince oneself that the relations are true without having to compute the corresponding

(a) Background:

<code>Boolean.agda</code>	The type of booleans.
<code>Proposition.agda</code>	Basic definitions in propositional logic.
<code>Equality.agda</code>	Basic properties of equality.
<code>Decidable.agda</code>	Some definitions to deal with decidable properties.
<code>Inspect.agda</code>	Agda’s “inspect” paradigm, to assist with pattern matching.
<code>Nat.agda</code>	Basic properties of the natural numbers.
<code>Maybe.agda</code>	The “Maybe” type.
<code>List.agda</code>	Basic properties of lists.

(b) Statement of the result

<code>Word.agda</code>	Basic properties of words.
<code>Generator.agda</code>	Generators and relations for our two groups, and statement of main result.

(c) Proof of the result

<code>Word-Lemmas.agda</code>	Basic lemmas about monoids and groups, and equational reasoning.
<code>Reidemeister-Schreier.agda</code>	Two versions of the Reidemeister-Schreier theorem.
<code>Word-Tactics.agda</code>	Some tactics for proving properties of words.
<code>Clifford-Lemmas.agda</code>	A decision procedure for equality of 2-qubit Clifford operators.
<code>CliffordT-Lemmas.agda</code>	Properties and tactics for Clifford+ T operators.
<code>Greylyn-Lemmas.agda</code>	Some automation for Greylyn’s 1- and 2-level operators.
<code>Soundness.agda</code>	Proof of soundness.
<code>Greylyn-Simplified.agda</code>	A smaller set of generators and relations for Greylyn’s operators.
<code>PauliRotations.agda</code>	Definitions, properties, and tactics for Pauli rotations.
<code>Equation1.agda – Equation46.agda</code>	Explicit proofs of 46 relations required for completeness.
<code>Completeness.agda</code>	Proof of completeness.

(d) Top-level proof witness

<code>Proof.agda</code>	The final witness for soundness and completeness.
-------------------------	---

Figure 4: List of Agda files. The files are listed in order of dependency, i.e., each file only imports earlier files.

matrices.

Note that we are not claiming that axioms (C1)–(C20) are independent; for example, (C8) clearly follows from (C14) and (C16); however, we found it useful to separate the Clifford relations from the rest, which is why (C8) was included. It would be nice to know whether axioms (C18)–(C20) are independent from the others and from each other, and this seems likely to be true, but we do not know.

The axioms in groups (a)–(c) are well-known; they merely express the Clifford relations [21] and the fact that operators on disjoint qubits commute. Relations (C14) and (C15) express the well-known facts that $T^2 = S$ and $(TX)^2 = \omega$, whereas relation (C16) holds because diagonal operators commute. Note that the upside-down version of relation (C16) was not included among our axioms; this is because it is actually derivable from the remaining axioms. Relation (C17) becomes obvious once one realizes that the swap gate can be expressed as a sequence of three controlled-not gates:

$$\text{swap} = \text{CNOT}_{12} \text{CNOT}_{21} \text{CNOT}_{12}$$

Relation (C17) is then obtained by simplifying the following, which expresses the fact that a T -gate can be moved past a swap-gate:

$$\text{swap} \cdot T = T \cdot \text{swap}$$

We will now focus on the “non-obvious” relations (C18)–(C20). Relations (C18) and (C19) are of the form

$$\text{CNOT}_{12} \cdot A \cdot \text{CNOT}_{12} = \text{CNOT}_{12} \cdot A^\dagger \cdot \text{CNOT}_{12} \quad (4)$$

They hold because positively controlled gates commute with negatively controlled gates. Note that there are infinitely many relations of the form (4), where A is any single-qubit Clifford+ T operator, but our completeness proof shows that, in the presence of the remaining axioms, two of them are sufficient to prove all the others.

Relation (C20) is more interesting. It, too, states that two operators commute, but it is less obvious why this is so. Ideally, we would be able to find some simpler and more obvious relations that imply (C20). While we have not been able to find such simpler relations in the Clifford+ T generators, we can do this if we permit ourselves a controlled T -gate. Note that the controlled T -gate is not itself a member of the 2-qubit Clifford+ T group, since representing it as a Clifford+ T operator requires an ancilla [10]. But the use of controlled T -gates is nevertheless helpful in explaining relation (C20). We start by noting that the controlled T -gate satisfies the following obvious circuit identities (and their upside-down versions):

$$\text{CNOT}_{12} \cdot T = T \cdot \text{CNOT}_{12} \quad (5)$$

$$\text{CNOT}_{12} \cdot T \cdot \text{CNOT}_{12} = T \quad (6)$$

$$\text{CNOT}_{12} \cdot T \cdot \text{CNOT}_{12} = T \cdot \text{CNOT}_{12} \quad (7)$$

$$\text{CNOT}_{12} \cdot H \cdot T \cdot H = T \cdot H \cdot T \cdot H \quad (8)$$

Identities (5)–(7) are obvious because all of the operators in them are diagonal. Identity (8) holds by case distinction: this circuit applies either HT or TH to the bottom qubit, depending on whether the top qubit

is $|0\rangle$ or $|1\rangle$. Using these identities, we can easily prove (C20):

$$\begin{array}{l}
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \circ \text{---} \end{array} \quad \begin{array}{c} \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \circ \text{---} \\ | \\ \text{---} \circ \text{---} \end{array} \\
\text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \circ \text{---} \\
\text{---} \circ \text{---} \\
\text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \circ \text{---} \\
\text{---} \circ \text{---}
\end{array} \quad \begin{array}{l}
\stackrel{(8)}{=} \\
\stackrel{(5)}{=} \\
\stackrel{(6)}{=} \\
= \\
\stackrel{(5),(7)}{=} \\
= \\
\stackrel{(6)}{=} \\
\stackrel{(5)}{=} \\
\stackrel{(8)}{=}
\end{array} \quad \begin{array}{c}
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array} \\
\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \text{H} \text{---} \text{T} \text{---} \text{H} \text{---} \text{T} \text{---} \circ \text{---} \end{array}
\end{array}
\end{array}$$

Note that there is again an infinite family of such relations, because in the above derivation, we could have used any gate in place of H . However, due to completeness, all other such relations are consequences of (C18)–(C20) and the remaining axioms.

Another way to look at relations (C18)–(C20) is in terms of their Pauli rotation representations. As we already mentioned in Section 4.3, up to basis changes, the three relations can be written in terms of Pauli rotations, respectively as follows:

$$\begin{aligned}
R_{IX}R_{IZ}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{ZZ}R_{IX}, \\
R_{IX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{IX}, \\
R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ} &= R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ}R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}.
\end{aligned}$$

When written in this form, the first two of these relations only use X and Z Paulis, and use only Z on the left qubit. This indicates that these relations are about controlled gates. We can also see that in both cases, the relation exchanges the positions of the leftmost R_{IX} and the rightmost R_{ZX} . The first relation can also be seen to express the fact that $R_{IZ}R_{ZZ}$ commutes with $R_{ZX}R_{IX}^{-1}$, and similarly for the second relation. The third relation again takes the form of an operator commuting with its upside-down version.

7 Conclusion and future work

We gave a presentation of the 2-qubit Clifford+T group by generators and relations. We did this by applying the Reidemeister-Schreier theorem to Greylyn's presentation of the group of unitary 4×4 -matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. Since there is a very large number of relations to check and simplify, and checking them by hand or by an unverified computer program would be error-prone, we used the proof assistant Agda to formalize our proof. The latter process is painstaking and took us more than 5 years to complete after our result was first announced in [7].

An obvious candidate for future work would be to find a complete set of relations for the Clifford+ T group with 3 or more qubits. This is currently out of reach for two reasons: first, the computations required to simplify any potential set of relations will be even more labor-intensive than in the 2-qubit case. Second, and more seriously, there is no known presentation of the group of unitary $n \times n$ -matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ for $n > 4$.

Another project that is currently in progress is to apply the method of this paper to restrictions of the Clifford+ T group for which presentations of the corresponding matrix group are known. This includes the Clifford+Toffoli gate set and the Clifford+controlled- S gate set.

References

- [1] *Agda Documentation*. <https://agda.readthedocs.io/>. Accessed: 2022-02-15.
- [2] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T -Depth Optimization of Clifford+ T Circuits Via Matroid Partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953. Also available from arXiv:1303.2042.
- [3] Matthew Amy, Dmitri Maslov, Michele Mosca & Martin Roetteler (2013): *A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32(6), pp. 818–830, doi:10.1109/TCAD.2013.2244643. Also available from arXiv:1206.0758v2.
- [4] Matthew Amy & Michele Mosca (2019): *T -count optimization and Reed-Muller codes*. *IEEE Transactions on Information Theory* 65(8), pp. 4771–4784, doi:10.1109/TIT.2019.2906374. Also available from arXiv:1601.07363.
- [5] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Fast and effective techniques for T -count reduction via spider nest identities*. In Steven T. Flammia, editor: *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020), Leibniz International Proceedings in Informatics (LIPIcs)* 158, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 11:1–23, doi:10.4230/LIPIcs.TQC.2020.11. Also available from arXiv:2004.05164.
- [6] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Techniques to reduce $\pi/4$ -parity-phase circuits, motivated by the ZX calculus*. *Electronic Proceedings in Theoretical Computer Science* 318, p. 131–149, doi:10.4204/eptcs.318.9. Available from arXiv:1911.09039.
- [7] Xiaoning Bian & Peter Selinger (2015): *Relations for the 2-qubit Clifford+ T operator group*. Slides presented at the Workshop on Quantum Programming and Circuits, Waterloo, Canada, June 8–11, 2015. Available from https://mathstat.dal.ca/~xbian/talks/slide_cliffordt2.pdf.
- [8] Xiaoning Bian & Peter Selinger (2022): *Agda code accompanying this paper*. Available as ancillary material at arXiv:2204.02217.
- [9] Harry Buhrman, Richard Cleve, Monique Laurent, Noah Linden, Alexander Schrijver & Falk Unger (2006): *New Limits on Fault-Tolerant Quantum Computation*. In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pp. 411–419, doi:10.1109/FOCS.2006.50. Also available from arXiv:quant-ph/0604141.
- [10] Brett Giles & Peter Selinger (2013): *Exact synthesis of multiqubit Clifford+ T circuits*. *Physical Review A* 87(3), p. 032332 (7 pages), doi:10.1103/PhysRevA.87.032332. Also available from arXiv:1212.0506.
- [11] Georges Gonthier (2008): *Formal Proof—The Four Color Theorem*. *Notices of the American Mathematical Society* 55(11), pp. 1382–1393.
- [12] David Gosset, Vadym Kliuchnikov, Michele Mosca & Vincent Russo (2014): *An Algorithm for the T -Count*. *Quantum Information and Computation* 14(15–16), pp. 1261–1276, doi:10.26421/QIC14.15-16-1. Also available from arXiv:1308.4134.

- [13] Seth E. M. Greylyn (2014): *Generators and relations for the group $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$* . M.Sc. thesis, Dalhousie University. Available from arXiv:1408.6204.
- [14] Thomas C. Hales (2008): *Formal Proof*. *Notices of the American Mathematical Society* 55(11), pp. 1370–1380.
- [15] Luke E. Heyfron & Earl T. Campbell (2018): *An efficient quantum compiler that reduces T count*. *Quantum Science and Technology* 4(1), p. 015004, doi:10.1088/2058-9565/aad604. Also available from arXiv:1712.01557.
- [16] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Phys. Rev. A* 102, p. 022406, doi:10.1103/PhysRevA.102.022406. Preprint available from arXiv:1903.10477.
- [17] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs & Dmitri Maslov (2018): *Automated optimization of large quantum circuits with continuous parameters*. *NPJ Quantum Information* 4(1), doi:10.1038/s41534-018-0072-4. Also available from arXiv:1710.07345.
- [18] Michael A. Nielsen & Isaac L. Chuang (2002): *Quantum Computation and Quantum Information*. Cambridge University Press.
- [19] Kurt Reidemeister (1927): *Knoten und Gruppen*. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 5(1), pp. 7–23, doi:10.1007/BF02952506.
- [20] Otto Schreier (1927): *Die Untergruppen der freien Gruppen*. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 5(1), pp. 161–183, doi:10.1007/BF02952517.
- [21] Peter Selinger (2015): *Generators and Relations for n-Qubit Clifford Operators*. *Logical Methods in Computer Science* 11(2:10), pp. 1–17, doi:10.2168/LMCS-11(2:10)2015. Also available from arXiv:1310.6813.

Reducing 2-QuBit Gate Count for ZX-Calculus based Quantum Circuit Optimization

Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann

MNM-Team

Ludwig-Maximilians-Universität München
Munich, Germany

{staudacher, guggemos, grundner-culemann}@nm.ifi.lmu.de

Wolfgang Gehrke

Research Institute CODE

Universität der Bundeswehr München Munich, Germany

wolfgang.gehrke@unibw.de

In the near term, programming quantum computers will remain severely limited by low quantum volumes. Therefore, it is desirable to implement quantum circuits with the fewest resources possible. For the common Clifford+T circuits, most research is focused on reducing the number of T gates, since they are an order of magnitude more expensive than Clifford gates in quantum error corrected encoding schemes. However, this optimization sometimes leads to more 2-qubit gates, which, even though they are less expensive in terms of fault-tolerance, contribute significantly to the overall circuit cost. Approaches based on the *ZX-calculus* have recently gained some popularity in the field, but reduction of 2-qubit gates is not their focus. In this work, we present an alternative for improving 2-qubit gate count of a quantum circuit with the *ZX-calculus* by using heuristics in *ZX*-diagram simplification. Our approach maintains the good reduction of the T gate count provided by other strategies based on *ZX-calculus*, thus serving as an extension for other optimization algorithms. Our results show that combining the available *ZX-calculus*-based optimizations with our algorithms can reduce the number of 2-qubit gates by as much as 40% compared to current approaches using *ZX-calculus*. Additionally, we improve the results of the best currently available optimization technique of Nam et. al [22] for some circuits by up to 15%.

1 Introduction

Many famous quantum algorithms, like Shor [26], HHL [13] or Grover [12], base upon techniques like Quantum Fourier Transformation, Quantum Phase Estimation or Amplification, respectively. Although these algorithms provide significant (sometimes even exponential) speed-ups, current quantum chips can only execute toy problems, mostly due to the low gate fidelity. Even for problems that can be easily solved on a state-of-the-art desktop PC, those algorithms require tens of thousands of gates, and are therefore infeasible to run on near-term quantum devices. However, applications in quantum simulation are supposed to achieve significant improvements in quantum chemistry, material sciences, or high-energy physics on near-term devices. With variational algorithms (e.g., QAOA [10] or VQE [25]), real-world applications like optimization problems on real quantum chips may become feasible. While the associated speed-up is unknown for many use cases, they require only few qubits and quantum gates to achieve promising results. Quantum Machine Learning (QML) is such an example: Here, the combination of clever encoding strategies, variational algorithms, and classical pre- and post-processing achieves high accurate classification rates with fewer qubits compared to classical bits.

Still, even algorithms with smaller circuits cannot be executed on current devices and gate optimization is a vibrant research topic. While global optimization of arbitrary quantum circuits is generally QMA-hard [16], different algorithms like quantum optimal control [18] have been proposed to reduce the size of a quantum circuit. In this context the so-called ZX-calculus [6] is considered a promising tool. It provides an abstract graphical language for describing quantum systems and can be seen as an alternative to the predominant description in the Hilbert space. We can transform any quantum circuit into a ZX-diagram equivalent, apply the rules of the ZX-calculus to simplify the diagram, and re-extract a quantum circuit from it.

Scope of this work

Our work is based on optimizing circuits with ZX-calculus, where several optimization strategies have been proposed recently [9, 20, 27]. Currently, these strategies yield very good results for pure *Clifford* circuits, as well as for T gate elimination in *Clifford+T* circuits, which is worthwhile for fault-tolerant quantum computers with error-corrected gates. For such devices, the cost of a T gate is sometimes estimated to be up to a hundred times higher than the cost of a CNOT gate [24] (even though recent studies suggest lower rates [21]).

However, reducing 2-qubit gates is generally of interest for quantum hardware that is not error corrected (e.g., NISQ devices) or in which quantum states do not tend to interact easily, e.g., in Photonic Quantum computing [3]. A major drawback of the current ZX-calculus based strategies is that these gates in particular are not optimized very well; in fact, for many large Clifford+T circuits, the 2-qubit gate count even increases when using algorithms like the one in [9].

We propose new optimization approaches especially for reducing 2-qubit gates. To do so, we use heuristics for estimating the 2-qubit gate count in ZX-diagrams as cost functions for classical search algorithms like **I.**) random selection and **II.**) greedy algorithm. By combining them with existing optimization approaches, we maintain the T gate count reduction rate and improve the total gate count and the 2-qubit gate count for most given circuits. We evaluate the performance on circuits from the *Tpar* benchmark [1]. We find that our optimizations can outperform existing ZX-based approaches and can additionally be used to further improve already optimized circuits.

2 Background

Throughout this paper, we use the notation from [23] for quantum gates; the most essential ones are detailed in Table 1 by name and matrix-, gate- and ZX-calculus-representation. Every unitary operation can be decomposed into a combination of CNOTs and single-qubit gates [23]. A well-studied example for a minimal gate set with which to approximate any unitary operation is the so-called *Clifford+T set*, i.e., the gate set generated by $\{H, T, CNOT\}$. That is why many optimization algorithms target circuits generated with the *Clifford+T set*. The *Clifford* set generated by $\{H, S, CNOT\}$ is also well-known and useful for quantum circuit simulations on classical computers, but not every unitary operation can be represented with it. For convenience, we abbreviate some gates in the *Clifford+T* set, namely X, Y, Z, S , and CZ (instead of writing, for example, $Z = T \cdot T \cdot T \cdot T$).

2.1 ZX-Calculus

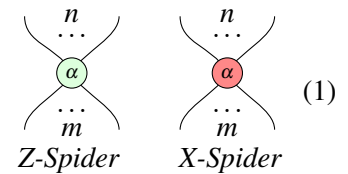
Since ZX-calculus and its optimization strategies rely on graph theory, we provide some background in Appendix B. The ZX-calculus [7, 8] is a graphical language for expressing linear maps on qubits as

Table 1: Overview of important quantum gates and the respective ZX-Spiders.

Name	Identity	Z	Z-Phase	T	X	X-Phase	H	CNOT
Matrix	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{2} \begin{pmatrix} 1+e^{i\alpha} & 1-e^{i\alpha} \\ 1-e^{i\alpha} & 1+e^{i\alpha} \end{pmatrix}$	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Gate								
Spider/Wire								

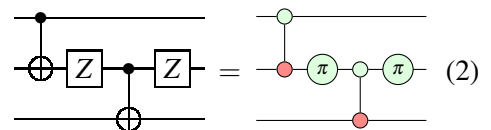
ZX-diagrams. Relations in multi-qubit systems are often difficult to understand in Dirac notation, since the matrix size doubles with every qubit and the complex number space quickly becomes confusing.

ZX-calculus provides a way to represent quantum circuits as 2-dimensional diagrams where nodes (*spiders*) and edges (*wires*) form an undirected graph. In contrast to quantum circuits, the number of input- and output wires does not have to match, hence the resulting transformations are not necessarily unitary. However, many important concepts in quantum mechanics follow very intuitively from this representation and we will briefly introduce the main principles.



2.1.1 Representing Quantum Circuits

Any transformation on a single qubit can be described as a rotation around the X and Z axes. Further, we can represent any quantum gate as a combination of X- (red) and Z-spiders (green) in ZX-diagrams (c.f. Eq. 1), of which the most important are shown in Table 1. We call the wires on the left- and rightmost the *input* and *output* of the graph, respectively.



The three generators of the universal *Clifford+T* set are constructed with the H-wire, the Z-spider with phase $\pi/4$, and a combination of an empty X- and Z-spiders (phase $\alpha = 0$) representing a CNOT. In general, we can read a ZX-diagram in any direction since only the connectivity of the spiders matters, but for comparison with common quantum circuits it is convenient to read ZX-diagrams horizontally as shown in Eq. 2.

2.1.2 Basic Rules

We introduce the most important transformation rules in the ZX language that are useful for optimization [6] in Figure 1. All ZX-rules can be applied in both directions and also apply with inverted colors.

Any two *Clifford* diagrams (i.e., diagrams only containing spiders with a *Clifford* phase $\alpha = k \cdot \frac{\pi}{2}; k \in \mathbb{Z}$) that represent the same linear map can be transformed into each other by some combination of those rules. Recent developments have introduced rule sets where this is also possible for *Clifford+T* diagrams and for all ZX-diagrams [17, 28].

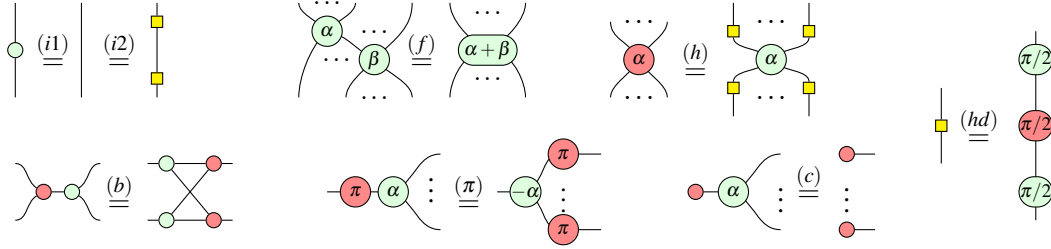


Figure 1: The important rules in ZX-Calculus that can be used for optimization are: Identity- (i1,i2), Fusion- (f), Hadamard- (h), Bialgebra- (b), Pi- (π), Copy- (c) and Hadamard-Decomposition (hd) rule. Each holds for all $\alpha, \beta \in [0, 2\pi]$. Due to (h) and (i2), all rules hold with the colours interchanged.

3 Circuit optimization with ZX-calculus

With the rules of ZX-calculus, the optimization of quantum circuits becomes a *simplification* problem on the ZX-diagram. By simplification we mean reducing the total number of either spiders or wires in a diagram in order to obtain a smaller diagram. The general process is as follows:

- 1.) Transform the circuit to a ZX-diagram
- 2.) (optional:) transform to a *graph-like* diagram, i.e.:
 - All spiders are Z-spiders. wires and are connected to at most one spider.
 - All connections are Hadamard wires. Every spider has at most one input and one output.
 - There are no loops.
 - Inputs and outputs are the only non-Hadamard
- 3.) Simplify the diagram using ZX-rules.
- 4.) Extract a quantum circuit out of the ZX-diagram.

This allows powerful optimization of circuits, which are not obvious at a first glance (we provide an intuitive example in Appendix C).

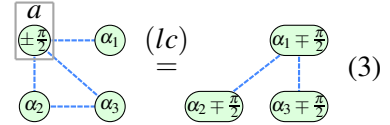
3.1 Diagram simplification

The presented ZX-rules allow many degrees of freedom, hence, simplification is still a difficult problem. The term “simplification of diagrams” has to be taken with a grain of salt since decreasing the number of spiders in a diagram can also lead to more complex extracted circuits. Since rules can be applied in both directions it is important to find *terminating* algorithms for diagram simplification. A common approach has been to only use ZX-rules which *decrease* the total number of spiders in a diagram with every application, thus ensuring termination. We present some common approaches, many of which are implemented in the PyZX-library [19].

3.1.1 Clifford spider simplification

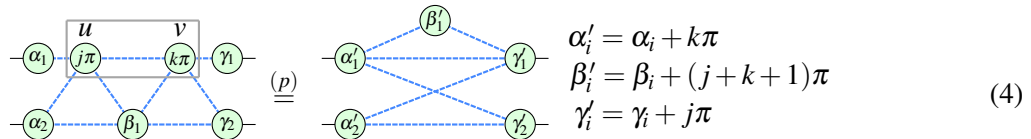
The core of most strategies are two rules from graph theory – namely *local complementation* and *pivoting* – which work on diagrams that are *graph-like*. Both rules allow the elimination of interior Clifford spiders (phase $0, \pi/2, \pi$, or $-\pi/2$; not connected to an input or output) from ZX-diagrams.

Local complementation (lc) In ZX-calculus, local complementation from Section B.1 is applicable on graph-like diagrams. If the spider a in $G \star a$ has a phase of $\pm\pi/2$, the phase is subtracted from those of the neighboring spiders and the spider is eliminated for simplification of the diagram as shown in Eq. 3.



$$(lc) \quad \begin{array}{c} a \\ \pm\frac{\pi}{2} \end{array} \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{array} = \begin{array}{c} \alpha_1 \mp \frac{\pi}{2} \\ \alpha_2 \mp \frac{\pi}{2} \\ \alpha_3 \mp \frac{\pi}{2} \end{array} \quad (3)$$

Pivoting (p) Similarly we can eliminate a pair of spiders uv with phase 0 or π by applying a graph-theoretic pivot $G \wedge uv$ (c.f. Section B.2) on the diagram as in the following example ($j, k \in \mathbb{Z}$):



$$(p) \quad \begin{array}{c} u \quad v \\ j\pi \quad k\pi \end{array} \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \gamma_1 \\ \gamma_2 \end{array} = \begin{array}{c} \beta'_1 \\ \alpha'_1 \\ \alpha'_2 \\ \gamma'_1 \\ \gamma'_2 \end{array} \quad \begin{array}{l} \alpha'_i = \alpha_i + k\pi \\ \beta'_i = \beta_i + (j+k+1)\pi \\ \gamma'_i = \gamma_i + j\pi \end{array} \quad (4)$$

3.1.2 Clifford simplification algorithm

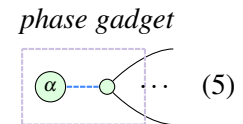
These rules allow constructing an algorithm for graph-like diagrams which removes most interior Clifford spiders [9]. The procedure is as follows:

1. Eliminate empty spiders with two wires using the identity rule and subsequently fuse the adjacent spiders in order to maintain a graph-like diagram.
2. Apply local complementation on every spider of phase $\pm\pi/2$ and pivoting on every pair of connected spiders of phase 0 or π as often as possible.
3. If step 2 modified the diagram, start again with step 1, else stop the iteration.

That allows us to remove every interior spider with phase $\pm\pi/2$ and every pair of connected spiders with phase 0 or π . However, after simplification some interior Clifford spiders with phase 0 or π may remain.

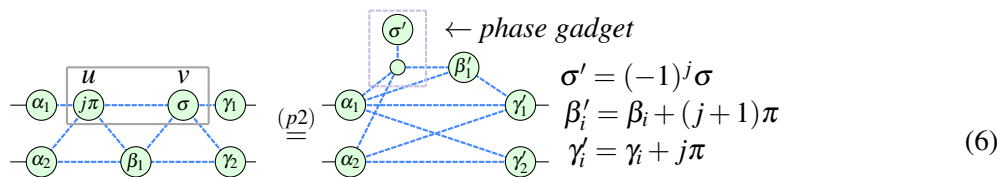
3.1.3 Phase gadget simplification (p2)

We can use phase gadgets to apply pivoting on a pair of spiders where one spider has a non-Clifford phase ($\neq 0, \pi/2, \pi, -\pi/2$). A phase gadget as defined in [20] is a parameterized spider with only one wire connected via Hadamard edge to a phaseless spider as in Eq. 5.



$$\text{phase gadget} \quad \begin{array}{c} \alpha \end{array} \begin{array}{c} \dots \end{array} \quad (5)$$

We can modify pivoting (Eq. 4) to exchange the spider with a non-Clifford phase to a phase gadget:



$$(p2) \quad \begin{array}{c} u \quad v \\ j\pi \quad \sigma \end{array} \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \gamma_1 \\ \gamma_2 \end{array} = \begin{array}{c} \sigma' \\ \beta'_1 \\ \alpha'_1 \\ \alpha'_2 \\ \gamma'_1 \\ \gamma'_2 \end{array} \quad \begin{array}{l} \sigma' = (-1)^j \sigma \\ \beta'_i = \beta_i + (j+1)\pi \\ \gamma'_i = \gamma_i + j\pi \end{array} \quad (6)$$

With the additional rules in Appendix A, we can eliminate every interior Clifford spider in a diagram [20].

3.2 Circuit extraction

Extracting a quantum circuit from a simplified diagram can be challenging, since spiders with an arbitrary number of wires have no direct gate representation [2]. The most general circuit extraction routine makes use of so called “flow properties” originating in measurement-based quantum computing

(MBQC). Graph-like ZX-diagrams can be seen as an extension of MBQC graph states where the phases of spiders represent measurements in either the XY, XZ or YZ plane of the Bloch sphere:

meas. plane	XY	XZ	YZ
meas. effect	α	$\alpha \oplus \frac{\pi}{2}$	$\alpha \oplus \frac{\pi}{2}$

(7)

Diagrams simplified with the Clifford simplification algorithm from [9] only contain spiders in the XY measurement plane and preserve a flow property called *focused generalized flow (gflow)*. Those diagrams can be extracted by converting every spider with phase α to an $R_z(\alpha)$ gate and a Hadamard wire as either a H or CZ gate or a combination of $CNOT$ gates. As an example consider the diagrams from Eq. 4. Extracting the diagram on the left hand side yields the following circuit:

(8)

whereas extracting the diagram after rule application yields the equivalent smaller circuit:

(9)

However, ZX-diagrams simplified with Eq. 6 may contain spiders in XZ and YZ plane as well. While it has been shown that those diagrams still preserve *generalized flow (gflow)*, the circuit extraction routine has to convert those spiders back into XY spiders using either pivoting (YZ) or a combination of local complementation and pivoting (XZ) before extracting the diagram [2]. An algorithm to efficiently extract diagrams that do not admit the gflow property is yet to be discovered; however, recent findings suggest that such an algorithm may not exist for general ZX-diagrams [4]. Hence, even though the diagram may represent a unitary matrix, we cannot extract a quantum circuit from the diagram efficiently.

4 Enhancing reduction of 2-qubit gates

As seen in Eq. 8 and 9, a Hadamard wire gets extracted to H, CNOT or CZ gates. The number of Hadamard wires in a graph-like ZX-diagram therefore correlates with the number of 2-qubit gates in the extracted circuit. The diagram simplification algorithms shown in Section 3 focus on eliminating spiders while neglecting – or even increasing – the number of Hadamard wires. Hence, this section introduces methods which additionally minimize the amount of Hadamard wires (ref. as `#wires` in the following).

To do so, it is crucial to examine *where* and *when* local complementation and pivoting are applied. Both rules can either increase or decrease `#wires`, depending on the connectivity of the relevant neighbors. Eq. 3 and 4 show examples in which `#wires` decreases. However, as Eq. 6 shows it can also increase `#wires` and we easily construct extreme cases like the one shown in Eq. 10. Applying local complementation to the central spider with phase $\pi/2$ yields a diagram containing one spider less but a significantly higher `#wires`. Extracting the left diagram with the current version of the PyZX-library produces a circuit with six 2-qubit gates, while the diagram on the right gets extracted as a circuit with 21 2-qubit gates. Generally, applying local complementation on a spider with

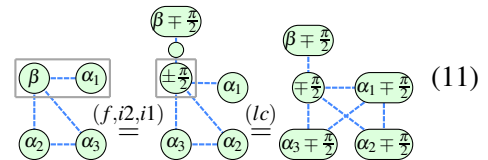
(10)

n unconnected neighbors leads to $\frac{n(n+1)}{2} - n$ new wires. As pivoting involves local complementation on two spiders, the effect usually even worsens for this rule.

To prevent such cases and to guide the simplification process towards a minimal $\#wires$, we introduce cost functions for local complementation and pivoting allowing us to calculate $\#wires$ after rule applications. We take those as a heuristic for estimating how rule applications change the number of 2-qubit gates and implement decision strategies for diagram simplification based on the heuristics.

4.1 Pivoting and local complementation on spiders with arbitrary phases

In contrast to the Clifford simplification algorithm from Section 3, we can apply local complementation and pivoting on spiders with *arbitrary* phases. Similar to the *Pivot Phase Gadget* (p2) rule, we can change a spider with non-Clifford phase by a combination of the rules $(f, i2, i1)$ as in



Eq. 11. With that we can apply local complementation on spiders with phase different from $\pm\pi/2$ (this introduces one XZ -spider) and pivoting on pairs of spiders where one/no spider has a phase of 0 or π (this introduces one/two YZ -spiders). Note that this does not change the gflow property (c.f. [2, Lemma 3.1]).

4.2 Local Complementation Heuristic (LCH)

The costs for local complementation are calculated on the following proposition:

Proposition 4.1 *Let $G = (V, E)$ be an open graph; $u \in V$ an arbitrary vertex with neighbors $N(u) \subset V$; $n = |N(u)|$ the number of neighbors; and m the number of edges between the neighbors, i.e.,*

$$m = |\{(a, b) \in E | a, b \in N(u)\}|.$$

For $G \star u$, n remains the same, but m changes to $m' = \Delta_{n-1} - m$, where $\Delta_{n-1} = \frac{n(n-1)}{2}$.

Hence, the difference in the number of wires after application of the local complementation rule is:

$$(n + m) - (n + (\Delta_{n-1} - m)) = 2m - \Delta_{n-1} \tag{12}$$

With respect to the phase $\varphi(u)$ of the spider u , the graph changes as follows:

- If $\varphi(u) = \pm\pi/2$: Remove u from the graph and eliminate all wires between u and $N(u)$.
- If $\varphi(u)$ is non-Clifford: All wires between u and $N(u)$ remain and we get an additional wire for the *phase gadget*.
- If $\varphi(u)$ is 0 or π : No phase gadget is needed and we can use the π -copy rule.

The *LCH* is calculated as follows:

$$LCH(u) = \begin{cases} 2m - \Delta_{n-1} + n & \text{if } \varphi(u) = \pm\frac{\pi}{2} \\ 2m - \Delta_{n-1} & \text{if } \varphi(u) = k \cdot \pi, k \in \mathbb{Z} \\ 2m - \Delta_{n-1} - 1 & \text{otherwise} \end{cases} \tag{13}$$

4.3 Pivoting Heuristic (PH)

We calculate the upper bound of new connections with the sets A, B, C (see Appendix B):

$$C_{max} = |A| \cdot |B| + |A| \cdot |C| + |B| \cdot |C| \quad (14)$$

We denote the number of neighbors of u and v by $n_u = |N(u)|$ and $n_v = |N(v)|$, respectively, and the number of edges between neighbors of different sets as m .

The changes of the graph G to $G \wedge uv$ have the following cases ($j, k \in \mathbb{Z}$):

- (C1) $\varphi(u) = j \cdot \pi, \varphi(v) = k \cdot \pi$: If both spiders have a phase of 0 or π , all connections between $\{u, v\}$ and $N(u) \cup N(v)$ are eliminated.
- (C2) $\varphi(u) = j \cdot \pi, \varphi(v) \neq k \cdot \pi$: If v becomes a phase gadget and u gets eliminated, all neighbors of u get connected to v and we have an additional wire for the phase gadget.
- (C3) $\varphi(u) \neq j \cdot \pi, \varphi(v) = k \cdot \pi$: If u becomes a phase gadget and v gets eliminated, all neighbors of v get connected to u and we have an additional wire for the phase gadget.
- (C4) $\varphi(u) \neq j \cdot \pi, \varphi(v) \neq k \cdot \pi$: If both spiders become phase gadgets, all neighbors of u get connected to v and all neighbors of v get connected to u . Furthermore, u gets connected to v again and we have two more wires for the phase gadgets.

With these conditions, the PH is calculated as follows:

$$PH(u, v) = \begin{cases} 2m - C_{max} + n_u + n_v - 1 & \text{for (C1)} \\ 2m - C_{max} + n_v - 1 & \text{for (C2)} \\ 2m - C_{max} + n_u - 1 & \text{for (C3)} \\ 2m - C_{max} - 2 & \text{for (C4)} \end{cases} \quad (15)$$

4.4 Decision strategies

With the two heuristics (LCH, PH) at hand we can now implement different strategies to decide where and when local complementation or pivoting are applied during the simplification. A single simplification step in our procedure consists of the following actions:

1. Filter all possible rule applications of the current ZX-diagram.
2. Select rule according to selection strategy (see below).
3. Apply rule on the ZX-diagram.

For filtering rule applications we can specify a lower bound for the heuristic, e.g., LCH or $PH = -5$ says that we do not consider rule applications which increase $\#wires$ by more than five. We can also specify whether rule applications are allowed on boundary spiders (c.f. Appendix A) and whether rules are allowed on arbitrary phased spiders. For selecting a rule we implemented two different strategies:

- **Random selection:** Rules are chosen by a random coin flip.
- **Greedy selection:** Chooses the rule application which maximally decreases $\#wires$.

Each algorithm terminates if we allow only rule applications with a $LCH/PH > 0$ and when there is no rule left that decreases $\#wires$. They also terminate if we allow negative gains ($LCH/PH \leq 0$) and restrict the matches to interior spiders that do not generate new spiders. This is the case in standard local complementation on a spider with phase $\pm\pi/2$ and pivoting on a pair of spiders with phase 0 or π . The

algorithm eliminates at least one spider in every step and terminates when neither interior spiders with phase $\pm\pi/2$ nor pairs with phase 0 or π are left.

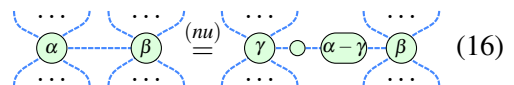
On the other hand, allowing rule applications on spiders of arbitrary phases which increase `#wires` may result in loops and therefore no termination. For such cases we only allow rule applications which increase `#wires` on spiders present since the very beginning of our simplification procedure. On newly generated spiders only rules which decrease `#wires` are allowed.

5 New optimization rule: Neighbor Unfusion

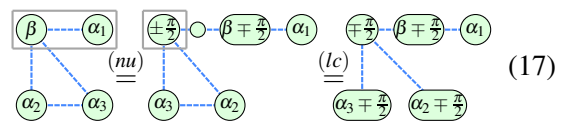
As shown in Eq. 6, the application of local complementation (lc) and pivoting (p) on spiders with many neighbors can not only decrease but also increase `#wires`. The heuristics shown in Section 4 may help to identify and prevent extreme cases as in Eq. 10. However, spiders that are measured in YZ - or XZ -plane (c.f. Section 3.2) require special attention: When extracting a spider in YZ -plane (e.g., the empty spider of the phase gadget in Eq. 5 and 6), pivoting has to be applied to maintain the focused gflow property. The same happens for spiders in XZ -plane, but they are resolved by local complementation [2]. This affects `#wires` after the simplification and a simplified diagram containing some spiders in YZ - and XZ -plane may result in an expensive circuit.

However, when applying either rule on diagrams with arbitrary spiders as discussed in Section 4.1), spiders in YZ - or XZ -plane are generated. We introduce the *neighbor unfusion* (nu) rule, which allows lc and p on such arbitrary-phase spiders without introducing spiders in YZ - or XZ -plane.

Neighbor unfusion combines the fusion (f) and identity rules ($i1, i2$) as shown in Eq. 16. If a spider with phase α is connected to a neighbor, we change its phase to an arbitrary phase γ by inserting an empty spider and a spider with phase $\alpha - \gamma$ between the spider and its neighbor. It allows changing the phase of an arbitrary spider to $\gamma = \pm\frac{\pi}{2}$ and thus local complementation and removal of spiders with arbitrary phases.



For illustration, we apply neighbor unfusion (nu) to the example of Eq. 11. We can move the β spider to any direction (in Eq. 17 towards α_1), so it is then not affected by the application of local complementation (lc).



Comparing Eq. 11 and Eq. 17 we see that we not only reduce `#wires`, but also prevent the generation of a spider in XZ -plane. However, the neighbor unfusion rule sometimes leads to diagrams which do not have focused gflow property. This is due to the insertion of the empty spider and the spider with phase $\alpha - \gamma$ in 16. We observed that this problem does not occur if the spiders with phase α and β get extracted to the same qubit. Currently, we find such pairs of spiders by using the flow hierarchy of the *maximally delayed gflow* of the diagram (c.f. [2]), which is quite costly, because we need to recalculate the gflow at each simplification step. Therefore, diagram simplification with neighbor unfusion has a much higher runtime than the other simplification procedures. It is an open question whether neighbour unfusion can only destroy the focused gflow property, or also more general flow properties like gflow or Pauli flow.

6 Evaluation

We evaluate our heuristic-based simplification algorithms on a set of circuits first used in [1]. They implement various arithmetic problems as quantum circuits and were used as a benchmark set for comparing different optimization strategies [20, 22]. We use it to compare our heuristic-based approaches

Table 2: Circuit metrics for original benchmark circuits, Post-optimization metrics of the standard Clifford simplification [9], PyZX [20], Nam et al. [22], our heuristic-based simplification method, and the combined approach of Nam et al. and our heuristic-based methods. Only our best optimization is presented: 1) Greedy, 2) Random, 3) Greedy with neighbor unfusion, 4) Random with neighbor unfusion, the lower bound for the heuristics is denoted in brackets. If the best PyZX result is achieved by the $t|ket\rangle$ -library, the respective cell is marked with \star . The best results of each metric in each row are marked.

Circuit	Original		Clifford algorithm		PyZX/ $t ket\rangle^*$		Nam et al.		Heuristic algorithm			Nam+Heuristic		
	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Alg.	Σ	$2Q$	Alg.
Mod 5 ₄	63	28	36	21	24*	12*	51	28	41	23	2(-20)	38	23	3(1)
VBE-Adder ₃	150	70	116	59	101	54	89	50	87	42	3(1)	87	42	4(1)
CSLA-MUX ₃	170	80	177	97	156	75	155	70	155	74	3(-5)	156	67	3(1)
CSUM-MUX ₃	420	168	455	271	327*	158*	266	140	303	150	3(1)	266	140	1(1)
QCLA-Com ₇	443	186	397	223	316	148	284	132	295	138	4(-5)	275	132	1(1)
QCLA-Mod ₇	884	382	903	475	717	324	-	-	705	311	4(-20)	-	-	-
QCLA-Adder ₁₀	521	233	562	305	435	199	399	183	417	193	4(-20)	398	182	4(1)
Adder ₈	900	409	779	429	675	339	606	291	597	295	4(1)	514	256	4(1)
RC-Adder ₆	200	93	206	113	393*	164*	140	71	159	71	1(1)	152	71	1(1)
Mod-Red ₂₁	278	105	260	130	217	93	180	77	196	85	3(1)	179	76	1(1)
Mod-Mult ₅₅	119	48	124	74	91	42	91	40	90	40	1(1)	90	41	1(1)
Toff-Barenc ₃	58	24	50	26	59*	18*	40	18	46	21	1(1)	40	18	3(-5)
Toff-NC ₃	45	18	41	20	40	16	35	14	36	15	3(1)	35	14	1(1)
Toff-Barenc ₄	114	48	117	60	95	44	72	34	88	40	4(1)	72	34	3(1)
Toff-NC ₄	75	30	86	43	65	26	55	22	57	24	3(1)	55	22	1(1)
Toff-Barenc ₅	170	72	149	86	140	66	104	50	122	57	4(1)	102	48	3(1)
Toff-NC ₅	105	42	92	42	90	36	75	30	78	33	3(1)	75	30	1(1)
Toff-Barenc ₁₀	450	192	392	196	365	176	264	130	325	151	4(1)	252	118	3(1)
Toff-NC ₁₀	255	102	237	100	215	86	175	70	183	78	3(1)	175	70	1(1)
GF(2 ⁴)-Mult	225	99	245	140	193	99	187	99	195	101	2(1)	180	98	3(-5)
GF(2 ⁵)-Mult	347	154	351	197	304	154	296	154	306	156	1(1)	289	155	4(-20)
GF(2 ⁶)-Mult	495	221	545	308	422	221	403	221	418	217	4(-5)	390	218	3(-5)
GF(2 ⁷)-Mult	669	300	736	417	573	300	555	300	572	299	4(-5)	535	292	4(-20)
GF(2 ⁸)-Mult	883	405	1015	606	745	405	712	405	745	405	1(1)	691	399	1(1)
Avg. reduction			~ 3%	~ -22%	~ 14%	~ 9%	~ 27%	~ 19%	~ 23%	~ 16%		~ 29%	~ 21%	

with the Clifford simplification algorithm described in [9], and some of the best results reported for circuit optimizations with [20] and without using ZX-calculus [22]. We also investigate how ZX-calculus based approaches perform when using the TODD-algorithm [14] for additional T gate reduction.

6.1 Implementation

With the exception of [20] – which omits simplifying the diagram (step 3) and extraction (step 4) – we use the following pipeline for ZX-calculus based optimization algorithms:

1. Optimize circuit using gate cancellation and commutation.
2. Transform circuit to ZX-diagram and apply phase teleportation to reduce T-count (as in [20]).
3. Simplify ZX-diagram (standard Clifford or heuristic-based simplification).
4. Extract circuit from ZX-diagram.
5. Optimize circuit as in step 1.

Since our heuristic-based algorithms do not reduce T gates, we always apply the phase teleportation in step 2 since this reduces T gates as far as currently possible with ZX-calculus. This ensures comparable results regarding the 2-qubit gate count against non-ZX-calculus based approaches that optimize all types of gates. We implemented our algorithms in a clone of the PyZX library which also contains our optimized circuits in the OpenQASM format¹. All results were proven to be correct by checking whether the optimized circuit together with the adjoint of the original circuit can be reduced to the identity.

6.2 Results

For each circuit we compare the total gate count Σ and the 2-qubit gate count $2Q$. The results are summarized in Table 2 and 3: Their columns show circuit name, metrics of the original circuit of the benchmark, metrics of one (or more) existing optimization algorithms and the metrics of the best performing heuristic-based algorithm. For the latter, we denote the simplification strategy achieving the best result in the last column: 1. Greedy, 2. Random, 3. Greedy with neighbor unfusion, 4. Random with neighbor unfusion.

As a very first result, the last column in the “Heuristic Algorithm” section of Table 2 prominently indicates the great value of neighbor unfusion (Alg. 3 and 4), as it achieves the best performance of our heuristics in $> 70\%$ of the cases.

We now compare our heuristic-based simplifications following against other ZX-calculus based optimizations in Table 2. For most circuits our heuristic-based simplification clearly outperforms the standard Clifford simplification [9], both in total and 2-qubit gate reduction. Moreover, while our approaches almost always decrease circuit metrics, the standard approach often yields circuits with higher metrics than the original circuit (e.g., “CSLA-MUX₃”, “GF(2⁶)-Mult”). Especially for 2-qubit gates our approaches *decrease* 2-qubit gate count by 16%, while the standard approach even *increases* the count by 22%. In a direct comparison our approaches have up to 33% (“Toff-NC₄”) fewer total and 47% (“Mod-Mult₅₅”) fewer 2-qubit gates than the standard Clifford approach.

Second, we compare against the best available PyZX implementation [20] and the recommended optimization pipeline of the t|ket)-library [27] with the routines PauliSimp and FullPeepholeOptimize, which use similar strategies. The column “PyZX/t|ket)” in Table 2 shows the best optimization results for both implementations and \star indicates results from t|ket). Except for two circuits (“Mod 5₄” and “Toff-Barenco₃”), our algorithms outperform all ZX-calculus based algorithms in terms of total gate count and 2-qubit gate count.

Third, Table 2 also shows our result in comparison to the cutting-edge non-ZX-calculus based algorithm from Nam et al. [22]. It can be seen that the algorithm from [22] outperforms any ZX-calculus based algorithm for most circuits. Still, we were able to achieve better results for the circuits “VBE-Adder₃” and “Mod-Mult₅₅”. Note that we did not compare for the “QCLA-Mod₇” circuit, because [15] reports that the optimized circuit from [22] does not correspond to the original.

Last, the rightmost columns of Table 2 show a combination of Nam et al’s approach with ours. We

Table 3: Circuit metrics for the original benchmark circuits, Post-optimization metrics for PyZX+TODD and of our heuristic-based algorithms+TODD.

Circuit	Original			PyZX+TODD			Heuristic+TODD			Alg.
	Σ	$2Q$	\mathcal{F}	Σ	$2Q$	\mathcal{F}	Σ	$2Q$	\mathcal{F}	
CSLA-MUX ₃	170	80	70	262	175	43	257	169	43	2(1)
CSUM-MUX ₃	420	168	196	575	428	74	411	261	74	4(-5)
QCLA-Com ₇	443	186	203	454	274	93	389	211	93	4(1)
QCLA-Adder ₁₀	521	233	238	800	517	143	677	391	143	4(-20)
Mod-Mult ₅₅	119	48	49	107	56	27	104	55	27	4(-5)
GF(2 ⁴)-Mult	225	99	112	298	221	52	295	220	52	1(-5)
GF(2 ⁵)-Mult	347	154	175	538	420	88	524	403	88	3(1)
GF(2 ⁶)-Mult	495	221	252	943	764	134	933	750	134	3(1)
GF(2 ⁷)-Mult	669	300	343	1253	1036	180	1223	993	180	4(1)
GF(2 ⁸)-Mult	883	405	448	1791	1521	224	1780	1507	224	3(1)

¹<https://github.com/mnm-team/pyzx-heuristics>

use the output circuits from the Nam et al. optimization as input for our algorithms and observe that we achieve equally good or better results for almost all circuits. The larger the circuit, the more significantly this combination improves the previous best known results. Most notably, we improve the total count of the “Adders” circuit by more than 15% and the 2-qubit gate count by more than 12%.

Apart from the 2-qubit gate count, the T gate count of a quantum circuit is an important metric, since T-gates are more complex to implement for an error-corrected quantum computer. Therefore, we compare our algorithms to the other ZX-calculus based approaches using the TODD algorithm as optimization step 1). It is designed to reduce T gate count by introducing ancilla qubits, but sometimes also reduces T-gates in the ancilla-free case. Table 3 shows those benchmarks circuits where the combination of TODD and a ZX-calculus based algorithm reduces T gate count even more compared to the best result in Table 2. We compare the best combination of our heuristic-based algorithm and TODD against the best combination of an existing ZX-calculus based algorithm and TODD.

While we observe a general increase in 2-qubit and total gates using TODD algorithm, our best algorithm yields better results than the existing ZX-calculus based algorithms in every case.

7 Conclusions and Future Work

In this work we introduce two functions, namely the Local Complementation Heuristic LCH (for the local complementation rule) and the Pivot Heuristic PH (for the pivot rule). The functions calculate the number of Hadamard wires that would be added or removed by applying the respective rule, thus serving as a heuristic for estimating the 2-qubit gate count of the underlying circuit. This allows us to develop a more sophisticated strategy for ZX-diagram simplification: First, dismiss the applicable rules that cost too much and then either select a rule randomly or select the rule with the best wire count decrease.

Notably, the T gate count remains unchanged throughout this process, which is why our approach and others that mainly decrease the T gate count complement each other well. Further, we introduce the new *Neighbor Unfusion* rule which combines the established *fusion* and *identity* rules. This rule allows introducing spiders with arbitrary phases into the circuit if needed, for example when the local complementation or pivot rule would be useful to reduce Hadamard wires. As a side note, we also formally describe how to use the local complementation and the pivoting rule on spiders with non-Clifford phases, which is a common implementation practice but has never been mentioned in theory.

We measure the impact of aligning the optimization strategy with the heuristics and adding the neighbor unfusion rule by comparing our algorithm to four other approaches, some based on ZX-calculus and some not, on a set of 24 well-established benchmark circuits. Our approaches show significant improvements compared to all other ZX-based approaches, especially in 2-qubit gate reduction. On their own, non-ZX-based approaches still yield slightly better results than our ZX-based approaches. However, when combining both we are able to optimize circuits better than the previously best known result, which seems to be a promising field for further research.

Using heuristics for ZX-diagram simplification also provides many possibilities for future improvement. Regarding the selection of rules, both random and greedy strategy are non-optimal for finding a ZX-diagram with minimal number of wires. Instead, we propose using a metaheuristic selection strategy like simulated annealing for escaping local minima during simplification. Furthermore, since simplification with neighbor unfusion tends to yield the best results, we think it is important to further investigate in which cases neighbor unfusion generates XY spiders and if we can preserve valid ZX-diagrams when allowing unfusion on spiders which get extracted on different qubits.

Acknowledgment

This work is partially supported by the German Federal Ministry of Education and Research (BMBF) under the funding programme Quantum Technologies - From Basic Research to Market under contract number 13N16077.

References

- [1] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T-Depth Optimization of Clifford+T Circuits Via Matroid Partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953.
- [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.
- [3] Stefanie Barz (2015): *Quantum computing with photons: introduction to the circuit model, the one-way quantum computer, and the fundamental principles of photonic experiments*. *Journal of Physics B: Atomic, Molecular and Optical Physics* 48(8), p. 083001, doi:10.1088/0953-4075/48/8/083001.
- [4] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-Diagrams Can Be #P-Hard*. doi:10.4230/LIPICS.ICALP.2022.119.
- [5] André Bouchet (1988): *Transforming trees by successive local complementations*. *Journal of Graph Theory* 12(2), pp. 195–207, doi:10.1002/jgt.3190120210.
- [6] 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.
- [7] Bob Coecke, Dominic Horsman, Aleks Kissinger & Quanlong Wang (2022): *Kindergarden quantum mechanics graduates ...or how I learned to stop gluing LEGO together and love the ZX-calculus*. *Theoretical Computer Science* 897, pp. 1–22, doi:10.1016/j.tcs.2021.07.024.
- [8] Bob Coecke & Aleks Kissinger (2018): *Picturing quantum processes*. In: *International Conference on Theory and Application of Diagrams*, Springer, pp. 28–31, doi:10.1007/978-3-319-91376-6_6.
- [9] 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.
- [10] Edward Farhi, Jeffrey Goldstone & Sam Gutmann: *A Quantum Approximate Optimization Algorithm*, doi:10.48550/arXiv.1411.4028.
- [11] Jim Geelen & Sang-il Oum (2009): *Circle graph obstructions under pivoting*. *Journal of Graph Theory* 61(1), pp. 1–11, doi:10.1002/jgt.20363.
- [12] Lov K. Grover (1996): *A fast quantum mechanical algorithm for database search*. In Gary L. Miller, editor: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, New York, NY, pp. 212–219, doi:10.1145/237814.237866.
- [13] Aram W. Harrow, Avinandan Hassidim & Seth Lloyd (2009): *Quantum algorithm for solving linear systems of equations*. *Physical Review Letters* 103(15), p. 1474, doi:10.1103/PhysRevLett.103.150502.
- [14] Luke E Heyfron & Earl T Campbell (2018): *An efficient quantum compiler that reduces T count*. *Quantum Science and Technology* 4(1), p. 015004, doi:10.1088/2058-9565/aad604.
- [15] Keshav Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu & Michael Hicks (2021): *A Verified Optimizer for Quantum Circuits*. *Proc. ACM Program. Lang.* 5(POPL), doi:10.1145/3434318.
- [16] Dominik Janzing, Pawel Wocjan & Thomas Beth (2005): *“Non-Identity-Check” is QMA-complete*. *International Journal of Quantum Information* 03(03), pp. 463–473, doi:10.1142/S0219749905001067.
- [17] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic*

- in *Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, p. 559–568, doi:10.1145/3209108.3209131.
- [18] Navin Khaneja & Steffen J. Glaser (2001): *Cartan decomposition of $SU(2n)$ and control of spin systems*. *Chemical Physics* 267(1-3), pp. 11–23, doi:10.1016/S0301-0104(01)00318-4.
- [19] 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.
- [20] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Phys. Rev. A* 102, p. 022406, doi:10.1103/PhysRevA.102.022406.
- [21] Daniel Litinski (2019): *Magic State Distillation: Not as Costly as You Think*. *Quantum* 3, p. 205, doi:10.22331/q-2019-12-02-205.
- [22] Yunseong Nam, Neil J Ross, Yuan Su, Andrew M Childs & Dmitri Maslov (2018): *Automated optimization of large quantum circuits with continuous parameters*. *npj Quantum Information* 4(1), pp. 1–12, doi:10.1038/s41534-018-0072-4.
- [23] Michael A. Nielsen & Isaac L. Chuang (2013): *Quantum computation and quantum information*, first south asia edition edition. Cambridge University Press, Cambridge, doi:10.1017/CBO9780511976667.
- [24] Joe O’Gorman & Earl T. Campbell (2017): *Quantum computation with realistic magic-state factories*. *Physical Review A* 95(3), doi:10.1103/PhysRevA.95.032338.
- [25] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik & Jeremy L. O’Brien (2014): *A variational eigenvalue solver on a photonic quantum processor*. *Nature Communications* 5(1), p. 4213, doi:10.1038/ncomms5213.
- [26] P. W. Shor (1994): *Algorithms for quantum computation: Discrete logarithms and factoring*. In Shafi Goldwasser, editor: *Foundations of Computer Science, 35th Symposium on (FOCS '94)*, IEEE Computer Society Press, pp. 124–134, doi:10.1109/SFCS.1994.365700.
- [27] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *t|ket>: a retargetable compiler for NISQ devices*. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92.
- [28] 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.

A Further rules

In addition to the rules in Section 3, additional rules have been developed to eliminate *every* interior Clifford spider.

A.1 Pivoting Boundary Spiders (p1)

The pivoting rule can also be applied if one of the spiders is a boundary spider, i.e., connected to an input or output, using the following transformation:

$$\begin{array}{c} u \\ j\pi \end{array} \text{---} \begin{array}{c} v \\ k\pi \end{array} \text{---} \begin{array}{c} (i1, i2) \\ \text{---} \end{array} \begin{array}{c} u \\ j\pi \end{array} \text{---} \begin{array}{c} v \\ k\pi \end{array} \text{---} \begin{array}{c} (p) \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (18)$$

Here v gets transformed to an interior spider and both u and v can be removed using the pivoting rule.

A.2 Gadget Fusion (gf):

An important feature of phase gadgets is that we can fuse two phase gadgets connected to the same neighbors by summing up their phases.

$$\begin{array}{c} \alpha \\ \beta \end{array} \text{---} \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_n \end{array} \xrightarrow{(gf)} \begin{array}{c} \alpha + \beta \\ \text{---} \end{array} \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_n \end{array} \quad (19)$$

This rule is used for eliminating non-Clifford spiders in a diagram, for instance, two phase gadgets with phase $\pi/4$ connected to the same set of neighbours can be fused into a single phase gadget with phase $\pi/2$. Combining the Clifford simplification algorithm with those extended rules we can eliminate *all* interior Clifford spiders (in exchange for phase gadgets) and *some* interior non-Clifford spiders.

B Graph Theory

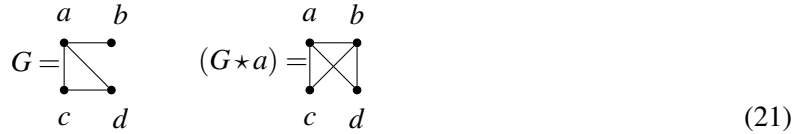
Since ZX-calculus and its optimization strategies rely on graph operations on undirected graphs, we provide some background on it: An *undirected graph* is a tuple $G = (V, E)$ with *vertices* (or “nodes”) V and *edges* $E \subseteq V \times V$.

B.1 Local Complementation

The local complementation \star [5] of an undirected graph $G = (V, E)$ about a vertex u is defined as follows (Δ is the symmetric set difference: $A \Delta B := (A \cup B) \setminus (A \cap B)$):

$$G \star u := (V, E \Delta \{(a, b) \mid (a, u), (b, u) \in E, a \neq b\}) \quad (20)$$

The following example shows a graph G and its local complementation about a . Intuitively, local complementation connects two neighbours of a if they are *not* connected (e.g., b, c) and disconnects them otherwise (e.g., c, d).



B.2 Pivoting [11]

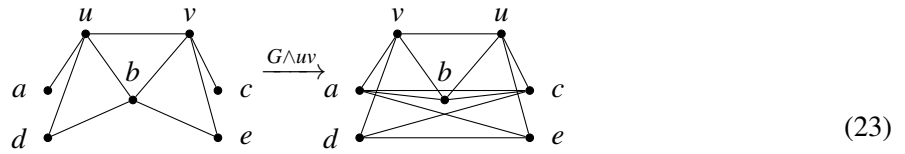
Pivoting \wedge rewrites an edge $(u, v) \in E$ by triple local complementation:

$$G \wedge uv := ((G \star u) \star v) \star u \quad (22)$$

To derive the new graph, we consider three disjoint sets (where the neighborhood of vertex x is defined as $N(x) = \{y \in V \mid (x, y) \in E\}$):

- $A := N(u) \cap N(v)$: Vertices connected to u and v .
- $B := N(u) \setminus N(v)$: Vertices connected to u and not to v .
- $C := N(v) \setminus N(u)$: Vertices connected to v and not to u .

In a pivoted graph $G \wedge uv$, two vertices from different sets A, B or C are connected if, and only if, the two are not connected in G . Connections between vertices of the same set are not modified. As an example, consider the following graphs G (left) and $G \wedge uv$ (right), where $A = \{b\}$, $B = \{a, d\}$, $C = \{c, e\}$. Intuitively, pivoting connects all vertices between A, B, C that are not connected in G (e.g., a, b) and disconnects them otherwise (e.g., b, d):



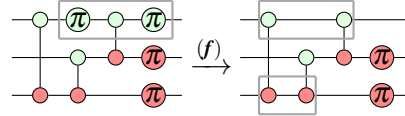
C Example for ZX optimization

The following circuit can be optimized as follows:

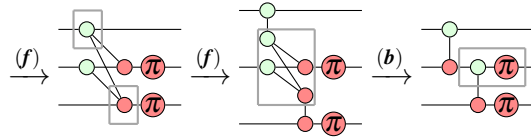
(24)

We use the following rules² (the affected spiders/wires to which a rule is applied are framed):

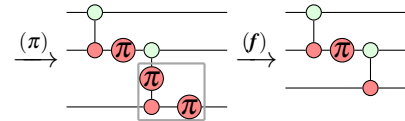
- 1) Eliminate the two Z-gates using spider fusion (f):



- 2) Reduce from 3 to 2 CNOTs with fusion (f) and bialgebra rule (b):



- 3) Eliminate one X by the π copy rule:



²The example is inspired by a talk of Russ Duncan “Quantum Formal Methods” from 2021 (1h 35min) which is publicly available.

Building Qutrit Diagonal Gates from Phase Gadgets

John van de Wetering

University of Oxford
john@vdwetering.name

Lia Yeh

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

Phase gadgets have proved to be an indispensable tool for reasoning about ZX-diagrams, being used in optimisation and simulation of quantum circuits and the theory of measurement-based quantum computation. In this paper we study phase gadgets for qutrits. We present the *flexsymmetric* variant of the original qutrit ZX-calculus, which allows for rewriting that is closer in spirit to the original (qubit) ZX-calculus. In this calculus phase gadgets look as you would expect, but there are non-trivial differences in their properties. We devise new qutrit-specific tricks to extend the graphical Fourier theory of qubits, resulting in a translation between the ‘additive’ phase gadgets and a ‘multiplicative’ counterpart we dub *phase multipliers*.

This enables us to generalise the qubit notion of multiple-control to qutrits in two ways. The first type is controlling on a single tritstring, while the second type applies the gate a number of times equal to the tritwise multiplication modulo 3 of the control qutrits. We show how both types of control can be implemented for any qutrit Z or X phase gate, ancilla-free, and using only Clifford and phase gates. The first requires a polynomial number of gates and exponentially small phases, while the second requires an exponential number of gates, but constant sized phases. This is interesting, because such a construction is not possible in the qubit setting.

As an application of these results we find a construction for *emulating* arbitrary qubit diagonal unitaries, and specifically find an ancilla-free emulation for the qubit CCZ gate that only requires three single-qutrit non-Clifford gates — provably lower than the four T gates needed for qubits with ancilla.

1 Introduction

Most quantum computing theory developed thus far has focussed on qubits — two-level quantum systems. However, there has been a recent surge of interest in studying the more general case of d -level quantum systems, called *qudits*. This has led to applications of qudits for quantum algorithms [52], improving magic state distillation noise thresholds [12], and communication noise resilience [20]. Qudits have been experimentally demonstrated on quantum processors based on ion traps [46] and superconducting devices [8, 55, 57, 34].

The specific case of *qutrits*, where $d = 3$, has been used to improve qubit readout [40], but most notably, qutrits have been used to study *emulation*: where qubit computation is emulated inside a subspace of the qudits to enable more resource-efficient gate implementations. In contrast, it has been argued that qubits cannot simulate qudit (where $d > 2$ and d is not a power of 2) computation efficiently [11].

Much work on qutrits and emulation has focussed on *classical* functions: those that come from a map of classical trits. For instance, using qutrits we can build logarithmic-depth Toffolis [27, 41] and binary AND gates on superconducting qutrits [16]. This leaves open the question of whether there are any advantages to emulation by studying ‘truly’ quantum gates such as diagonal unitaries. For qubits a useful tool for understanding diagonal unitaries has been the concept of a *phase gadget* [38]. This is a type of symmetric multi-qubit interaction that occurs naturally in many hardware architectures [43, 42, 47], and serves as a good basis for optimising quantum circuits [18, 19, 7, 6, 54, 5]. Any diagonal qubit unitary can be expressed as a product of phase gadgets by writing the unitary as a *phase polynomial* [1, 31].

In this paper we study the generalisation of phase gadgets to the qutrit setting. We do this by adapting the qutrit ZX-calculus of Refs. [28, 50] and transforming it into a *flexsymmetric* calculus [15] where the spiders have more desirable symmetry properties. We find this calculus has a simple set of rules for the Clifford fragment. We define phase gadgets analogously to the qubit case, meaning that as diagrams they look nearly identical. There are however significant differences between the qubit and qutrit gadgets. We will show that we can nevertheless use qutrit phase gadgets to construct some useful qutrit diagonal unitaries, such as controlled phase gates, and a type of gate we dub a *phase multiplier*. This last one is possible by generalising the formula that leads to the *graphical Fourier theory* for qubit diagonal unitaries [39].

As an application of our results we show how we can emulate an arbitrary qubit diagonal unitary using qutrit phase gadgets. This leads us to a construction of the emulated qubit CCZ gate that requires only three non-Clifford qutrit *R gates* [26]. This is surprising because using just qubits, we would require at least four *T gates* to implement the CCZ [35].

We start the paper by reviewing the basics of qutrit quantum computation in Section 2. Then we introduce the flexsymmetric qutrit ZX-calculus in Section 3. Diagonal qutrit unitaries, phase gadgets, controlled phase gates, and phase multipliers are studied in Section 4. We show how to use these to emulate diagonal qubit unitaries in Section 5 and end with some discussion on future work in Section 6.

2 Qutrit quantum computation

A qubit is a two-dimensional Hilbert space. Similarly, a qutrit is a three-dimensional Hilbert space. We will write $|0\rangle$, $|1\rangle$, and $|2\rangle$ for the standard computational basis states of a qutrit. Any normalised qutrit state can then be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$ where $\alpha, \beta, \gamma \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.

Several concepts for qubits extend to qutrits, or more generally to *qudits*, which are d -dimensional quantum systems. In particular, the concept of Pauli's and Cliffords. For a d -dimensional qudit, we define the respective Pauli X and Z gates as

$$X|k\rangle = |k+1\rangle \quad Z|k\rangle = \omega^k|k\rangle \quad (1)$$

where $\omega := e^{2\pi i/d}$ is such that $\omega^d = 1$, and the addition $|k+1\rangle$ is taken modulo d [30, 36]. Note that for qubits this X gate is just the NOT gate, while $Z = \text{diag}(1, -1)$. We call unitaries generated by products and tensor products of the X and Z gate *Pauli gates*. In this paper we will work solely with qutrits, so we take ω to always be equal to $e^{2\pi i/3}$. Note that $\omega^{-1} = \omega^2 = \bar{\omega}$ where \bar{z} denotes the complex conjugate of z .

For a qubit there is only one non-trivial permutation of the standard basis states, implemented by the X gate. For qutrits there are five non-trivial permutations of the basis states. By analogy we will call them all ternary X gates. These gates are X_{+1} , X_{-1} , X_{01} , X_{12} , and X_{02} . The gate $X_{\pm 1}$ sends $|t\rangle$ to $|(t \pm 1) \bmod 3\rangle$ for $t \in \{0, 1, 2\}$; X_{01} is just the qubit X gate which is the identity when the input is $|2\rangle$; X_{12} sends $|1\rangle$ to $|2\rangle$ and $|2\rangle$ to $|1\rangle$, and likewise for X_{02} . Note that the qutrit Pauli X gate is the X_{+1} gate, while $X^\dagger = X_{-1} = X^2$.

Another concept that translates to qutrits (or more generally qudits) is that of Clifford unitaries.

Definition 2.1. Let U be a qudit unitary acting on n qudits. We say it is *Clifford* when every Pauli is mapped to another Pauli under conjugation by U . I.e. if UPU^\dagger is a Pauli for any Pauli P .

The set of n -qudit Cliffords forms a group under composition. For qubits, this group is generated by the S , Hadamard and CX gates. The same is true for qutrits, for the right generalisation of these gates¹ [30].

¹The gate definitions for various qudit Cliffords may vary across the literature up to a global phase. Indeed, by Definition 2.1, whether a gate is Clifford is invariant under changes in global phase.

Definition 2.2. The qutrit S gate is $S := \text{diag}(1, 1, \omega)$. I.e. it multiplies the $|2\rangle$ state by the phase ω .

For qubits, the Hadamard gate interchanges the Z eigenbasis $\{|0\rangle, |1\rangle$ and the X eigenbasis consisting of the states $|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. The same holds for the qutrit Hadamard. In this case the X basis consists of the following states:

$$|+\rangle := \frac{1}{\sqrt{3}}(|0\rangle + |1\rangle + |2\rangle) \quad |\omega\rangle := \frac{1}{\sqrt{3}}(|0\rangle + \omega|1\rangle + \bar{\omega}|2\rangle) \quad |\bar{\omega}\rangle := \frac{1}{\sqrt{3}}(|0\rangle + \bar{\omega}|1\rangle + \omega|2\rangle)$$

Definition 2.3. The qutrit Hadamard gate H is the unitary mapping $|0\rangle \mapsto |+\rangle$, $|1\rangle \mapsto |\omega\rangle$ and $|2\rangle \mapsto |\bar{\omega}\rangle$.

$$H := \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega & \bar{\omega} \\ 1 & \bar{\omega} & \omega \end{pmatrix} \quad (2)$$

Note that, unlike the qubit Hadamard, the qutrit Hadamard is *not* self-inverse. Instead we have $H^2 = X_{12}$ so that $H^4 = \mathbb{I}$. This means that $H^\dagger = H^3$.

The final Clifford gate we need is the qutrit CX.

Definition 2.4. The qutrit CX is defined such that $\text{CX}|i, j\rangle = |i, (i+j) \bmod 3\rangle$, where $i, j \in \{0, 1, 2\}$.

Any qutrit Clifford unitary can be written as a composition of S , H and CX gates (up to global phase). Clifford gates are efficiently classically simulable, so we need to add a non-Clifford gate to get an (approximately) universal gate set for quantum computing [30]. Here we consider when this is a phase gate.

Definition 2.5. We write $Z(a, b)$ for the *phase gate* that acts as $Z(a, b)|0\rangle = |0\rangle$, $Z(a, b)|1\rangle = \omega^a|1\rangle$ and $Z(a, b)|2\rangle = \omega^b|2\rangle$ where we take $a, b \in \mathbb{R}$.

We define $Z(a, b)$ in this way, taking a and b to correspond to powers of ω , because then $Z(a, b)$ is Clifford iff a and b are both integers, so that we can easily see from the parameters whether the gate is Clifford or not. The group of $Z(a, b)$ phase gates constitutes the group of diagonal single-qutrit unitaries modded out by a global phase. Composition of these gates is given by $Z(a, b) \cdot Z(c, d) = Z(a+c, b+d)$. Note that $S = Z(0, 1)$. This brings us to the definition of the qutrit T gate.

Definition 2.6. The qutrit T gate is defined as $T := Z(\frac{1}{3}, -\frac{1}{3}) = \text{diag}(1, \omega^{\frac{1}{3}}, \omega^{-\frac{1}{3}})$ [44, 13, 36].

Like the qubit T gate, the qutrit T gate belongs to the third level of the Clifford hierarchy, can be injected into a circuit using magic states, and its magic states can be distilled by magic state distillation. This means that we can fault-tolerantly implement this qutrit T gate on many types of quantum error correcting codes. Also as for qubits, the qutrit Clifford+ T gate set is approximately universal, meaning that we can approximate any qutrit unitary using just Clifford gates and the T gate [23, Theorem 1].

There is another useful single-qutrit non-Clifford gate.

Definition 2.7. The qutrit *reflection* gate is defined as $R := Z(0, 3/2) = \text{diag}(1, 1, -1)$.

Like the T gate, the R gate can be added to the Clifford gate set to attain universality [30], as explicitly proved in Ref. [23, Theorem 2]. It can be exactly synthesized fault-tolerantly in three known ways: magic state distillation followed by repeat-until-success injection [2], braiding and topological measurement of weakly-integral non-Abelian anyons [22, 23] followed by repeat-until-success injection [2], or unitarily in qutrit Clifford+ T [26].

2.1 Controlled unitaries

When we have an n -qubit unitary U , we can speak of the controlled gate that implements U . This is the $(n + 1)$ -qubit gate that acts as the identity when the first qubit is in the $|0\rangle$ state, and implements U on the last n qubits if the first qubit is in the $|1\rangle$ state. For qutrits there are multiple notions of control.

Definition 2.8. Let U be a qutrit unitary. Then the $|2\rangle$ -controlled U is the unitary that acts as

$$|0\rangle \otimes |\psi\rangle \mapsto |0\rangle \otimes |\psi\rangle \quad |1\rangle \otimes |\psi\rangle \mapsto |1\rangle \otimes |\psi\rangle \quad |2\rangle \otimes |\psi\rangle \mapsto |2\rangle \otimes U|\psi\rangle$$

I.e. it implements U on the last qutrits if and only if the first qutrit is in the $|2\rangle$ state.

Note that by conjugating the first qutrit with X_{+1} or X_{-1} gates we can make the gate also be controlled on the $|1\rangle$ or $|0\rangle$ state. A different notion of qutrit control was introduced in Ref. [10] where if the control is in the $|x\rangle$ state, then it should apply U^x on the target, i.e. apply U once iff $x = 1$ and U^2 iff $x = 2$. An example of this is the Clifford CX gate defined earlier, which applies X_{+1} when the control is $|1\rangle$ and X_{+2} when it is $|2\rangle$. Note that we can get this latter notion of control from the former: just apply a $|1\rangle$ -controlled U , followed by a $|2\rangle$ -controlled U^2 .

A number of Clifford+ T constructions for controlled qutrit unitaries are already known. For instance, all the $|2\rangle$ -controlled permutation X gates can be built from the constructions given in Ref. [9]. In our previous work, we provided ancilla-free explicit constructions for any multiple-controlled Clifford+ T unitary in the Clifford+ T gate set, with gate count polynomial in the number of controls [56]. In this work, by using the qutrit ZX-calculus, we will build upon our previous results and show how to construct multiple-controlled phase gates for an arbitrary phase.

3 The qutrit ZX-calculus

We will assume the reader has some familiarity with the original qubit ZX-calculus [17]. For a review see Ref. [53].

A qutrit ZX-calculus was presented and used in Refs. [45, 51, 28, 50, 48]. While quite similar to the qubit one, it loses some of the properties that make the original easy to work with. In particular, for each X-spider, the distinction between its input wires and output wires becomes important. This means we can no longer treat qutrit ZX-diagrams as undirected graphs with the spiders as vertices. This makes intuitive reasoning about these diagrams harder, and also complicates the implementation of software for dealing with these diagrams.

Here we will present a variation on the qutrit ZX-calculus of Refs. [28, 50] where the spiders do enjoy this additional symmetry between inputs and outputs. The way we do this is by redefining the X-spider. In the original qutrit ZX-calculus we have

$$\begin{array}{c} \diagup \\ \vdots \\ \text{---} \\ \vdots \\ \diagdown \end{array} \otimes \sum_{\substack{\vec{x}, \vec{y} \\ x_1 + \dots + x_n = y_1 + \dots + y_m}} |\vec{y}\rangle \langle \vec{x}|. \tag{3}$$

Here the sum $x_1 + \dots + x_n = y_1 + \dots + y_m$ is taken modulo 3. If we put a cup on one of the wires to turn an output into an input, then this has the effect of introducing a minus sign on that variable, changing for instance $x_1 + x_2 = y_1 + y_2$ into $x_1 + x_2 - y_2 = y_1$. For qubits this is not a problem since $-x = x$ modulo 2, but for qutrits this changes the map. We fix this by defining a new X-spider as

$$\begin{array}{c} \diagup \\ \vdots \\ \text{---} \\ \vdots \\ \diagdown \end{array} \otimes \sum_{\substack{\vec{x}, \vec{y} \\ x_1 + \dots + x_n + y_1 + \dots + y_m = 0}} |\vec{y}\rangle \langle \vec{x}|. \tag{4}$$

We see that in this definition the inputs and outputs are treated on equal footing. In order to prevent confusion with earlier work, we will denote this new X-spider in pink, instead of in red².

Let's now give the full definition of the spiders. We define the Z-spider as

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \end{array} \begin{array}{c} \alpha \\ \beta \end{array} = |0 \cdots 0\rangle \langle 0 \cdots 0| + \omega^\alpha |1 \cdots 1\rangle \langle 1 \cdots 1| + \omega^\beta |2 \cdots 2\rangle \langle 2 \cdots 2|.$$

Here we have two phase angles α and β , as opposed to just the one angle in qubit ZX. In general, for a d -dimensional spider, you will need to specify $d - 1$ phases. In particular, when written in a spider $\frac{\alpha}{\beta}$ should be interpreted as two different phases and not as the fraction α/β . Note that we define the phase angles as ω^α and ω^β so that these correspond to the complex phases $e^{\frac{2\pi i}{3}\alpha}$ and $e^{\frac{2\pi i}{3}\beta}$. This means that when α and β are integers, that the spiders correspond to the Clifford fragment of the calculus. We define the X-spider similarly, but with respect to the X-basis:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \end{array} \begin{array}{c} \alpha \\ \beta \end{array} = |+\cdots+\rangle \langle +\cdots+| + \omega^\alpha |\bar{\omega} \cdots \bar{\omega}\rangle \langle \omega \cdots \omega| + \omega^\beta |\omega \cdots \omega\rangle \langle \bar{\omega} \cdots \bar{\omega}|$$

This requires some explanation, because this does not look symmetric in the inputs and outputs. However, note that $\langle \omega| = (|\omega\rangle)^\dagger = (|0\rangle + \omega|1\rangle + \omega^2|2\rangle)^\dagger = \langle 0| + \bar{\omega}\langle 1| + \omega\langle 2|$. Hence, if we take the transpose of $|\omega\rangle$ we actually get $\langle \bar{\omega}|$. It is straightforward to check that with $\alpha = \beta = 0$ we get back Eq. (4). These definitions of the Z-spider and X-spider satisfy the symmetry properties we want, namely:

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

These symmetries mean our spiders are *flexsymmetric*, as defined by Carette [15], and as a result we may treat our ZX-diagrams as undirected graphs with the spiders as vertices. Note that here the cups and caps are defined with respect to the Z basis: $\subset = |00\rangle + |11\rangle + |22\rangle$. As usual, our calculus also formally has generators for the identity wire and the swap.

It will be useful to introduce an additional graphical generator for the Hadamard:

$$\llbracket \text{---} \text{---} \text{---} \rrbracket = |+\rangle \langle 0| + |\omega\rangle \langle 1| + |\bar{\omega}\rangle \langle 2| = |0\rangle \langle +| + |1\rangle \langle \bar{\omega}| + |2\rangle \langle \omega| \quad (5)$$

We write the Hadamard as a slanted box, because it is self-transpose, but not self-adjoint, and so should be denoted in a way that is symmetric under a rotation, but not a reflection.

Our redefinition of the X-spider comes at a 'cost'. Namely, the 1-input, 1-output X-spider is no longer the identity: $\text{---} \text{---} = |0\rangle \langle 0| + |2\rangle \langle 1| + |1\rangle \langle 2| = |+\rangle \langle +| + |\bar{\omega}\rangle \langle \omega| + |\omega\rangle \langle \bar{\omega}|$. This map is implementing $|x\rangle \mapsto |-x\rangle$ where $-x$ is taken modulo 3, and is equal to X_{12} . Additionally, the X-spider is not really a spider any more in the sense that it doesn't satisfy the standard spider-fusion equation. Instead it satisfies the 'harvestman equation' [15] that also holds for for instance the W-spider [32] and H-box [4]:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \end{array} \begin{array}{c} \eta \\ \beta \end{array} = \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \vdots \\ \text{---} \end{array} \begin{array}{c} \alpha + \eta \\ \beta + \theta \end{array}$$

In Figure 1, we present a full set of rewrite rules for this qutrit ZX-calculus. We have accounted for the global phase for each rule here as a complex number, as those will be relevant to us. Note however that the rewrite rules are not scalar-accurate as we are ignoring factors of $\sqrt{3}$.

²We have checked the accessibility of this color scheme; in fact, a red-green colorblind person greatly preferred this pink to the default ZX red.

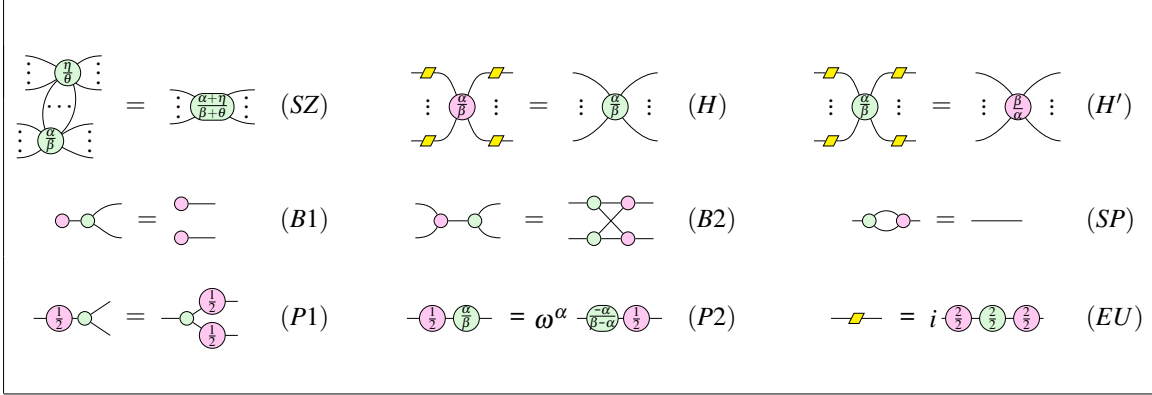


Figure 1: Rules for the flexsymmetric qutrit ZX-calculus. These hold for all $\alpha, \beta, \eta, \theta \in \mathbb{R}$, and for any permutation of the input and output wires. Additional useful derived rules are presented in Figure 2. The letters stand respectively for (S)pider-Z, (H)adamard, (B)ialgebra, (SP)ecial, (P)auli, and (EU)ler decomposition.

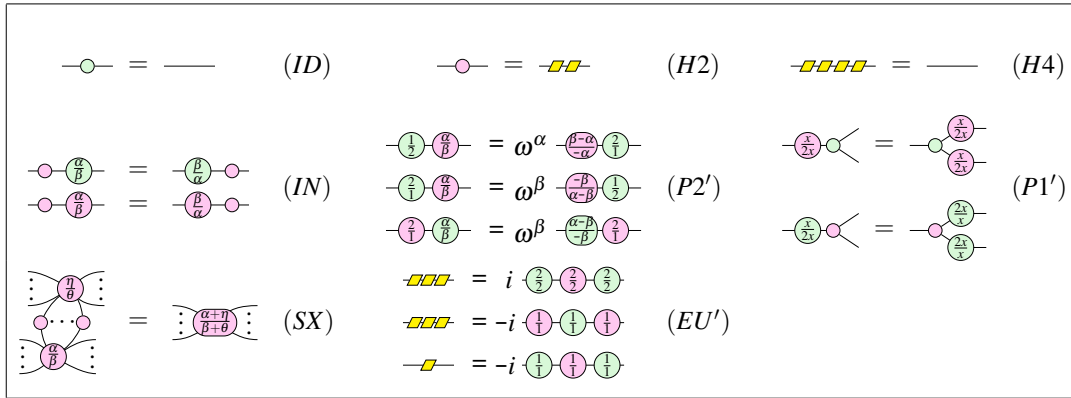


Figure 2: These rules are derivable from the rules of Figure 1 for any $\alpha, \beta, \eta, \theta \in \mathbb{R}$ and $x \in \{0, 1, 2\}$. The new letters stand respectively for (ID)entity, (IN)vert and (S)pider-X.

Using these rules, other useful qutrit ZX-calculus rewrite rules may be derived. In particular, we can use these rules to prove the derived rules presented in Figure 2. As these rules are (a slight variation) on the non-flexsymmetric qutrit rules of Ref. [50], our calculus is also complete for the qutrit Clifford fragment (when ignoring non-zero scalars). The proofs of the derived rules of Figure 2 are given in Appendix A.2. We show in Appendix A.1 that most rules of Figure 1 are necessary (i.e. not derivable from the others).

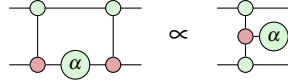
We see in these rules that there is a special role for phases of the form $\frac{x}{2x}$ where $x \in \{0, 1, 2\}$. This is because $\frac{x}{2x} \propto |x\rangle$ and $-\frac{x}{2x} = Z^x$. These relations can be derived by using the identity $1 + \omega + \bar{\omega} = 0$ together with $\omega^2 = \omega^{-1} = \bar{\omega}$. In general we will see a lot of $-\frac{\alpha}{2\alpha}$ phases because they implement the $|x\rangle \mapsto \omega^{\alpha x} |x\rangle$ phase gate. Additionally, note that the (P2) rule on the qubit subspace is exactly the familiar qubit ZX rule $-\pi \alpha = e^{i\alpha} -\alpha \pi$, since the red π is the qubit Pauli X while the pink $\frac{1}{2}$ phase is the qutrit Pauli X.

4 Diagonal qutrit gates

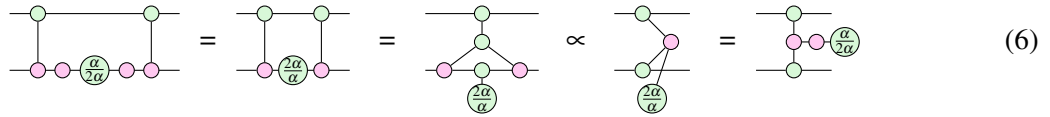
4.1 Phase gadgets

For qubits the concept of a *phase gadget* has proven very useful. There's several different ways we can define a qubit phase gadget. One way is to consider it as the diagonal gate $|x, y\rangle \mapsto e^{i\alpha(x \oplus y)} |x, y\rangle$ (for simplicity we are only considering a two-qubit phase gadget). This applies a phase of $e^{i\alpha}$ when $x \oplus y = 1$. Here \oplus is the XOR operation, which is the addition on \mathbb{Z}_2 . This suggests that we should define the qutrit phase gadget as $|x, y\rangle \mapsto e^{i\alpha(x+y)} |x, y\rangle$ where now we take $x+y$ to be modulo 3.

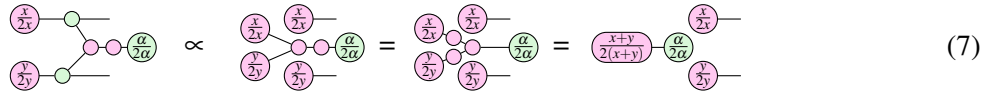
We could also define a phase gadget by its circuit realisation or diagrammatic representation. For qubits [38]:



We claim the qutrit variant of this construction is given by the following circuit which can be simplified to a similar diagrammatic representation:

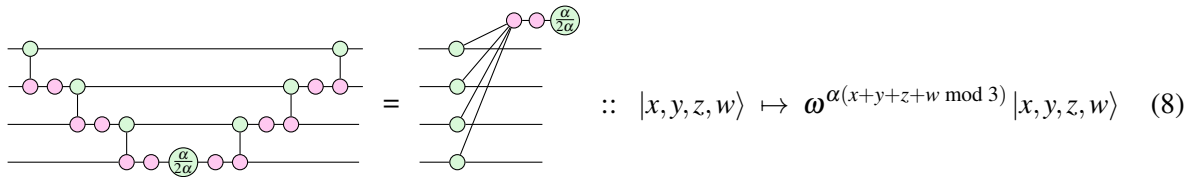


Indeed, inputting $|x, y\rangle$ into this diagram allows us to calculate its action:



This 'floating scalar' expression evaluates to $\sqrt{3}\omega^{\alpha(x+y \bmod 3)}$, so that this diagram indeed implements the operation we want, and we see that these three ways to define a qubit phase gadget—via the action, via the circuit, or via the diagrammatic representation—are also equal for qutrits.

We can easily generalise this construction to an arbitrary number of qutrits:



We can also define more general phase gadgets where the phases don't have to be related to each other, i.e. we can replace $Z(\alpha, 2\alpha)$ with $Z(\alpha, \beta)$. In this case we would still be calculating the value $x+y+z+w$ modulo 3, but then we apply a different phase depending on the value of this sum: if it is 0 we don't apply any phase; if it is 1 we apply ω^α ; and if it is 2 we apply ω^β .

A particularly relevant choice of phases here is when $\alpha = \beta$. In this case, we apply the phase iff the sum value is not 0. For a trit x it turns out that $x^2 = 0$ if $x = 0$ and $x^2 = 1$ otherwise — this is actually a consequence of Fermat's little theorem and generalises to $x^{p-1} = 1$ modulo p when $x \neq 0$ for p prime. Hence:

$$\text{---} \left(\frac{\alpha}{\alpha} \right) \text{---} :: |x\rangle \mapsto \omega^{\alpha(x^2 \bmod 3)} |x\rangle. \quad (9)$$

There is a complication with the phase gadget circuit representation that doesn't arise in the qubit setting, which is that the CNOT gate is self-inverse while the CX qutrit gate is not. In Eq. (6) we needed

to pair a CX with a CX[†] to make the construction work. If we instead have a pair of CX gates, we get something a bit more complicated:

$$\text{Diagram 1} = \text{Diagram 2} \stackrel{(6)}{=} \text{Diagram 3} = \text{Diagram 4} \quad (10)$$

Remark 4.1. Another way to view qubit phase gadgets is as an exponentiated Pauli $e^{i\alpha Z \otimes Z}$ [18, 19]. This however does *not* generalise to qutrits, as the qutrit Pauli Z is not self-adjoint, and hence cannot be exponentiated to give a unitary. In fact, a qutrit phase gadget cannot be represented as the exponential of a ‘pure tensor’ like $e^{i\alpha A \otimes B}$. This does suggest that there could be another suitable generalisation of a phase gadget that is the exponential of a tensor of *Gell-Mann matrices*, a qutrit basis of self-adjoint matrices.

4.2 Controlled phase gates

The other type of useful diagonal gate for qubits is the *controlled phase gate*. Such a gate applies a $Z(\alpha)$ gate on a qubit controlled on the value of a control. There are multiple ways in which we can generalise these to the qutrit setting. The type of control we will consider first is the $|2\rangle$ -control of Definition 2.8. To see how we can build a $|2\rangle$ -controlled Z phase gate, we will take inspiration from the qubit construction. Recall that there we have:

$$\text{Diagram} \quad (11)$$

We can ‘port’ the right-hand side to the qutrit setting, by taking each of the phases to be a $Z(\alpha, \beta)$. However, we then run into some problems. It is easy to check that when the top qutrit (the control) is $|0\rangle$ that the diagram indeed acts as the identity on the bottom qutrit (the target). However, it implements a different phase gate on the target depending on whether the control is in $|1\rangle$ or $|2\rangle$:

$$\text{Diagram} \rightsquigarrow \begin{cases} Z(0,0) & \text{if control is } |0\rangle \\ Z(2\alpha - \beta, \alpha + \beta) & \text{if control is } |1\rangle \\ Z(\alpha + \beta, 2\beta - \alpha) & \text{if control is } |2\rangle \end{cases} \quad (12)$$

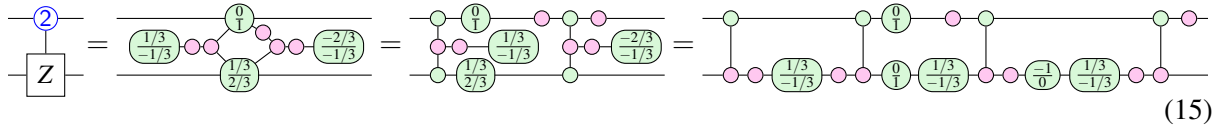
Seeing as we want to construct the $|2\rangle$ -controlled gate that should act as the identity when the control is $|1\rangle$ this is a problem. We solve this issue by ‘doubling up’ the construction, with the second construction being conjugated by X_{12} on the control in order to interchange the role of $|1\rangle$ and $|2\rangle$:

$$\text{Diagram 1} = \text{Diagram 2} = \text{Diagram 3} \quad (13)$$

By referring to Eq. (12) we see then that in order for Eq. (13) to be equal to the $|2\rangle$ -controlled $Z(\theta, \phi)$ gate it needs to satisfy a set of linear equations. We can solve these to get a (unique up to some Clifford phases) solution:

$$\alpha = \frac{\theta - \phi}{3} \quad \beta = \frac{\theta}{3} \quad \gamma = \frac{\phi}{3} \quad \delta = \frac{\phi - \theta}{3} \quad (14)$$

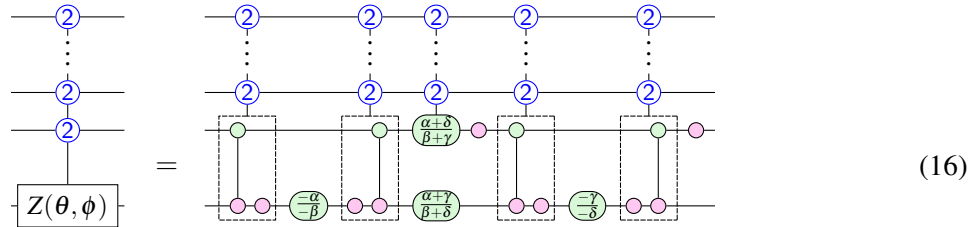
We can hence write any $|2\rangle$ -controlled phase gate using at most four CX gates and four phase gates. For example, if we pick $\theta = 1$ and $\phi = 2$ (so that we are constructing the controlled Z gate) we get:



Here we write this blue dot with a 2 in it to denote a $|2\rangle$ -control. We see then that our construction in the special case of $|2\rangle$ -controlled Paulis indeed achieves the lowest known T -count of 3 [9]. By conjugating the control wire by either X_{+1} or X_{-1} we can make the gate instead be controlled on either $|1\rangle$ or $|0\rangle$.

We can add any number of controls to our construction in Eq. (13) to make it controlled on any tritstring. Without loss of generality, let us say the tritstring in question is $|2\rangle^{\otimes n}$ (by conjugating with X gates, we can make this into a control on any tritstring of length n). The naïve way to construct this controlled gate is to inductively add controls to each gate in the decomposition: controlled constructions for the X or CX gate are described in, for instance, Ref. [56], while controlled Z phase gates can be constructed by recursively applying Eq. (13). However, this method is not efficient as it requires an exponential number of gates as the number of controls increases.

We can do better by not adding controls to all the gates in the decomposition:



In the case where all the controlled gates fire, this indeed implements any desired Z phase gate on the target qutrit. Otherwise, none of the controlled gates fire, and then the bottom two qutrits becomes identity (use $(H2)$ and $(H4)$ on the top qutrit and (SZ) and (ID) on the bottom qutrit). We hence get the following proposition.

Proposition 4.2. Any tritstring-controlled qutrit Z or X phase gate can be constructed without ancillae and with a polynomial number of Clifford and phase gates.

Proof. The X phase gates can be constructed from the Z phase gates by conjugating by Hadamards, so we only need to describe how to construct tritstring-controlled Z phase gates. Suppose we wish to construct a phase gate with n controls. By our prior work in Ref. [56], each $|2\rangle^{\otimes(n-1)}$ -controlled CX gate can be built ancilla-free using $O(n^{2.585})$ qutrit Clifford+T gates. It then remains to show how to construct the $|2\rangle^{\otimes(n-1)}$ -controlled Z phase gate in Eq. (16). We do this recursively. To construct the gate with k controls, we need four controlled CX gates with $k - 1$ controls and a $k - 1$ controlled Z phase gate, which then needs four controlled CX gates with $k - 2$ controls, and so on. The total asymptotic gate count is then $4O(n^{2.585}) + 4O((n - 1)^{2.585}) + \dots$ which gives us a gate count of $O(n^{3.585})$. \square

Note that in this construction, the size of the phases involved becomes exponentially smaller in the number of controls. We will next see that there is a notion of control which circumvents this issue.

4.3 Phase multipliers

The $|2\rangle$ -controlled phase gate is just one possible way to extend the idea of a controlled-phase gate from qubits. Another way is to realise that for qubits we can describe the action of a controlled phase gate as $|x, y\rangle \mapsto e^{i\alpha \cdot x \cdot y} |x, y\rangle$. Indeed, if the control qubit is in the state $x = 0$, then this is just the identity, while if $x = 1$, we apply $e^{i\alpha y}$ which corresponds to a $Z(\alpha)$ gate on the $|y\rangle$ qubit.

We see then that while a phase gadget is based on the addition operation of \mathbb{Z}_2 , controlled phase gates are based on the multiplication operation of \mathbb{Z}_2 . This suggests that the controlled phase gate equivalent for qutrits should be $|x, y\rangle \mapsto e^{i\alpha x \cdot y} |x, y\rangle$ where now we take $x \cdot y$ modulo 3. We will show how we can construct this operation using phase gadgets. In order to distinguish this type of gate from the previously considered controlled phase gates, we will refer to a gate where the phase depends on $x \cdot y$ as a *phase multiplier*. Before we show how to build phase multipliers for qutrits, we first need to understand how to build them for qubits. For bits x and y we have the relation

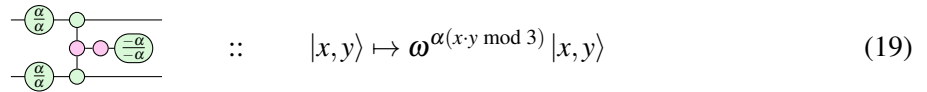
$$x \cdot y = \frac{1}{2}(x + y - (x \oplus y)). \tag{17}$$

Importantly, we are considering the $+$ operation here not modulo 2, but just as an action on real numbers, and we are writing \oplus for addition modulo 2. Using this relation we can write $e^{i\alpha(x \cdot y)} = e^{i\frac{1}{2}\alpha(x+y-(x \oplus y))} = e^{i\frac{1}{2}\alpha x} e^{i\frac{1}{2}\alpha y} e^{-i\frac{1}{2}\alpha(x \oplus y)}$. This is where the circuit decomposition of Eq. (11) comes from. This relation between additive and multiplicative phase gates follows from a Fourier-type duality that exists for semi-Boolean functions, which is explored in detail in Ref. [39].

It turns out that a similar decomposition is possible for qutrits. Note that we can derive Eq. (17) by starting with the equation $(x + y)^2 = x^2 + y^2 + 2x \cdot y$ and then realising that $x^2 = x$ for $x \in \{0, 1\}$ so that this reduces to $x \oplus y = x + y + 2x \cdot y$ for bits. When working with trits we can't remove these squares, but we can still get a useful relation. Bring terms to the other side to get $-2x \cdot y = x^2 + y^2 - (x + y)^2$ and then use the fact that modulo 3 we have $-2 = 1$ to get $x \cdot y = x^2 + y^2 - (x + y)^2$. It is now straightforward to check that this continues to hold when we interpret the outer $+$ and $-$ here not modulo 3, but as operations on the real numbers, so that we get the relation:

$$x \cdot y \text{ mod } 3 = (x^2 \text{ mod } 3) + (y^2 \text{ mod } 3) - ((x + y)^2 \text{ mod } 3) \tag{18}$$

Hence, using Eq. (9) we get the following decomposition:



$$\tag{19}$$

We can easily generalise Eq. (18) to as many variables as desired by iterating it. For three trits:

$$\begin{aligned} (x \cdot y) \cdot z &= x^2 \cdot z + y^2 \cdot z - (x + y)^2 \cdot z \\ &= x^4 + z^2 - (x^2 + z)^2 + y^4 + z^2 - (y^2 + z)^2 - (x + y)^4 - z^2 + ((x + y)^2 + z)^2 \\ &= x^2 + y^2 + z^2 - (x^2 + z)^2 - (y^2 + z)^2 - (x + y)^2 + ((x + y)^2 + z)^2 \end{aligned} \tag{20}$$

Here we used that $x^4 = x^2$ modulo 3.

Note that Eq. (9) shows how to apply a phase proportional to the input trit squared modulo 3. However, in order to use this trick to apply a phase proportional to a higher order term such as $(y + x^2)^2$, we need a way to compute $y + x^2$ and store it ‘‘on the wire’’. In other words, we need to construct a circuit for the unitary defined by $|x, y\rangle \mapsto |x, y + x^2\rangle$. Because this simply adds 1 (modulo 3) to y iff $x \neq 0$, we construct

it by adding 1 to the second qubit, and then applying a $|0\rangle$ -controlled X_{-1} gate. To build this gate, we use the $|2\rangle$ -controlled Z gate that we built from two phase gadgets in Eq. (15):

$$\begin{array}{c} \text{---} \textcircled{0} \text{---} \\ | \\ \text{---} X_{+1} X_{-1} \text{---} \end{array} = \begin{array}{c} X_{-1} \textcircled{2} X_{+1} \\ | \\ \text{---} H^\dagger Z^\dagger Z H \text{---} \end{array} \quad \because \quad |x, y\rangle \mapsto |x, y + x^2\rangle \quad (21)$$

The above trick was also described in Ref. [9]. We further use this to build the type of phase gate below.

$$\begin{array}{c} \text{---} \textcircled{0} \text{---} \textcircled{0} \text{---} \\ | \\ \text{---} X_{+1} X_{-1} \textcircled{\frac{\alpha}{2}} X_{+1} X_{-1} \text{---} \end{array} \quad \because \quad |x, y\rangle \mapsto \omega^{\alpha(y+x^2)^2} |x, y\rangle. \quad (22)$$

We now have all the ingredients necessary to build the phase multiplier corresponding to the formula (20). What is interesting about this is that we do not have to use smaller factors of α . This is in contrast to the qubit counterpart of the formula (20) where we get a factor of $\frac{1}{4}$, due to the factor of $\frac{1}{2}$ in Eq. (17). In fact, for qubits, the generalisation to n variables will have a prefactor of $(1/2)^{n-1}$ so that for instance the three-qubit-controlled Z and controlled T gates cannot be constructed without ancillae in Clifford+ T [25], as we need $\pi/8$ phase gates. Instead, no matter the number of qutrits, we do not get such a prefactor and can iteratively construct it as in the formula (20), as we did for qutrit Clifford+ T in Ref. [56]. The circuit (22), alongside the square phase of Eq. (9) suffices to generalise Eq. (19) to any number of qutrits.

Proposition 4.3. We can construct, without ancillae and using $O(2^n)$ Clifford+ T , $Z(\alpha, \alpha)$, and $Z(-\alpha, -\alpha)$ gates, the n -qutrit phase multiplier gate defined by $|x_1, \dots, x_n\rangle \mapsto \omega^{\alpha((x_1 \cdots x_n) \bmod 3)} |x_1, \dots, x_n\rangle$.

See Appendix B for the details.

Remark 4.4. In Ref. [21] the diagonal gates at all levels of the Clifford hierarchy are analysed for any qudit of prime dimension. They show for instance that the gate implementing $|x_1 \cdots x_n\rangle \mapsto \omega^{x_1 \cdots x_n} |x_1 \cdots x_n\rangle$ (which is the n -controlled $2\pi/3$ phase multiplier gate) is in the n th level of the Clifford hierarchy. This might be surprising as our construction shows how to build this gate, for any n , only using gates from the third level of the Clifford hierarchy (namely Clifford gates and the T gate). However, note that while the *diagonal* gates on a level of the hierarchy form a group, the full set of (not necessarily diagonal) gates is not closed under composition, and hence we can build higher-level unitaries using lower-level ones.

5 Applications

We've now seen that we can use phase gadgets to build a number of useful diagonal unitaries. In this section we will see how we can build more general diagonal qutrit unitaries, and specifically those that *emulate* qubit operations. Qudit emulation of qubit operations can result in efficiency gains, by using higher level states rather than ancillae. While there has been significant work on emulating qubits using qutrits and qudits, much of this has been limited to realising gates within classical reversible computing such as multiple-controlled Toffolis. In contrast, fewer works have addressed qutrit gate sets containing arbitrary phases. Examples include a $|2\rangle$ -controlled $Z(0, \phi)$ decomposition in terms of qutrit-controlled qubit $\phi/3$ rotations [24] or quantum multiplexers and uniformly-controlled Givens rotations from the cosine-sine decomposition [37].

Throughout this section, we will write $\stackrel{e}{=}$ to denote that a qubit unitary is emulated by a qutrit unitary.

We will first see how to emulate arbitrary qubit diagonal unitaries. Note that when we restrict to the $\{|0\rangle, |1\rangle\}$ subspace, that a qutrit phase multiplier $|x_1, \dots, x_n\rangle \mapsto e^{i\alpha((x_1 \cdots x_n) \bmod 3)} |x_1, \dots, x_n\rangle$ only applies a

phase α if and only if all n qubits are in the $|1\rangle$ state. Hence, for instance, the two-qubit $\text{CZ}(\alpha)$ gate is directly emulated by its two-qutrit counterpart of Eq. (19) with action $|x, y\rangle \mapsto e^{i\alpha x \cdot y} |x, y\rangle$. Consequently, by Proposition 4.3 we see that using qutrit Clifford+ T gates along with $Z(\alpha, \alpha)$ and $Z(-\alpha, -\alpha)$ we can emulate the multiple-controlled $Z(\alpha)$ qubit gate without ancillae.

Now, by conjugating a multiple-controlled $\text{CZ}(\alpha)$ gate by the appropriate X_{01} gates, we can decide on which input the $e^{i\alpha}$ phase is applied. Using multiple of these gates we can then arbitrarily decide for each input which phase should be applied to it. This then allows us to emulate an arbitrary diagonal qubit gate.

Proposition 5.1. We can emulate the $\text{diag}(\omega^{\alpha_1}, \dots, \omega^{\alpha_{2^n}})$ qubit unitary using qutrit Clifford+ T , $Z(\alpha_j, \alpha_j)$ and $Z(-\alpha_j, -\alpha_j)$ gates and without using ancillae.

When using a standard qubit unitary synthesis algorithm, the desired phases $e^{i\alpha_j}$ would be implemented using many-controlled phase gates that require exponentially small angles $e^{i\alpha_j/2^n}$, which is problematic when the use-case is in fault-tolerant computing where non-Clifford phase gates must be constructed using magic state distillation and injection. Our construction could hence lead to some benefits in synthesising diagonal qubit unitaries using less non-Clifford resources.

It turns out that for the specific case of a qubit CCZ gate, that we can emulate it using qutrits in an even more efficient way. While we could use the emulation construction above, it turns out to be better to consider an altered construction.

Lemma 5.2. Given a qutrit $|2\rangle$ -controlled U gate for an emulated qubit unitary U , we can construct a qutrit emulation of the qubit CCU gate with the same non-Clifford cost as the $|2\rangle$ -controlled U gate.

Proof. One can readily verify, by initializing the top two qubits to $\{|00\rangle, |01\rangle, |10\rangle, \text{ and } |11\rangle\}$, that the below qutrit decomposition from Ref. [27] emulates the qubit CCU gate.

$$\begin{array}{c} \bullet \\ | \\ \bullet \\ | \\ \square U \end{array} \stackrel{e}{=} \begin{array}{c} \textcircled{1} \quad \textcircled{1} \\ | \quad | \\ \square X_{+1} \quad \square X_{-1} \\ | \quad | \\ \textcircled{2} \\ | \\ \square U \end{array} \quad (23)$$

We can then replace the two non-Clifford $|1\rangle$ -controlled X_{+1} and $|1\rangle$ -controlled X_{-1} by CX and CX^\dagger , preserving correctness of the emulation as the action on the $\{|0\rangle, |1\rangle\}$ subspace is unchanged [10]. \square

Using this lemma we see that to get an efficient emulation of the qubit CCZ, it remains to find an efficient qutrit emulation of the $|2\rangle$ -controlled qubit Z gate.

Lemma 5.3. Let $U = \text{diag}(1, \omega^\eta)$ be an arbitrary qubit Z phase gate. Then we can build the $|2\rangle$ -controlled emulated U using the controlled phase gate of Eq. (12):

$$\begin{array}{c} \textcircled{2} \\ | \\ \square U \end{array} \stackrel{e}{=} \begin{array}{c} \textcircled{1} \quad \textcircled{2} \\ | \quad | \\ \textcircled{?} \quad \textcircled{?} \\ | \quad | \end{array} \stackrel{e}{=} \begin{array}{c} \textcircled{1} \quad \textcircled{2} \\ | \quad | \\ \frac{2\alpha-\beta}{\alpha+\beta} \quad \frac{\alpha+\beta}{2\beta-\alpha} \\ | \quad | \end{array} = \begin{array}{c} \frac{\alpha}{\beta} \\ | \\ \textcircled{?} \quad \textcircled{?} \\ | \\ \frac{\alpha}{\beta} \end{array} \quad (24)$$

Here α and β satisfy, for some $k \in \mathbb{Z}$, $2\alpha - \beta = 3k$ and $\alpha + \beta = \eta$ and the questionmarks ? denote that these phases are irrelevant for the emulation.

If we choose $\alpha = 3/2$ and $\beta = 0$ in this construction we are emulating the $|2\rangle$ -controlled qubit Z gate, because the phase of $\omega^{3/2} = -1$ applies iff the control qutrit is $|2\rangle$ and the target qubit is $|1\rangle$. Note that a $Z(3/2, 0)$ phase is equal to $X_{12} R X_{12}$, referring to the R gate from Definition 2.7. Hence:

Corollary 5.4. The $|2\rangle$ -controlled qubit Z gate can be emulated with R -count 3.

Combining this corollary with Lemma 5.2 we arrive at our result.

Proposition 5.5. The qubit CCZ gate can be emulated ancilla-free in qutrit Clifford+ R with R -count 3.

As shown in Ref. [35], any implementation of a CCZ gate requires at least four qubit T gates. Additionally any unitary implementation requires at least seven [29]. Here we see that surprisingly, by embedding the CCZ into qutrit space, we can construct it using just three non-Clifford single-qutrit gates and that moreover this is unitary and ancilla-free. This is also a new minimum amongst qudit emulations: for instance, in Ref. [33], they needed four qudit (for prime $d > 3$) T gates to emulate qubit CCZ.

6 Conclusion

We introduced phase gadgets in the qutrit ZX-calculus. To do this, we adapted the original qutrit ZX-calculus to be flexsymmetric so that the phase gadgets' behaviour would not depend on the directionality of their edges. Using phase gadgets we showed how to build two types of qutrit controlled phase gates: tritstring-controlled phase gates and phase multipliers. This allowed us to emulate the qubit CCZ gate using just three single-qudit non-Clifford gates.

While some of our constructions will naturally generalise to arbitrary qudit dimension, some things are qutrit specific. It seems to be a coincidence that for qutrits, in contrast with other-dimensional qudits, you can derive a relation between modular multiplication and addition (18) from the same binomial as for qubits (17), which comes from having a natural way to express $x^2 \bmod 3$ thanks to Fermat's little theorem. As a result, qubit and qutrit phase multipliers admit constructions which are structurally similar, despite the fact that for qubits it applies a phase of α on only one possible input — where all n qubits are $|1\rangle$ — while for qutrits it applies a phase, which can be α or 2α , for 2^n of the 3^n possible input basis states. Moreover, it seems quite special that Eq. (18) does not have any factors making the size of the phases internal to the decomposition decrease (in contrast to the qubit case).

We believe we could use these results as a stepping stone towards defining a qutrit ZH-calculus [4]. In the qubit ZH-calculus, the H-boxes represent matrices with coefficients $a^{i_1 \dots i_m j_1 \dots j_n}$ for a complex number a and $i_1, \dots, i_m, j_1, \dots, j_n \in \{0, 1\}$. Therefore, the obvious generalisation to qutrits (at least for a a complex phase) corresponds to our qutrit phase multipliers. Phase gadgets and phase multipliers could then be related in the same way as they are for qubit ZX and ZH [39].

An open problem is to find a suitable qutrit equivalent of exponentiated Paulis. The canonical self-adjoint generalisation of qubit Paulis to qutrits, the Gell-Mann matrices, can be exponentiated to unitaries, but it is not clear how they are related to the qutrit Paulis exactly. A starting point to find the proper relation here is to express exponentiations using a Hermitian operator basis constructed from the qutrit Paulis [3].

Finally, let us mention that based on work on an earlier draft of this paper, a proposed scheme for physically implementing a qutrit phase gadget in superconducting qutrit hardware was made [14].

Acknowledgements: JvdW is supported by an NWO Rubicon personal fellowship. LY is supported by an Oxford - Basil Reeve Graduate Scholarship at Oriel College with the Clarendon Fund. The authors wish to thank Aleks Kissinger, Shuxiang Cao, Alex Cowtan and Will Simmons for valuable discussions.

References

- [1] Matthew Amy (2019): *Towards Large-scale Functional Verification of Universal Quantum Circuits*. In Peter Selinger & Giulio Chiribella, editors: *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018, Electronic Proceedings in Theoretical Computer Science 287*, Open Publishing Association, pp. 1–21, doi:10.4204/EPTCS.287.1.

- [2] 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.
- [3] Ali Asadian, Paul Erker, Marcus Huber & Claude Klöckl (2016): *Heisenberg-Weyl Observables: Bloch vectors in phase space*. *Phys. Rev. A* 94, p. 010301, doi:10.1103/PhysRevA.94.010301.
- [4] Miriam Backens & Aleks Kissinger (2019): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*. In Peter Selinger & Giulio Chiribella, editors: *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018, Electronic Proceedings in Theoretical Computer Science* 287, Open Publishing Association, pp. 23–42, doi:10.4204/EPTCS.287.2.
- [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): *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.
- [7] 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.
- [8] M. S. Blok, V. V. Ramasesh, T. Schuster, K. O’Brien, J. M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao & I. Siddiqi (2021): *Quantum Information Scrambling on a Superconducting Qutrit Processor*. *Phys. Rev. X* 11, p. 021010, doi:10.1103/PhysRevX.11.021010.
- [9] 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.
- [10] 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.
- [11] Stephen Bullock, Dianne O’Leary & Gavin Brennen (2005): *Asymptotically Optimal Quantum Circuits for d -level Systems*. *Physical Review Letters* 94(23), doi:10.1103/physrevlett.94.230502.
- [12] Earl T. Campbell (2014): *Enhanced Fault-Tolerant Quantum Computing in d -Level Systems*. *Phys. Rev. Lett.* 113, p. 230501, doi:10.1103/PhysRevLett.113.230501.
- [13] 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.
- [14] Shuxiang Cao, Lia Yeh, Bakr Mustafa S, Giulio Campanaro, Simone D Fasciati, James F Wills, Boris Shteynas, Vivek Chidambaram, John van de Wetering & Peter J Leek (2022): *Qutrit-ZX Calculus on superconducting transmon qutrits*. Available at <https://meetings.aps.org/Meeting/MAR22/Session/F37.7>.
- [15] Titouan Carette (2021): *When Only Topology Matters*. *arXiv preprint arXiv:2102.03178*, doi:10.48550/arXiv.2102.03178.
- [16] Ji Chu, Xiaoyu He, Yuxuan Zhou, Jiahao Yuan, Libo Zhang, Qihao Guo, Yongju Hai, Zhikun Han, Chang-Kang Hu, Wenhui Huang, Hao Jia, Dawei Jiao, Sai Li, Yang Liu, Zhongchu Ni, Lifu Nie, Xianchuang Pan, Jiawei Qiu, Weiwei Wei, Wuerkaixi Nuerbolati, Zusheng Yang, Jiajian Zhang, Zhida Zhang, Wanjing Zou, Yuanzhen Chen, Xiaowei Deng, Xiuhao Deng, Ling Hu, Jian Li, Song Liu, Yao Lu, Jingjing Niu, Dian Tan, Yuan Xu, Tongxing Yan, Youpeng Zhong, Fei Yan, Xiaoming Sun & Dapeng Yu (2023): *Scalable Algorithm Simplification Using Quantum AND Logic*. *Nature Physics* 19(1), pp. 126–131, doi:10.1038/s41567-022-01813-7.
- [17] 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.

- [18] 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.
- [19] Alexander Cowtan, Will Simmons & Ross Duncan (2020): *A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz*. *arXiv preprint arXiv:2007.10515*, doi:10.48550/arXiv.2007.10515.
- [20] Daniele Cozzolino, Beatrice Da Lio, Davide Bacco & Leif Katsuo Oxenløwe (2019): *High-Dimensional Quantum Communication: Benefits, Progress, and Future Challenges*. *Advanced Quantum Technologies* 2(12), p. 1900038, doi:10.1002/qute.201900038.
- [21] Shawn X. Cui, Daniel Gottesman & Anirudh Krishna (2017): *Diagonal gates in the Clifford hierarchy*. *Phys. Rev. A* 95, p. 012329, doi:10.1103/PhysRevA.95.012329.
- [22] Shawn X. Cui, Seung-Moon Hong & Zhenghan Wang (2015): *Universal quantum computation with weakly integral anyons*. *Quantum Information Processing* 14(8), p. 2687–2727, doi:10.1007/s11128-015-1016-y.
- [23] 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.
- [24] Yao-Min Di & Hai-Rui Wei (2013): *Synthesis of multivalued quantum logic circuits by elementary gates*. *Phys. Rev. A* 87, p. 012325, doi:10.1103/PhysRevA.87.012325.
- [25] Brett Giles & Peter Selinger (2013): *Exact synthesis of multiqubit Clifford+T circuits*. *Physical Review A* 87(3), doi:10.1103/physreva.87.032332.
- [26] 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.
- [27] Pranav Gokhale, Jonathan M. Baker, Casey Duckering, Natalie C. Brown, Kenneth R. Brown & Frederic T. Chong (2019): *Asymptotic improvements to quantum circuits via qutrits*. *Proceedings of the 46th International Symposium on Computer Architecture*, doi:10.1145/3307650.3322253.
- [28] Xiaoyan Gong & Quanlong Wang (2017): *Equivalence of Local Complementation and Euler Decomposition in the Qutrit ZX-calculus*, doi:10.48550/arXiv.1704.05955.
- [29] David Gosset, Vadym Kliuchnikov, Michele Mosca & Vincent Russo (2014): *An Algorithm for the T-Count*. *Quantum Info. Comput.* 14(15-16), pp. 1261–1276, doi:10.5555/2685179.2685180.
- [30] Daniel Gottesman (1999): *Fault-Tolerant Quantum Computation with Higher-Dimensional Systems*. *Chaos, Solitons & Fractals* 10(10), p. 1749–1758, doi:10.1016/s0960-0779(98)00218-5.
- [31] Arianne Meijer-van de Griend & Ross Duncan (2020): *Architecture-aware synthesis of phase polynomials for NISQ devices*. *ArXiv Preprint arxiv:2004.06052*, doi:10.48550/ARXIV.2004.06052.
- [32] 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.
- [33] Luke E. Heyfron & Earl Campbell (2019): *A quantum compiler for qudits of prime dimension greater than 3*. *ArXiv Preprint arxiv:1902.05634*, doi:10.48550/arXiv.1902.05634.
- [34] Alexander D. Hill, Mark J. Hodson, Nicolas Didier & Matthew J. Reagor (2021): *Realization of arbitrary doubly-controlled quantum phase gates*. doi:10.48550/arXiv.2108.01652.
- [35] Mark Howard & Earl Campbell (2017): *Application of a Resource Theory for Magic States to Fault-Tolerant Quantum Computing*. *Phys. Rev. Lett.* 118, p. 090501, doi:10.1103/PhysRevLett.118.090501.
- [36] 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.

- [37] Faisal Shah Khan & Marek Perkowski (2006): *Synthesis of multi-qudit hybrid and d -valued quantum logic circuits by decomposition*. *Theoretical Computer Science* 367(3), pp. 336–346, doi:10.1016/j.tcs.2006.09.006.
- [38] 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.
- [39] Stach Kuijpers, John van de Wetering & Aleks Kissinger (2019): *Graphical Fourier Theory and the Cost of Quantum Addition*. *arXiv preprint arXiv:1904.07551*, doi:10.48550/arXiv.1904.07551.
- [40] François Mallet, Florian R. Ong, Agustin Palacios-Laloy, François Nguyen, Patrice Bertet, Denis Vion & Daniel Esteve (2009): *Single-shot qubit readout in circuit quantum electrodynamics*. *Nature Physics* 5(11), pp. 791–795, doi:10.1038/nphys1400.
- [41] A. S. Nikolaeva, E. O. Kiktenko & A. K. Fedorov (2022): *Decomposing the generalized Toffoli gate with qutrits*. *Physical Review A* 105(3), doi:10.1103/physreva.105.032621.
- [42] L. Petit, M. Russ, H. G. J. Eenink, W. I. L. Lawrie, J. S. Clarke, L. M. K. Vandersypen & M. Veldhorst (2020): *High-fidelity two-qubit gates in silicon above one Kelvin*, doi:10.48550/ARXIV.2007.09034.
- [43] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson & B. Neyenhuis (2021): *Demonstration of the trapped-ion quantum CCD computer architecture*. *Nature* 592(7853), pp. 209–213, doi:10.1038/s41586-021-03318-4.
- [44] 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(3), doi:10.1103/physreva.98.032304.
- [45] André Ranchin (2014): *Depicting qudit quantum mechanics and mutually unbiased qudit theories*. In Bob Coecke, Ichiro Hasuo & Prakash Panangaden, editors: *Proceedings of the 11th workshop on Quantum Physics and Logic, Kyoto, Japan, 4-6th June 2014, Electronic Proceedings in Theoretical Computer Science* 172, Open Publishing Association, pp. 68–91, doi:10.4204/EPTCS.172.6.
- [46] Martin Ringbauer, Michael Meth, Lukas Postler, Roman Stricker, Rainer Blatt, Philipp Schindler & Thomas Monz (2022): *A universal qudit quantum processor with trapped ions*. *Nature Physics* 18, pp. 1053–1057, doi:10.1038/s41567-022-01658-0.
- [47] Sarah Sheldon, Easwar Magesan, Jerry M. Chow & Jay M. Gambetta (2016): *Procedure for systematically tuning up cross-talk in the cross-resonance gate*. *Phys. Rev. A* 93, p. 060302, doi:10.1103/PhysRevA.93.060302.
- [48] Alex Townsend-Teague & Konstantinos Meichanetzidis (2021): *Classifying Complexity with the ZX-Calculus: Jones Polynomials and Potts Partition Functions*. *arXiv preprint arXiv:2103.06914*, doi:10.48550/arXiv.2103.06914.
- [49] 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.
- [50] Quanlong Wang (2018): *Qutrit ZX-calculus is Complete for Stabilizer Quantum Mechanics*. In Bob Coecke & Aleks Kissinger, editors: *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017, Electronic Proceedings in Theoretical Computer Science* 266, Open Publishing Association, pp. 58–70, doi:10.4204/EPTCS.266.3.
- [51] Quanlong Wang & Xiaoning Bian (2014): *Qutrit Dichromatic Calculus and Its Universality*. In Bob Coecke, Ichiro Hasuo & Prakash Panangaden, editors: *Proceedings of the 11th workshop on Quantum Physics and Logic, Kyoto, Japan, 4-6th June 2014, Electronic Proceedings in Theoretical Computer Science* 172, Open Publishing Association, pp. 92–101, doi:10.4204/EPTCS.172.7.
- [52] Yuchen Wang, Zixuan Hu, Barry C. Sanders & Sabre Kais (2020): *Qudits and High-Dimensional Quantum Computing*. *Frontiers in Physics* 8, p. 479, doi:10.3389/fphy.2020.589504.
- [53] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. *arXiv preprint arXiv:2012.13966*, doi:10.48550/arXiv.2012.13966.

- [54] John van de Wetering (2021): *Constructing quantum circuits with global gates*. *New Journal of Physics* 23(4), p. 043015, doi:10.1088/1367-2630/abf1b3.
- [55] Biaoliang Ye, Zhen-Fei Zheng, Yu Zhang & Chui-Ping Yang (2018): *Circuit QED: single-step realization of a multiqubit controlled phase gate with one microwave photonic qubit simultaneously controlling $n - 1$ microwave photonic qubits*. *Optics Express* 26(23), p. 30689, doi:10.1364/oe.26.030689.
- [56] 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.
- [57] M. A. Yurtalan, J. Shi, M. Kononenko, A. Lupascu & S. Ashhab (2020): *Implementation of a Walsh-Hadamard Gate in a Superconducting Qutrit*. *Phys. Rev. Lett.* 125, p. 180504, doi:10.1103/PhysRevLett.125.180504.

A Qutrit ZX-calculus

A.1 Necessity of rules

We can show that most of the rules in Figure 1 are *necessary*, meaning that they cannot be derived from the other rules. We do this by adapting the reasoning of Ref. [49]. Namely, the following rules are definitely necessary:

- (SZ): this is the only rule which can decompose a generator with four or more legs into generators with fewer legs.
- (P): this is the only rule which resolves diagrams containing generators to the identity.
- (B1): this is the only rule that can transform a connected diagram into a disconnected one.
- (EU): this is necessary per the argument of Ref. [51, Proposition 3.2].
- At least one of (H) and (H') is necessary as these are the only ones that can convert a diagram containing a X generator with a non-integer phase into one containing a Z generator with a non-integer phase.

We do not know whether the other rules are necessary, although we do suspect this is the case.

A.2 Proofs of the derived rules

Lemma A.1. The (ID) rule can be derived from the (SZ) and (SP) rules.

Proof.

$$\begin{array}{c} \text{(P)} \\ \text{---} \circ \text{---} \end{array} = \begin{array}{c} \text{(SZ)} \\ \text{---} \circ \text{---} \circ \text{---} \end{array} = \begin{array}{c} \text{(SP)} \\ \text{---} \circ \text{---} \circ \text{---} \end{array} = \text{---} \quad \square$$

Lemma A.2. The (H2) rule can be derived from the (H') and (ID) rules.

Proof.

$$\begin{array}{c} \text{(ID)} \\ \text{---} \color{red}{\diagup} \color{red}{\diagdown} \end{array} = \begin{array}{c} \text{(H')} \\ \text{---} \color{red}{\diagup} \circ \color{red}{\diagdown} \end{array} = \text{---} \circ \text{---} \quad \square$$

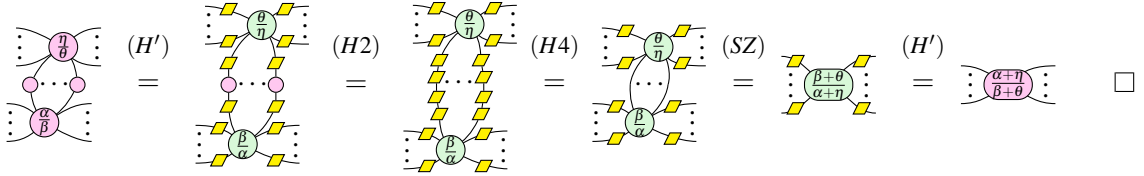
Lemma A.3. The (H4) rule can be derived from the (H), (ID), and (H2) rules.

Proof.

$$\begin{array}{c} \text{(H2)} \\ \text{---} \color{red}{\diagup} \color{red}{\diagdown} \color{red}{\diagup} \color{red}{\diagdown} \end{array} = \begin{array}{c} \text{(H)} \\ \text{---} \color{red}{\diagup} \circ \color{red}{\diagdown} \end{array} = \begin{array}{c} \text{(ID)} \\ \text{---} \circ \text{---} \end{array} = \text{---} \quad \square$$

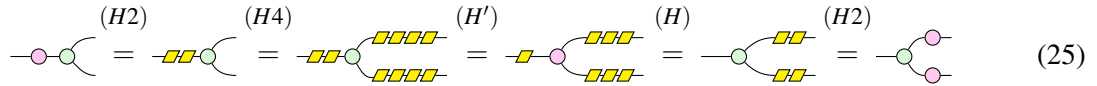
Lemma A.4. The (SX) rule can be derived from the (SZ) , (H') , $(H2)$, and $(H4)$ rules.

Proof.

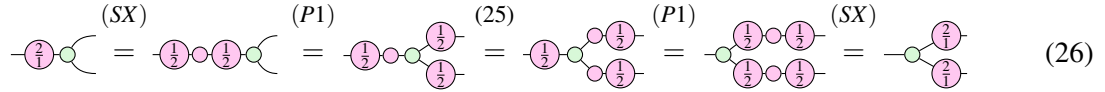


Lemma A.5. The $(P1')$ rules can be derived from the $(P1)$, (SX) , (H) , (H') , $(H2)$, and $(H4)$ rules.

Proof. Let's first derive the rule for $x = 0$:



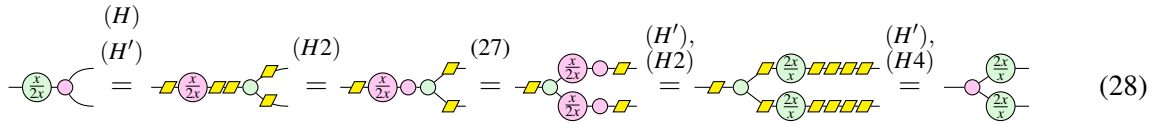
From that, we can derive the below rule:



The two above rules, along with the $(P1)$ rule, are captured by the following rule where $x \in \{0, 1, 2\}$:

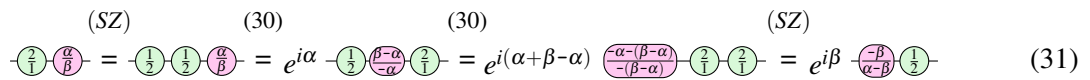
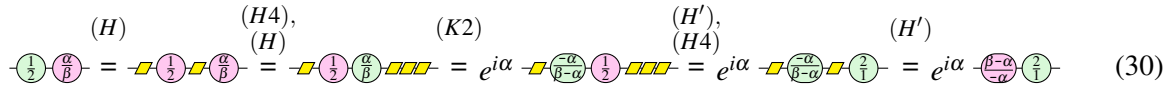
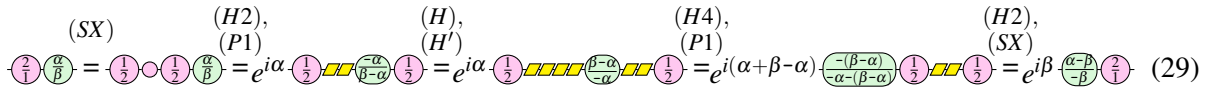


We now colour-change the above rule to finish deriving all the $(P1')$ rules:



Lemma A.6. The $(P2')$ rules can be derived from the $(P2)$, (H) , (H') , (SZ) , (SX) , $(H2)$, and $(H4)$ rules.

Proof. We prove them one by one:



Lemma A.7. The (EU') rules can be derived from the (EU) , (H) , (H') , (SZ) , (SX) , $(H2)$, and $(H4)$ rules.

Proof. We show the first equation directly:

$$\begin{array}{c} (EU) \\ \text{---} \end{array} = i \begin{array}{c} (H) \\ \text{---} \end{array} = i \begin{array}{c} (H) \\ \text{---} \end{array} = i \begin{array}{c} (H) \\ \text{---} \end{array} \quad (32)$$

For the second one we first note that:

$$\begin{array}{c} (H4) \\ (H2) \\ (25) \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} (25) \\ (EU) \\ (SX) \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} (25) \\ (SZ) \\ (ID) \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} (25) \\ (SX) \\ (H4) \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \text{---} \quad (33)$$

Then:

$$\begin{array}{c} (33) \\ \text{---} \end{array} = \begin{array}{c} (H4) \\ \text{---} \end{array} = \begin{array}{c} (H4) \\ \text{---} \end{array} \quad (34)$$

And finally, we find the different decomposition of H :

$$\begin{array}{c} (H4) \\ \text{---} \end{array} = \begin{array}{c} (34) \\ \text{---} \end{array} = \begin{array}{c} (H) \\ \text{---} \end{array} = \begin{array}{c} (H) \\ \text{---} \end{array} = \begin{array}{c} (H) \\ \text{---} \end{array} \quad (35)$$

□

B Constructing general phase multipliers

When we have two variables we use the formula

$$x \cdot y \bmod 3 = (x^2 \bmod 3) + (y^2 \bmod 3) - ((x+y)^2 \bmod 3) \quad (36)$$

to construct the two-qutrit phase multiplier. We generalised this to three variables in the following way:

$$(x \cdot y) \cdot z = x^2 \cdot z + y^2 \cdot z - (x+y)^2 \cdot z = x^2 + y^2 + z^2 - (x^2 + z)^2 - (y^2 + z)^2 - (x+y)^2 + ((x+y)^2 + z)^2. \quad (37)$$

To see how we go to 4 variables and beyond, we start with the expression $(x \cdot y \cdot z) \cdot w$ and decompose $x \cdot y \cdot z$ with the above formula resulting in terms t_1^2, \dots, t_n^2 . Each of these terms is a square, because that is the case for all the terms in Eq. (36). Since we are working with qutrits we have $(t_j^2)^2 = t_j^2$. The terms in our formula are now of the form $t_j^2 \cdot w$. We apply Eq. (36) to each of these. This gives us terms t_j^4 , w^2 and $(t_j^2 + w)^2$. The first of these is just t_j^2 , and by induction we already know how to construct the appropriate phase term on the circuit for this term. The second of these is w^2 , and hence corresponds to a simple phase gate. Note that this is the same for each $t_j^2 \cdot w$ we are decomposing. Furthermore, the plus signs and minus signs on the terms are such that they almost all cancel, and we will have one copy of w^2 .

The only ‘interesting’ new term we then get is hence $(t_j^2 + w)^2$. For instance, in Eq. (37) the terms of this form are $(x^2 + z)^2$, $(y^2 + z)^2$ and $((x+y)^2 + z)^2$. The corresponding phase terms are constructed by using the gadget of Eq. (21) to store $t_j^2 + w$ ‘on the wire’ and then applying a $Z(\alpha, \alpha)$ phase gate.

Hence, if we go from 3 to 4 variables we get each of the original terms t_j^2 , plus a w^2 term and a $(t_j^2 + w)^2$ term for each j . This straightforwardly generalises to n variables, and it is then easy to check that we will have $2^n - 1$ terms.

We can build a circuit for the $n > 2$ qutrit phase multiplier by first building the circuit for $n - 1$ qutrits, and then inserting the gadget of Eq. (22) after every application of a $Z(\alpha, \alpha)$ phase with as the target the n th qutrit. The reason this works is because the n -qutrit phase multiplier still contains every term of the $n - 1$ qutrit multiplier, but now also needs to combine those terms with the n th variable. In the four variable case, we would first store on a wire the value of the term t_j we need, and then apply a $Z(\alpha, \alpha)$ gate in order to get the phase $e^{i\alpha t_j^2}$. Then we would apply Eq. (22) on the qutrit of w in order to get the phase $e^{i\alpha(t_j^2 + w)^2}$. This construction involves temporarily storing $t_j^2 + w$ on the wire of w , so we can use this term if we want to construct the five-qutrit phase multiplier as well.

We then see that the cost of the n -qutrit phase multiplier in terms of (non-Clifford) gates is the cost of the $n - 1$ qutrit phase multiplier plus the cost of $2^{n-1} - 1$ applications of the Eq. (22) gadget. In particular, each phase term requires precisely one of either a $Z(\alpha, \alpha)$ or a $Z(-\alpha, -\alpha)$ gate, so that we need $2^n - 1$ of them. The circuit of Eq. (22) requires 6 T gates to construct, and hence the T -count of the n -qutrit phase multiplier is $6(2^{n-1} - 1) = 3 \cdot 2^n - 6$ (for $n > 2$).

Complete Flow-Preserving Rewrite Rules for MBQC Patterns with Pauli Measurements

Tommy McElvanney

School of Computer Science
University of Birmingham
txm639@student.bham.ac.uk

Miriam Backens

School of Computer Science
University of Birmingham
m.backens@cs.bham.ac.uk

In the one-way model of measurement-based quantum computation (MBQC), computation proceeds via measurements on some standard resource state. So-called flow conditions ensure that the overall computation is deterministic in a suitable sense, with Pauli flow being the most general of these. Existing work on rewriting MBQC patterns while preserving the existence of flow has focused on rewrites that reduce the number of qubits.

In this work, we show that introducing new Z -measured qubits, connected to any subset of the existing qubits, preserves the existence of Pauli flow. Furthermore, we give a unique canonical form for stabilizer ZX -diagrams inspired by recent work of Hu & Khesin [17]. We prove that any MBQC-like stabilizer ZX -diagram with Pauli flow can be rewritten into this canonical form using only rules which preserve the existence of Pauli flow, and that each of these rules can be reversed while also preserving the existence of Pauli flow. Hence we have complete graphical rewriting for MBQC-like stabilizer ZX -diagrams with Pauli flow.

1 Introduction

The one-way model of measurement-based quantum computation (MBQC) shows how to implement quantum computations by successive adaptive single-qubit measurements on a resource state [23], largely without using any unitary operations. This contrasts with the more commonly-used circuit model and has applications in server-client scenarios as well as for certain quantum error-correcting codes.

An MBQC computation is given as a *pattern*, which specifies the resource state – usually a graph state – and a sequence of measurements of certain types [12]. As measurements are non-deterministic, future measurements need to be adapted depending on the outcomes of past measurements to obtain an overall deterministic computation. Yet not every pattern can be implemented deterministically. Sufficient (and in some cases necessary) criteria for determinism are given by the different kinds of *flow*, which define a partial order on the measured qubits and give instructions for how to adapt the future computation if a measurement yields the undesired outcome [11, 8] (cf. Section 2.3).

In addition to the applications mentioned above, the flexible structure of MBQC patterns is also useful as a theoretical tool. For example, translations between circuits and MBQC patterns have been used to trade off circuit depth versus qubit number [7] or to reduce the number of T -gates in a Clifford+ T circuit [20]. When translating an MBQC pattern (back) into a circuit, it is important that the pattern still have flow, as circuit extraction algorithms rely on flow [11, 21, 14, 4]

This work uses the ZX -calculus, a graphical language for representing and reasoning about quantum computations, which is convenient for representing both quantum circuits and MBQC patterns, and for translating between the two. ZX -calculus diagrams directly corresponding to MBQC-patterns are said to be in *MBQC form*. The ZX -calculus has various complete sets of rewrite rules, meaning any two diagrams that represent the same linear map can be transformed into each other entirely graphically

[2, 18, 22]. Yet these rewrite rules do not necessarily preserve the existence of a flow, nor even the MBQC-form structure. Thus, circuit optimisation using MBQC and the ZX-calculus relies on proofs that certain diagram rewrites do preserve both [14, 4]. Work so far has focused on rewrite rules that maintain or reduce the number of qubits, which find direct application in T-count optimisation [14]. Nevertheless, it is sometimes desirable to increase the number of qubits in an MBQC pattern while preserving the existence of flow, such as for more involved optimisation strategies [25] or for obfuscation.

In this paper, we begin investigating rewrite rules that preserve the existence of flow while increasing the number of qubits. In particular, we prove that a rewrite rule that introduces a new Z-measured qubit preserves flow. Most work on flow-preserving rewriting so far has been done in the context of *generalised flow*, also known as *gflow* [8], in either its simple [14] or extended version [4]. Yet with the qubit introduction rule, the setting shifts to that of *Pauli flow* [8, 24] since preserving the interpretation of the diagram requires that the new qubit be measured in the Pauli-Z basis.

We show that adding this one new rule to the known flow-preserving rewrite rules suffices to get completeness for MBQC-form diagrams within the stabilizer fragment of the ZX-calculus. To achieve completeness, we introduce a new unique normal form for stabilizer ZX-calculus diagrams, which is close to the MBQC form. This normal form is based on work by Hu and Khesin [17] using the stabilizer graph notation of Elliott, Eastin and Caves [16], like the original stabilizer ZX-calculus completeness result [2]. As the proof by Hu and Khesin is somewhat difficult to follow, we give an alternative uniqueness proof using the language of affine spaces.

The remainder of this paper is structured as follows: in Section 2, we introduce the ZX-calculus, measurement-based quantum computing, and existing flow-preserving rewrite rules. Section 3 contains the new canonical form and its uniqueness proof. Section 4 presents the new flow-preserving rewrite rule and the completeness proof for the stabilizer MBQC-form fragment. The conclusions are in Section 5.

2 Preliminaries

In this section, we give an overview of the ZX-calculus and then use it to introduce measurement-based quantum computing. We discuss the notion of flow that will be used in this paper and some existing rewrite rules which preserve the existence of this flow.

2.1 The ZX-calculus

The ZX-calculus is a diagrammatic language for reasoning about quantum computations. We will provide a short introduction here; for a more thorough overview, see [27, 10].

A ZX-diagram consists of *spiders* and *wires*. Diagrams are read from left to right: wires entering a diagram from the left are inputs while wires exiting the diagram on the right are outputs, like in the quantum circuit model. ZX-diagrams compose in two distinct ways: *horizontal composition*, which involves connecting the output wires of one diagram to the input wires of another, and *vertical composition* (or the tensor product), which just involves drawing one diagram vertically above the other. The linear map corresponding to a ZX-diagram D is denoted by $\llbracket D \rrbracket$.

ZX-diagrams are generated by two families of spiders which may have any number of inputs or outputs, corresponding to the Z and X bases respectively. Z-spiders are drawn as green dots and X-spiders as red dots; with m inputs, n outputs, and using $(\cdot)^{\otimes k}$ to denote a k -fold tensor power, we have:

$$\left[\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \alpha \\ \diagdown \quad \diagup \\ \text{---} \\ \vdots \end{array} \right] = |0\rangle^{\otimes n} \langle 0|^{\otimes m} + e^{i\alpha} |1\rangle^{\otimes n} \langle 1|^{\otimes m} \quad \left[\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \alpha \\ \diagdown \quad \diagup \\ \text{---} \\ \vdots \end{array} \right] = |+\rangle^{\otimes n} \langle +|^{\otimes m} + e^{i\alpha} |-\rangle^{\otimes n} \langle -|^{\otimes m}$$

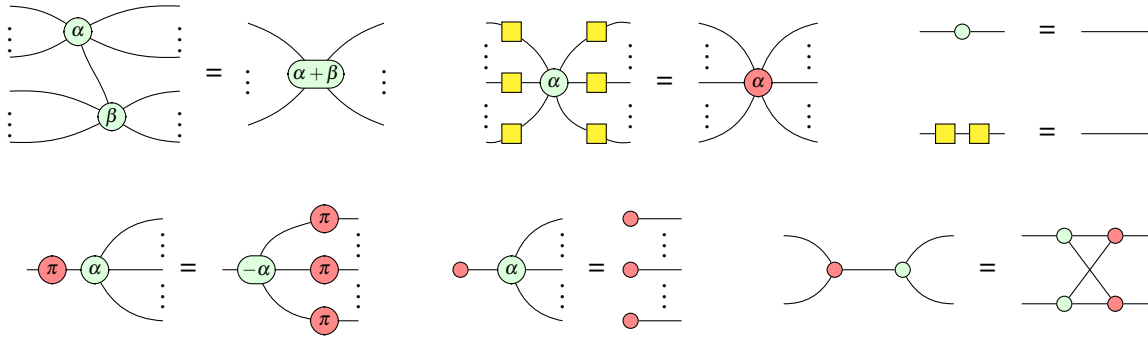


Figure 1: A complete set of rewrite rules for the scalar-free stabilizer ZX-calculus. Each rule also holds with the colours or the directions reversed.

Spiders with exactly one input and output are unitary, in particular $\llbracket -\alpha \rrbracket = |0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1| = Z_\alpha$ and $\llbracket -\alpha \rrbracket = |+\rangle\langle +| + e^{i\alpha}|-\rangle\langle -| = X_\alpha$.

Two diagrams D and D' are said to be equivalent if $\llbracket D \rrbracket = z \llbracket D' \rrbracket$ for some non-zero complex number z . For the rest of the paper, whenever we write a diagram equality we will mean equality up to some global scalar in this way. For treatments of the ZX-calculus which do not ignore scalars see [3] for the stabilizer fragment, [18] for the Clifford+T fragment and [19, 22] for the full ZX-calculus.

The Hadamard gate $H = |+\rangle\langle 0| + |-\rangle\langle 1| \cong Z_{\frac{\pi}{2}} \circ X_{\frac{\pi}{2}} \circ Z_{\frac{\pi}{2}}$ will be used throughout the paper (where \cong denotes equality up to non-zero scalar factor). It has two common syntactic sugars – a yellow square, or a blue dotted line – with the latter only used between spiders:

$$\text{---} \square \text{---} = \text{---} \left(\frac{\pi}{2} \right) \left(\frac{\pi}{2} \right) \left(\frac{\pi}{2} \right) \text{---} \quad \text{---} \circ \text{---} \square \text{---} \circ \text{---} = \text{---} \circ \text{---} \text{---} \circ \text{---}$$

The ZX-calculus is equipped with a set of rewrite rules which can be used to transform a ZX-diagram into another diagram representing the same linear map. As this paper focuses on stabilizer quantum mechanics, we give a rule set for the stabilizer ZX-calculus in Figure 1. Together with the definition of $\text{---} \square \text{---}$, this set of rewrite rules is complete: any two stabilizer ZX-diagrams which correspond (up to non-zero scalar factor) to the same linear map can be rewritten into one another using these rules [2].

2.2 Measurement-based Quantum computation

Measurement-based Quantum computation (MBQC) is a particularly interesting model of quantum computation with no classical analogue. In MBQC, one first constructs a highly entangled resource state that can be independent of the specific computation that one wants to perform (only depending on the ‘size’ of the computation) by preparing qubits in the $|+\rangle$ state and applying CZ-gates to certain pairs of qubits. The computation then proceeds by performing single qubit measurements in a specified order. MBQC is a universal model for quantum computation – any computation can be performed by choosing an appropriate resource state and then performing a certain combination of measurements on said state.

Measurement-based computations are traditionally expressed as *measurement patterns*, which use a sequence of commands to describe how the resource state is constructed and how the computation proceeds [12]. As the resource states are graph states, a graphical representation of MBQC protocols can be more intuitive; we shall therefore introduce MBQC with ZX-diagrams.

Definition 2.1 ([15]). A *graph state diagram* is a ZX-diagram where each vertex is a (phase-free) green spider, each edge connecting spiders has a Hadamard gate on it, and there is a single output wire incident on each vertex. A ZX-diagram is in *graph state with local Clifford (GS-LC) form* if it is a graph state up

operator	$\langle +_{XY,\alpha} _i$	$\langle +_{XZ,\alpha} _i$	$\langle +_{YZ,\alpha} _i$	$\langle +_X,0 _i$	$\langle +_Y,0 _i$	$\langle +_Z,0 _i$	$\langle +_X,\pi _i$	$\langle +_Y,\pi _i$	$\langle +_Z,\pi _i$
diagram									

Table 1: MBQC measurement effects in Dirac notation and their corresponding ZX-diagrams

to single qubit Clifford operators on the input and output wires. It is in *reduced GS-LC (rGS-LC) form* if those single-qubit Clifford operators are all in the set $\{-\frac{k\pi}{2}, -\frac{\pm\pi}{2}, \frac{\pi}{2}\}$ for some $k \in \mathbb{Z}_4$ and if no two qubits with red phases in their vertex operator are connected to each other.

Definition 2.2. [4, Definitions 2.18, 2.23] A ZX-diagram is in *MBQC-form* if it consists of a graph state diagram in which each vertex of the graph may furthermore be connected to an input (in addition to its output), and a measurement effect instead of its output. A ZX-diagram is in *MBQC+LC-form* if it is in MBQC-form up to single qubit Clifford operators on the input and output wires.

MBQC restricts the allowed single-qubit measurements to three planes of the Bloch sphere: those spanned by the eigenstates of two Pauli matrices, called the XY, YZ and XZ planes. Each time a qubit u is measured in a plane $\lambda(u)$ at an angle α , one may obtain either the desired outcome, denoted $\langle +_{\lambda(u),\alpha} |$, or the undesired outcome $\langle -_{\lambda(u),\alpha} | = \langle +_{\lambda(u),\alpha+\pi} |$. Measurements where the angle is an integer multiple of $\frac{\pi}{2}$ are Pauli measurements; the corresponding measurement type is denoted by simply X, Y, or Z. The ZX-diagram corresponding to each (desired) measurement outcome is given in Table 1. The structure of an MBQC protocol is formalised as follows.

Definition 2.3. A *labelled open graph* is a tuple $\Gamma = (G, I, O, \lambda)$, where $G = (V, E)$ is a simple undirected graph, $I \subseteq V$ is a set of input vertices, $O \subseteq V$ is a set of output vertices, and $\lambda : V \setminus O \rightarrow \{X, Y, Z, XY, XZ, YZ\}$ assigns a measurement plane or Pauli measurement to each non-output vertex.

In this paper, we consider *stabilizer MBQC diagrams*: MBQC-form diagrams where every non-output qubit has a Pauli measurement applied to it, i.e. where $\lambda : V \setminus O \rightarrow \{X, Y, Z\}$.

2.3 Pauli flow

Measurement-based computations are inherently probabilistic because measurements are probabilistic. Computations can be made deterministic overall (up to Pauli corrections on the outputs) by tracking which measurements result in undesired outcomes and then correcting for these by adapting future measurements. A sufficient (and in some cases necessary) condition for this to be possible on a given labelled open graph is *Pauli flow*. In the following, $\mathcal{P}(S)$ denotes the powerset of a set S .

Definition 2.4 ([8, Definition 5]). A labelled open graph (G, I, O, λ) has Pauli flow if there exists a map $p : V \setminus O \rightarrow \mathcal{P}(V \setminus I)$ and a partial order \prec over V such that for all $u \in V \setminus O$,

1. if $v \in p(u)$, $v \neq u$ and $\lambda(v) \notin \{X, Y\}$, then $u \prec v$.
2. if $v \in \text{Odd}_G(p(u))$, $v \neq u$ and $\lambda(v) \notin \{Y, Z\}$, then $u \prec v$.
3. if $\neg(u \prec v)$ and $\lambda(v) = Y$, then $v \in p(u) \iff v \in \text{Odd}_G(p(u))$.
4. if $\lambda(u) = XY$, then $u \notin p(u)$ and $u \in \text{Odd}_G(p(u))$.
5. if $\lambda(u) = XZ$, then $u \in p(u)$ and $u \in \text{Odd}_G(p(u))$.
6. if $\lambda(u) = YZ$, then $u \in p(u)$ and $u \notin \text{Odd}_G(p(u))$.
7. if $\lambda(u) = X$, then $u \in \text{Odd}_G(p(u))$.
8. if $\lambda(u) = Z$, then $u \in p(u)$.
9. if $\lambda(u) = Y$ then either $u \in p(u)$ and $u \notin \text{Odd}_G(p(u))$ or $u \notin p(u)$ and $u \in \text{Odd}_G(p(u))$.

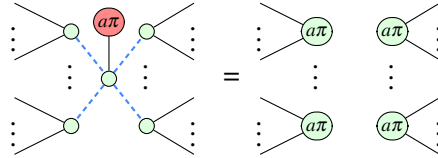
Here, the partial order restricts the time order in which the qubits need to be measured. The set $p(u)$ denotes qubits that are modified by Pauli- X to compensate for an undesired measurement outcome on u , $\text{Odd}_G(p(u))$ denotes the set of vertices that are modified by Pauli- Z .

Pauli flow is a sufficient condition for strong, stepwise and uniform determinism: this means all branches of the computation should implement the same linear operator up to a phase, any interval of the computation should be deterministic on its own, and the computation should be deterministic for all choices of measurement angles that satisfy λ [8, p. 5]. Pauli flow (and related flow conditions) are particularly interesting from a ZX-calculus perspective as there are polynomial-time algorithms for extracting circuits from MBQC-form ZX-diagrams with flow [14, 4, 24], while circuit extraction from general ZX-diagrams is #P-hard [5].

2.4 Existing flow-preserving rewrite rules

The basic ZX-calculus rewrite rules in Figure 1 do not generally preserve even the MBQC-form structure of a ZX-calculus diagram. Yet there are some more complex derived rewrite rules that are known to preserve both the MBQC-form structure and the existence of a flow. These rules were previously considered in the context of gflow [14] and extended gflow [4]; the Pauli-flow preservation proofs are due to [24]. The simplest of these rules is Z-deletion:

Lemma 2.5 ([24, Lemma D.6]). *Deleting a Z-measured vertex preserves the existence of Pauli flow.*



Other rewrite rules are based around quantum generalisations of two graph-theoretic operations.

Definition 2.6. Let $G = (V, E)$ be a graph and $u \in V$. The *local complementation of G about u* is the operation which maps G to $G \star u := (V, E \Delta \{(b, c) | (b, u), (c, u) \in E \text{ and } b \neq c\})$, where Δ is the symmetric difference operator given by $A \Delta B = (A \cup B) \setminus (A \cap B)$. The *pivot of G about the edge (u, v)* is the operation mapping G to the graph $G \wedge uv := G \star u \star v \star u$.

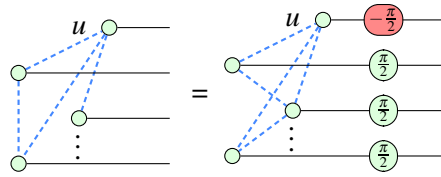
Local complementation keeps the vertices of the graph the same but toggles some edges: for each pair of neighbours of u , i.e. $v, v' \in N_G(u)$, there is an edge connecting v and v' in $G \star u$ if and only if there is no edge connecting v and v' in G . Pivoting is a series of three local complementations, but has some special properties which make it worth distinguishing. It interchanges the vertices u and v and complements (or ‘toggles’) the connectivity between the following three subsets of vertices [6, Section 8]:

- $N_G(u) \setminus (\{v\} \cup N_G(v))$, the neighbours of u that are neither neighbours of v nor v itself.
- $N_G(v) \setminus (\{u\} \cup N_G(u))$, the neighbours of v that are neither neighbours of u nor u itself.
- $N_G(u) \cap N_G(v)$, the common neighbours of u and v .

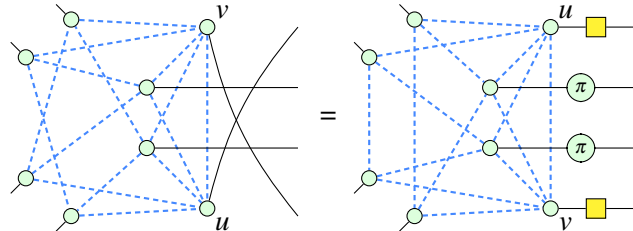
From the above characterisation we see that pivoting is symmetric, i.e. $G \wedge uv = G \wedge vu$.

Both local complementation and pivoting give rise to operations on MBQC-form diagrams which preserve the MBQC form as well as the existence of Pauli flow (after some simple merging of single-qubit Cliffords into measurement effects, cf. [4, Section 4.2]). We illustrate the operations with examples as they are difficult to express in ZX-calculus in generality.

Lemma 2.7 ([24, Lemma D.12]). *A local complementation about a vertex u preserves the existence of Pauli flow.*



Lemma 2.8 ([24, Lemma D.21]). *A pivot about an edge (u,v) preserves the existence of Pauli flow.*



Observation 2.9. *Lemmas 2.7 and 2.8 provide their own inverses since four successive local complementations about the same vertex, or two successive pivots about the same edge, leave the diagram invariant. Two successive local complementations correspond to the π -copy rule.*

While the inverse of the Z-deletion rule of Lemma 2.5 straightforwardly preserves the MBQC-form, it is not obvious that it also preserves the existence of Pauli flow. In Section 4.1, we will prove that this is indeed the case.

3 A canonical form for stabilizer state diagrams

Stabilizer state diagrams in the ZX-calculus have a pseudo-normal form: the rGS-LC form, which arises from the representation of a stabilizer state in terms of a graph state and local Clifford operators [2].

Here, we propose a new pseudo-normal form, based on the representation of a stabilizer state in terms of its affine support and a phase polynomial [1]. Like the rGS-LC form, this is closely related to the stabilizer graphs of Elliott et al. [16] but it translates them into the ZX-calculus differently. The new normal form allows (and in most cases requires) both green and red spiders, meaning it is not strictly ‘graph-like’.

Based on a recent proposal by Hu and Khesin [17], we then show how to make this new pseudo-normal form unique, yielding a canonical form for stabilizer state diagrams in the ZX-calculus¹. In the process, we simplify the uniqueness proof of Hu and Khesin by making use of formalisms and results from the literature about holant problems.

We first prove some lemmas about the algebraic representation of stabilizer states which will be useful in proving uniqueness of the canonical form. Next we introduce to the new pseudo-normal ‘phase polynomial form’ and show how it corresponds to stabilizer states in phase-polynomial representation. Finally, we define the canonical form, prove its uniqueness, and give an algorithm for rewriting diagrams into canonical form. Throughout this section, diagrams contain red spiders and thus are not in MBQC+LC-form; yet by colour changing all of the red vertices and unfusing phases these can straightforwardly be transformed into MBQC+LC-form diagrams.

¹At QCTIP 2022, we learned that an analogous result was independently derived by John van de Wetering [28].

3.1 Stabilizer states in terms of affine support and phase polynomial

It has long been known [13, 26] that an n -qubit stabilizer state can be written (up to normalisation) as

$$\sum_{x \in A} i^{l(x)} (-1)^{q(x)} |x\rangle, \quad (1)$$

where A is an affine subspace of \mathbb{Z}_2^n , $l(x) = \sum_j d_j x_j$ for some fixed $d_j \in \mathbb{Z}_2$ is a linear function computed modulo 2, and $q(x) = \sum_{j < k} c_{jk} x_j x_k + \sum_j c_j x_j$ for some fixed $c_{jk}, c_j \in \mathbb{Z}_2$ is a quadratic function. The functions l and q together form a phase polynomial for the state, while A determines the support.

Assuming $\dim(A) = n - m$, the elements of the affine space A are the solutions to a set of linear equations $Rx = b$, where R is an $m \times n$ binary matrix of rank m (with $0 \leq m \leq n$) and $b \in \mathbb{Z}_2^m$. Each component of x is considered a variable. With respect to this linear system, the variables x_1, \dots, x_n can be partitioned (not generally uniquely) into a set of $(n - m)$ free variables and a set of m dependent variables such that every assignment of values to the free variables induces exactly one assignment of values to the dependent variables which satisfies all the linear equations. This follows from a standard process of solving the system of linear equations, which also yields a linear equation in terms of the free variables for each dependent variable. In the following, we will denote the set of indices by $[n] := \{1, 2, \dots, n\}$ and the free variables by a subset $F \subseteq [n]$ of the indices, and write the dependent variables as $x_j = a_j \oplus \bigoplus_{k \in F} a_{jk} x_k$, where $a_j, a_{jk} \in \mathbb{Z}_2$ and the sum is modulo 2. If $a_{jk} = 1$, we say the variable x_j depends on x_k .

It will be useful to give a canonical choice of free variables, this is inspired by Hu and Khesin's normal form for stabilizer states [17], and will lead us to an analogous normal form for stabilizer diagrams.

Definition 3.1. We call the result of the following procedure the canonical set of free variables. Start with x_1 and consider the variables in ascending order. For each j , if the value of x_j is fixed by the requirement to satisfy $Rx = b$ given values for all free variables among x_1, \dots, x_{j-1} then we say that x_j is dependent. Otherwise we say that x_j is free.

Lemma 3.2. Given an affine space A , the canonical set F is the unique set of free variables with the following property: if x_j depends on the free variable x_k , then $k < j$.

Proof. Let F' be another set of free variables for A which also has the property that if x_j is a dependent variable and depends on the free variable x_k , then $k < j$. In other words, for each $j \in [n] \setminus F'$, there is an equation $x_j = a_j + \sum_{k < j} a_{jk} x_k$, where furthermore $a_{jk} = 0$ if $k \notin F'$.

Now suppose for a contradiction that $F \neq F'$. The two sets must have the same size $|F| = |F'| = \dim(A)$. Thus, there must be a smallest element $j \in F$ such that $j \notin F'$. Then F' induces an equation

$$x_j = a_j \oplus \bigoplus_{k \in F', k < j} a_{jk} x_k. \quad (2)$$

Suppose $a_{jk} = 1$ only if $k \in F$. Then the value of x_j is fixed by the free variables of lower index in F , so j should not be free according to Definition 3.1, a contradiction.

Otherwise, there exists some $k' \notin F$ such that $a_{jk'} = 1$. But then by the definition of F , there exists some equation $x_{k'} = b_{k'} \oplus \bigoplus_{\ell \in F, \ell < k'} b_{k'\ell}$. Thus we can substitute for $x_{k'}$ in (2) while preserving the property that x_j only depends on variables of lower index. The process eliminates one variable which is not in F from the decomposition and does not introduce any new variables which are not in F . Hence repeated application will terminate, at which point we have an equation that fixes x_j from only variables in F of index less than j . Again, this means j should not be in F , a contradiction.

Hence we must have $F = F'$. □

As pointed out in the holant literature, it is possible to express the functions l and q solely in terms of the free variables, while keeping their other properties the same [9, Definition 8]. We give a proof in Appendix A for completeness.

Lemma 3.3. *Suppose F denotes a set of free variables for the affine space A , and $|\psi\rangle$ is some stabilizer state with support on A . Then there exists a linear function l and a quadratic function q , both depending only on the free variables, as well as a scalar $\lambda \in \mathbb{C} \setminus \{0\}$, such that:*

$$|\psi\rangle = \lambda \sum_{x \in A} i^{l(x)} (-1)^{q(x)} |x\rangle.$$

There are generally multiple ways of expressing the same state in the form of (1). Yet if we pick a set of free variables F and require l and q to depend only on free variables, the representation becomes unique. Moreover, we can even give a unique representation in terms of a phase polynomial (evaluated modulo 4, rather than 2). Again, the proof is in Appendix A.

Lemma 3.4. *Given an n -qubit stabilizer state $|\psi\rangle$ and a set $F \subseteq [n]$, there exists a unique polynomial $p(x) = \sum_{j \in F} r_j x_j + 2 \sum_{j, k \in F, j < k} s_{jk} x_j x_k$ with $r_j \in \mathbb{Z}_4$ and $s_{jk} \in \mathbb{Z}_2$ and scalar $\lambda \in \mathbb{C} \setminus \{0\}$ such that $|\psi\rangle = \lambda \sum_{x \in A} i^{p(x)} |x\rangle$.*

3.2 A new pseudo-normal form related to phase polynomials

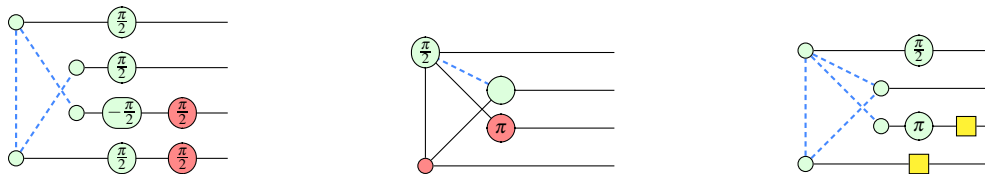
In the rGS-LC form for stabilizer state diagrams, local Clifford operators on the graph state are expressed in terms of green and red spiders. Alternatively, it is also possible to express local Clifford operators in terms of green spiders and Hadamards (and this is what is done in the stabilizer graph formalism of [16]). In ZX-terms, this means the allowed local Clifford operators are $-\frac{k\pi}{2}$ and $-\pi$ \square , where $k \in \mathbb{Z}_4$ and $a \in \mathbb{Z}_2$. As for red nodes in rGS-LC diagrams, qubits whose local Clifford operator contains an H are not allowed to be connected to each other; therefore we can ‘push’ the Hadamards through and get the following pseudo-normal form. It is possible to convert between the two kinds of local Clifford operators via local complementations on the qubits that have red nodes or Hadamards.

Definition 3.5. A stabilizer ZX-calculus diagram is in phase-polynomial form if the following hold:

- Each dangling edge is connected to a unique red or green spider.
- Red spiders have phases that are 0 or π .
- Green spiders have phases that are integer multiples of $\pi/2$.
- There may be edges connecting spiders of different colours.
- Furthermore, green spiders may be connected to other green spiders via Hadamard nodes.

Observation 3.6. *An rGS-LC diagram can be brought into phase-polynomial form via the following process. First, apply local complementations to all qubits that have red nodes in their local Cliffords. This maps $-\frac{\pi}{2}$ \circ $\frac{\pi}{2}$ to \square and $-\frac{\pi}{2}$ \circ $\frac{\pi}{2}$ to \square \circ π . Then, change the colour of all spiders which now have Hadamards as part of their vertex operators and merge adjacent spiders of the same colour.*

Example 3.7. Applying this procedure to the rGS-LC diagram on the left yields the phase polynomial-form diagram in the middle. Colour-changing each red spider and unfixing the phases leads to an equivalent GS-LC form diagram which we will say is in phase-polynomial form up to colour changing the spiders with Hadamard gates in their vertex operators.



Diagrams in phase-polynomial form correspond directly to pairs of a state and a set of free variables for the underlying affine support. Appendix B contains an example illustrating this correspondence.

Lemma 3.8. *Ignoring scaling, there is a bijection between phase-polynomial form diagrams and pairs $(|\psi\rangle, F)$, where $|\psi\rangle$ is an n -qubit stabilizer state and $F \subseteq [n]$ indicates a set of free variables for the affine space A which is the support of $|\psi\rangle$.*

Proof. By Lemma 3.4, there exists a unique function $p(x) = \sum_{j \in F} r_j x_j + 2 \sum_{j, k \in F, j < k} s_{jk} x_j x_k$ with $r_j \in \mathbb{Z}_4$ and $s_{jk} \in \mathbb{Z}_2$ such that $|\psi\rangle \cong \sum_{x \in A} i^{p(x)} |x\rangle$. To construct a diagram from a state and a set of free variables from this, proceed as follows:

- For each dependent variable x_k with $k \in [n] \setminus F$, find the unique linear expression $x_k = a_k \oplus \bigoplus_{j \in F} a_{kj} x_j$ which satisfies the defining linear equations $Rx = b$ of the affine space A .
- For each $j \in F$, place a green spider with an output wire. The phase of this spider is $r_j \frac{\pi}{2}$.
- For each $k \in [n] \setminus F$, place a red spider with an output wire. The phase of this spider is $a_j \pi$.
- Draw a (plain) edge connecting the green spider j to the red spider k whenever $a_{kj} = 1$.
- Draw a Hadamard edge connecting the green spiders j and j' whenever $s_{jj'} = 1$.

Conversely, given a diagram in phase-polynomial form, construct the corresponding state as below:

- The set F of free variables consists of the indices of the green spiders.
- The affine space A is defined by the set of equations $\left\{ x_j = a_j \oplus \bigoplus_{k \in N(j)} x_k \right\}_{j \in [n] \setminus F}$, where $a_j = 0$ if the phase of the red spider with index j is 0, and 1 otherwise.
- For each $j \in F$ such that the phase of the green spider j is α_j , define r_j to be the value in \mathbb{Z}_4 that is equivalent to $\frac{2\alpha_j}{\pi} \pmod{4}$.
- For each $j, k \in F$ with $j < k$, define $s_{jk} = 1$ if there exists a Hadamard edge between spiders j and k , and $s_{jk} = 0$ otherwise.

Let $p(x) := \sum_{j \in F} r_j x_j + 2 \sum_{j, k \in F, j < k} s_{jk} x_j x_k$, then the desired state is $\sum_{x \in A} i^{p(x)} |x\rangle$. The two procedures are inverses of each other (noting that $\frac{3\pi}{2} \equiv -\frac{\pi}{2} \pmod{2\pi}$).

Suppose D is the ZX-diagram corresponding to some stabilizer state $|\psi\rangle$ according to the above translation. Then it is straightforward to see that the support of $\llbracket D \rrbracket$ and the support of $|\psi\rangle$ are equal. Thus, by phase-polynomial techniques, it is quick to check that $\llbracket D \rrbracket$ equals $|\psi\rangle$ up to scalar factor. \square

3.3 The canonical phase-polynomial diagram

Using the bijection between phase-polynomial form diagrams and pairs of a state and a set of free variables, we can now define a unique canonical diagram for any stabilizer state.

Definition 3.9. Let $|\psi\rangle$ be a stabilizer state, then its canonical diagram is the one translated from $(|\psi\rangle, F)$ by Lemma 3.8, where F is the canonical set of free variables according to Definition 3.1.

Apart from the translation into our terminology, this differs from the normal form definition of Hu and Khesin [17] only by reversing the order: we ask for free variables to come first whereas they put them last. Our uniqueness proof, making use of the properties of the affine support of a stabilizer state is shorter and simpler than that in [17].

Theorem 3.10. *The canonical form is unique.*

Proof. This follows from the uniqueness of the canonical set of free variables proved in Lemma 3.2 and from the bijection between pairs consisting of a state and a set of free variables in Lemma 3.8. \square

Proposition 3.11. *Every phase-polynomial form diagram can be re-written into canonical form using only local complementation and pivoting.*

Proof. Pick some order $<$ on the spiders, say from top to bottom. We want each red spider to only be connected to spiders that appear earlier in $<$. While this does not hold, repeat the following procedure:

1. Let d_k be the minimal red spider under $<$ such that there exists some green spider f_j connected to d_k with $d_k < f_j$.
2. Let f_h be the maximal green spider under $<$ such that d_k is connected to f_h .
3. If f_h has a phase of $\pm\frac{\pi}{2}$, perform local complementation about f_h and then about d_k . Otherwise, pivot about the edge connecting f_h and d_k . After applying either of these equivalence transformations, f_h is now red and d_k is now green and the diagram is still in phase-polynomial form.
4. By maximality of f_h , we have that f_h is only connected to green spiders f_n with $f_n < f_h$. By minimality of d_k , we have that d_k is only connected to red spiders d_m with $d_k < d_m$.

This procedure strictly reduces the number of connections between red spiders and green spiders that appear later in the order. Hence repeating it will eventually terminate, transforming any phase-polynomial form diagram into canonical form. \square

Remark 3.12. The canonical form is unique only up to the choice of order on the qubits; different orders may yield different ‘canonical forms’. Thus the choice of order is arbitrary (but needs to happen in advance, independently of the diagram considered) – we have chosen top-to-bottom for simplicity.

4 Completeness

Having established a canonical form for stabilizer ZX-calculus diagrams, we now give the completeness proof. This first requires proving that a new rewrite rule preserves the existence of Pauli flow: an inverse to the Z-deletion rule of Lemma 2.5. While there has been a lot of previous research on rewrite rules which reduce the number of spiders while preserving flow conditions, rewrite rules which increase the number of spiders have not been studied beyond introducing new degree-2 vertices along input or output wires (e.g. [4, Lemma 4.1]).

4.1 Inserting new Z-measured qubits

Inserting Z-measured qubits into MBQC+LC form diagram preserves the existence of Pauli flow.

Proposition 4.1. *Let $G = (V, E, I, O, \lambda)$ be a labelled open graph with Pauli flow and let $W \subseteq V$ be some arbitrary subset of the vertices. Then $G' = (V', E', I, O, \lambda')$ has a Pauli flow, where $V' = V \cup \{x\}$, $E' = E \cup \{(x, w) \mid w \in W\}$ with $\lambda'(v) = \lambda(v)$ if $v \neq x$ and $\lambda'(x) = Z$.*

Proof. Let (p, \prec) be a Pauli flow for G and define $p' : V' \setminus O \rightarrow \mathcal{P}(V' \setminus I)$ by $p'(v) := p(v)$ if $v \neq x$ and $p'(x) := \{x\}$. For vertices from the original graph, measurement planes and correction sets remain the same while the only change to odd neighbourhoods is that x may be added. Thus conditions 4–7 and 9 remain trivially satisfied. Condition 8 holds for x as $x \in p'(x)$, and for all other Z-measured vertices because (p, \prec) is a Pauli flow.

Let \prec' be the transitive closure of $\prec \cup \{(x, v) \mid v \in N_G(x)\}$. Then \prec' is a partial order because \prec is a partial order and we only add successors for x . Now, condition 1 of Pauli flow is inherited from (p, \prec) for all $u \in V \setminus O$ because $u \notin p'(x)$. Condition 2 is satisfied for all $u \in V \setminus O$ because $\lambda(x) = Z$ and (p, \prec) is a Pauli flow. Condition 3 is inherited because the new vertex has only successors. \square

4.2 Complete flow-preserving rewrite rules

We are now able to assemble the main proof. In the following, we will say an MBQC+LC-form diagram has *no interior spiders* if the MBQC-form part of the diagram (i.e. ignoring the local Cliffords) has no interior vertices ($V \setminus (I \cup O) = \emptyset$). Additionally, we say an MBQC+LC-form diagram has Pauli flow if its MBQC-form part has Pauli flow (analogous to gflow in [4, Section 4.1]).

Theorem 4.2. *Given two equivalent stabilizer MBQC+LC-form diagrams D and D' with Pauli flow and satisfying $\llbracket D \rrbracket \cong \llbracket D' \rrbracket$, there exists a sequence of rewrite rules – each preserving the existence of Pauli flow and preserving the MBQC+LC-form – transforming D into D' .*

Proof. We begin by deleting all Z -measured vertices from both diagrams, keeping track of which vertices we delete and their set of neighbours when deleted. The resulting diagrams has Pauli flow by Lemma 2.5. After all Z -measured vertices are removed, the MBQC-form parts of the diagrams (ignoring the local Cliffords) only have X and Y measurements and are thus of the kind considered in [14]. Then, there exists a terminating procedure (consisting of a sequence of local complementations, pivots and Z -deletions) rewriting the two diagrams into MBQC+LC-form diagrams N and N' which contain no interior spiders [14, Theorem 5.4]. Since local complementation and pivoting also preserve the existence of Pauli flow (Lemmas 2.7 and 2.8), N and N' will also have Pauli flow.

As only X and Y measurements remain, they can be spider-merged and unmerged through each qubit to become local Cliffords on the outputs, thus N and N' are equivalent to GS-LC form diagrams. By [2, Theorem 13], every GS-LC form diagram can be rewritten into rGS-LC form using a sequence of local complementations, thus this step preserves Pauli flow. By Observation 3.6, we can then rewrite each diagram into phase polynomial form, again using only local complementations (along with some operations on the local Cliffords that do not alter the flow), thus preserving Pauli flow. Finally, by Proposition 3.11, we can rewrite each diagram into canonical form¹. The rewrite steps use only local complementations and pivoting, so they preserve Pauli flow. The resulting diagrams are equivalent and the canonical form is unique, so we have found a sequence of local complementations, pivots and Z -deletions rewriting D and D' into the same canonical form diagram C .

By Observation 2.9, local complementation and pivot can be inverted. Furthermore, Z -insert is a Pauli-flow preserving inverse to Z -delete. Thus the sequence of rewrites from D' to C can be inverted while still preserving Pauli-flow. By rewriting D to C , then rewriting C to D' , we obtain a sequence of flow-preserving rewrite rules transforming D into D' . This completes the proof. \square

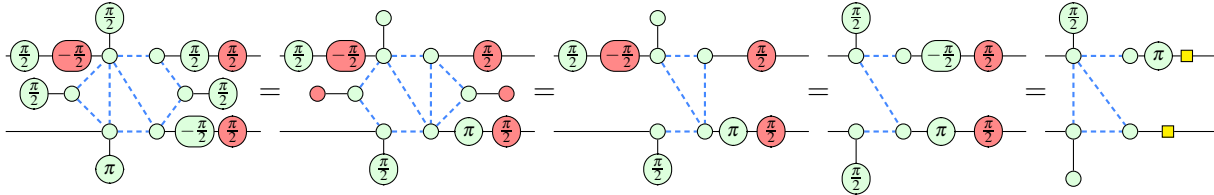
Example 4.3. We shall give a short example of this rewrite procedure in action. Consider the following two MBQC+LC-form diagrams, which we will call D and D' , and which satisfy $\llbracket D \rrbracket \cong \llbracket D' \rrbracket$ by (non-flow preserving) diagram simplification techniques.



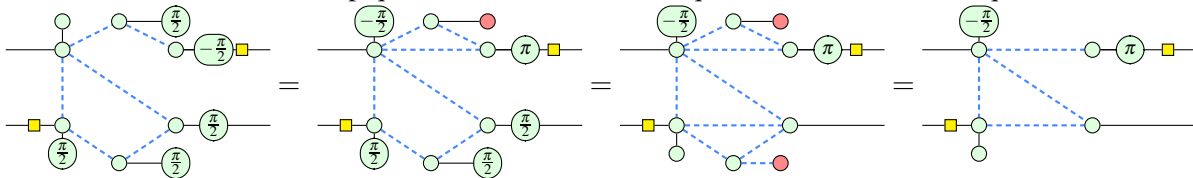
Using the procedure from the proof of Theorem 4.2, we first rewrite D to phase polynomial form. Perform triple local complementations (i.e. ‘inverse local complementations’) about both the left-most and right-most qubits in the MBQC-form part, then apply Z -deletion to these qubits. A local complementation

¹Up to map-state duality and colour changing vertices with Hadamard operators.

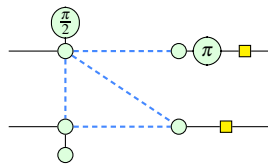
about the top left qubit gives us the fourth diagram, which is in rGS-LC form and in fact is equivalent to the left-most diagram in Example 3.7 up to map-state duality. We then obtain the final diagram by following the procedure in Observation 3.6; note that this diagram is already in canonical form (up to map-state duality and colour changing spiders with Hadamard gates in their vertex operators) assuming that the input qubits have lower indices than the output qubits.



For D' , we perform local complementation about the two interior qubits of the MBQC-form part (here we have done this about the top qubit first, then the bottom qubit), and Z-delete both qubits.



This final diagram is already in phase polynomial form (up to map state duality and colour changing the spiders with Hadamard edges in their vertex operators) without us having to go through rGS-LC form. To rewrite this diagram into canonical form, all that remains is to pivot along the edge connecting the bottom left qubit to the bottom right qubit, giving the following diagram:



We have therefore rewritten D and D' into the same canonical form diagram. Every rule used to re-write D and D' to canonical form is invertible and the inverses preserve Pauli flow, giving us a sequence of flow preserving rewrite rules taking D to D' .

5 Conclusions

We have presented the first flow-preserving rewrite rule that increases the number of qubits in an MBQC-form ZX-diagram, and shown that this – together with existing rewrite rules that preserve the MBQC form – is complete for stabilizer MBQC-form diagrams. The completeness proof goes via a new canonical form. The result may find applications in obfuscation or in more involved optimisation protocols.

Yet that is only the beginning of the investigation of flow-preserving rewrite rules and in future work we will consider more extensive sets of rewrite rules and ZX-diagrams. The recent proof that circuit extraction from general unitary ZX-diagrams is #P-hard [5] means this line of research is particularly important, as it allows us to explore the only family of ZX-diagrams for which a polynomial-time circuit-extraction algorithm is currently known.

Pauli flow is known not to be necessary for deterministic implementability of MBQC patterns with all-Pauli measurements [8]; it would also be interesting to see how it can be extended and what flow-preserving rewriting would look like under the new conditions.

Acknowledgements Thanks to Hex Miller-Bakewell for helpful comments on earlier notes about the phase-polynomial form.

References

- [1] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T -Depth Optimization of Clifford+ T Circuits Via Matroid Partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953.
- [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 (2015): *Making the stabilizer ZX-calculus complete for scalars*. *Electronic Proceedings in Theoretical Computer Science* 195, p. 17–32, doi:10.4204/eptcs.195.2.
- [4] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and back again: A circuit extraction tale*. *Quantum* 5, p. 421, doi:10.22331/q-2021-03-25-421.
- [5] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-diagrams can be #P-hard*, doi:10.48550/ARXIV.2202.09194. Available at <https://arxiv.org/abs/2202.09194>.
- [6] André Bouchet (1987): *Graphic Presentations of Isotropic Systems*. *Journal of Combinatorial Theory, Series B* 45(1), p. 58–76, doi:10.1016/0095-8956(88)90055-X.
- [7] Anne Broadbent & Elham Kashefi (2009): *Parallelizing quantum circuits*. *Theoretical Computer Science* 410(26), pp. 2489–2510, doi:10.1016/j.tcs.2008.12.046.
- [8] 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. 250–250, doi:10.1088/1367-2630/9/8/250.
- [9] Jin-Yi Cai, Pinyan Lu & Mingji Xia (2018): *Dichotomy for Real Holant^c Problems*. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, pp. 1802–1821, doi:10.1137/1.9781611975031.118.
- [10] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.
- [11] Vincent Danos & Elham Kashefi (2006): *Determinism in the one-way model*. *Phys. Rev. A* 74, p. 052310, doi:10.1103/PhysRevA.74.052310.
- [12] Vincent Danos, Elham Kashefi & Prakash Panangaden (2005): *Parsimonious and robust realizations of unitary maps in the one-way model*. *Physical Review A* 72(6), p. 064301, doi:10.1103/PhysRevA.72.064301.
- [13] Jeroen Dehaene & Bart De Moor (2003): *Clifford group, stabilizer states, and linear and quadratic operations over GF(2)*. *Phys. Rev. A* 68, p. 042318, doi:10.1103/PhysRevA.68.042318.
- [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] Ross Duncan & Simon Perdrix (2009): *Graph States and the Necessity of Euler Decomposition*. In Klaus Ambos-Spies, Benedikt Löwe & Wolfgang Merkle, editors: *Mathematical Theory and Computational Practice*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 167–177, doi:10.1007/978-3-642-03073-4_18.
- [16] Matthew B. Elliott, Bryan Eastin & Carlton M. Caves (2008): *Graphical description of the action of Clifford operators on stabilizer states*. *Phys. Rev. A* 77, p. 042307, doi:10.1103/PhysRevA.77.042307.
- [17] Alexander Tianlin Hu & Andrey Boris Khesin (2022): *Improved graph formalism for quantum circuit simulation*. *Phys. Rev. A* 105, p. 022432, doi:10.1103/PhysRevA.105.022432.
- [18] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+ T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic*

- in *Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, p. 559–568, doi:10.1145/3209108.3209131.
- [19] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, Association for Computing Machinery, New York, NY, USA, p. 569–578, doi:10.1145/3209108.3209139.
- [20] Aleks Kissinger & 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.
- [21] Jisho Miyazaki, Michal Hajdušek & Mio Muraao (2015): *Analysis of the trade-off between spatial and temporal resources for measurement-based quantum computation*. *Physical Review A* 91(5), p. 052302, doi:10.1103/PhysRevA.91.052302.
- [22] Kang Feng Ng & Quanlong Wang (2017): *A universal completion of the ZX-calculus*, doi:10.48550/arXiv.1706.09877.
- [23] Robert Raussendorf & Hans J. Briegel (2001): *A One-Way Quantum Computer*. *Phys. Rev. Lett.* 86, pp. 5188–5191, doi:10.1103/PhysRevLett.86.5188.
- [24] Will Simmons (2021): *Relating Measurement Patterns to Circuits via Pauli Flow*. In Chris Heunen & Miriam Backens, editors: *Proceedings 18th International Conference on Quantum Physics and Logic*, Gdansk, Poland, and online, 7-11 June 2021, *Electronic Proceedings in Theoretical Computer Science* 343, Open Publishing Association, pp. 50–101, doi:10.4204/EPTCS.343.4.
- [25] Korbini Staudacher (2021): *Optimization Approaches for Quantum Circuits using ZX-calculus*. Master's thesis, Ludwig-Maximilians-Universität, München. Available at <https://www.mnm-team.org/pub/Diplomarbeiten/stau21/PDF-Version/stau21.pdf>.
- [26] Maarten Van Den Nest (2010): *Classical Simulation of Quantum Computation, the Gottesman-Knill Theorem, and Slightly Beyond*. *Quantum Info. Comput.* 10(3), p. 258–271, doi:10.5555/2011350.2011356.
- [27] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*, doi:10.48550/ARXIV.2012.13966.
- [28] John van de Wetering (2022): *Personal communication*.

A Algebraic proofs for the canonical form

Proof of Lemma 3.3. In (1), the functions l and q are allowed to depend on all components of the bit string x , i.e. $l(x) = \bigoplus_j d_j x_j$ for some fixed $d_j \in \mathbb{Z}_2$ and $q(x) = \bigoplus_{j < k} c_{jk} x_j x_k \oplus \bigoplus_j c_j x_j$ for some fixed $c_{jk}, c_j \in \mathbb{Z}_2$.

Given the set of free variables F , solving the defining system of linear equations for A yields linear equations $x_j = a_j \oplus \bigoplus_{k \in F} a_{jk} x_k$ for every $j \in [n] \setminus F$, where $a_j, a_{jk} \in \mathbb{Z}_2$.

Now suppose $d_j \neq 0$ for some $j \notin F$. Then we can substitute

$$l(x) = \bigoplus_{j \in [n]} d_j x_j = \left(\bigoplus_{j \in [n] \setminus \{s\}} d_j x_j \right) \oplus a_s \oplus \bigoplus_{t \in F} a_{st} x_t = a_s \oplus \bigoplus_{j \in [n] \setminus \{s\}} (d_j \oplus a_{sj}) x_j,$$

where we define $a_{sj} = 0$ if $j \notin F$. The a_s is constant and the factor i^{a_s} can be absorbed into the overall scalar λ . Since l is computed modulo 2, the new function satisfies the same properties as the original one but no longer depends on x_s . Furthermore, as $a_{sj} = 0$ for all $j \notin F$, this process does not introduce any new dependencies on dependent variables.

Therefore, the substitution process strictly decreases the number of dependent variables that l depends on and successive applications will eventually yield a function that depends only on free variables. An analogous argument holds for q . \square

Lemma A.1. *Let $|\psi\rangle$ and $|\phi\rangle$ be two stabilizer states with the same support A , and let F be a set of free variables for A . Suppose there exists $\lambda, \mu \in \mathbb{C} \setminus \{0\}$ such that*

$$|\psi\rangle = \lambda \sum_{x \in A} i^{l(x)} (-1)^{q(x)} |x\rangle \quad \text{and} \quad |\phi\rangle = \mu \sum_{x \in A} i^{l'(x)} (-1)^{q'(x)} |x\rangle$$

where for some $d_j, d'_j, c_{jk}, c_j, c'_{jk}, c'_j \in \mathbb{Z}_2$,

$$\begin{aligned} l(x) &= \bigoplus_{j \in F} d_j x_j & q(x) &= \bigoplus_{j,k \in F, j < k} c_{jk} x_j x_k \oplus \bigoplus_j c_j x_j \\ l'(x) &= \bigoplus_{j \in F} d'_j x_j & q'(x) &= \bigoplus_{j,k \in F, j < k} c'_{jk} x_j x_k \oplus \bigoplus_j c'_j x_j. \end{aligned}$$

Then $|\psi\rangle$ and $|\phi\rangle$ are linearly dependent if and only if for all $j, k \in F$ we have $d_j = d'_j$, $c_{jk} = c'_{jk}$, and $c_j = c'_j$.

Proof. The ‘if’ direction is straightforward: if $d_j = d'_j$, $c_{jk} = c'_{jk}$, and $c_j = c'_j$ for all $j, k \in F$, then $\mu |\psi\rangle = \lambda |\phi\rangle$.

For the ‘only if’ direction, note that $l(x) = l'(x) = q(x) = q'(x) = 0$ if all variables in F are assigned 0, so by rescaling such that $\lambda = \mu$, we get $|\psi\rangle = |\phi\rangle$ if and only if they are linearly dependent.

By definition, each assignment of values to the free variables in F induces one assignment of values to all the variables that is in A . Suppose there exists a $j \in F$ such that $d_j \neq d'_j$, wlog assume $d_j = 1$ and $d'_j = 0$ (otherwise the argument is symmetric). Let ξ be the bit string in A that has every free variable set to 0 except the one with index j . Then $\langle \xi | \psi \rangle$ is imaginary while $\langle \xi | \phi \rangle$ is real, so since the two states have the same non-zero amplitude for the assignment induced by setting all free variables to 0, they cannot be linearly dependent.

Similarly, suppose there exists $j \in F$ such that $c_j \neq c'_j$, then for the same ξ we have $\langle \xi | \psi \rangle = -\langle \xi | \phi \rangle$, so again the two states cannot be linearly dependent.

So without loss of generality, assume that $d_j = d'_j$ and $c_j = c'_j$ for all $j \in F$. Now suppose there are $j, k \in F$ such that $c_{jk} \neq c'_{jk}$. Let ζ be the bit string induced by the assignment where $x_j = x_k = 1$ and all other free variables are 0. Then again, $\langle \zeta | \psi \rangle = -\langle \zeta | \phi \rangle$ so the two states cannot be linearly dependent.

Therefore, linear dependence implies that for all $j, k \in F$ we have $d_j = d'_j$, $c_{jk} = c'_{jk}$, and $c_j = c'_j$. \square

Proof of Lemma 3.4. Via Lemmas 3.3 and A.1, we can uniquely write $|\psi\rangle = \lambda \sum_{x \in A} i^{l(x)} (-1)^{q(x)} |x\rangle$, where $l(x) = \bigoplus_{j \in F} d_j x_j$ and $q(x) = \bigoplus_{j,k \in F, j < k} c_{jk} x_j x_k \oplus \bigoplus_j c_j x_j$ with all coefficients taking values in \mathbb{Z}_2 .

As $y \bmod 2 = y^2 \bmod 4$ for all $y \in \mathbb{Z}$, we have

$$\bigoplus_{j \in F} d_j x_j = \left(\sum_{j \in F} d_j x_j \right)^2 \bmod 4 = \left(\sum_{j \in F} d_j x_j + 2 \sum_{j,k \in F, j < k} d_j d_k x_j x_k \right) \bmod 4,$$

where we have used the fact that $d_j, x_j \in \mathbb{Z}_2$ for all j and hence $(d_j x_j)^2 = d_j x_j$. We can thus write

$$\sum_{x \in A} i^{l(x)} (-1)^{q(x)} |x\rangle = \sum_{x \in A} i^{p(x)} |x\rangle,$$

where

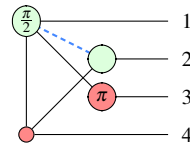
$$\begin{aligned} p(x) &= \sum_{j \in F} d_j x_j + 2 \left(\sum_{j,k \in F, j < k} c_{jk} x_j x_k + \sum_j c_j x_j \right) + 2 \sum_{j,k \in F, j < k} d_j d_k x_j x_k \\ &= \sum_{j \in F} (d_j + 2c_j) x_j + 2 \sum_{j,k \in F, j < k} (c_{jk} + d_j d_k) x_j x_k \end{aligned}$$

Now, $r_j := (d_j + 2c_j) \in \mathbb{Z}_4$. The coefficient $s_{jk} := c_{jk} + d_j d_k$ could take value 2, but as p is in the exponent of i and s_{jk} is multiplied by 2, we may without loss of generality replace it with $s_{jk} := c_{jk} \oplus d_j d_k$ so that $s_{jk} \in \mathbb{Z}_2$.

Conversely, we can find functions l and q from p by setting $d_j := r_j \bmod 2$, $c_j := \frac{1}{2}(r_j - d_j)$, and $c_{jk} := s_{jk} \oplus d_j d_k$. Thus, by uniqueness of l and q , the phase polynomial expression is also unique. \square

B An example illustrating Lemma 3.8

Consider the following phase-polynomial form diagram from Example 3.7, where we have numbered the qubits from top to bottom.



Following the procedure from Lemma 3.8, we construct the state corresponding to this diagram. The state will be expressed as $\sum_{x \in A} i^{p(x)} |x\rangle$, where $p(x) = \sum_{j \in F} r_j x_j + 2 \sum_{j,k \in F, j < k} s_{jk} x_j x_k$. Here, F is the set of free variables, A is the affine space on which the state has support, and $p(x)$ is the phase polynomial with $r_j \in \mathbb{Z}_4$ and $s_{jk} \in \mathbb{Z}_2$ for all $j, k \in F$.

- The set F of free variables corresponding to this diagram is $F = \{x_1, x_2\}$ since qubits 1 and 2 are denoted by green spiders.
- The affine space A is defined by the following set of equations arising from the red spiders:

$$x_3 = 1 \oplus x_1 \qquad x_4 = x_1 \oplus x_2 \qquad (3)$$

since qubit 3 has phase π (giving the constant 1 on the right-hand side) and is connected to qubit 1, while qubit 4 has phase 0 and is connected to both 1 and 2.

- For the linear terms in the phase polynomial, we get that $r_1 = 1$ and $r_2 = 0$ as the phase of x_1 is $\frac{\pi}{2}$ and the phase of x_2 is 0.
- For the quadratic terms in the phase polynomial, we have $s_{12} = 1$ as there is a Hadamard edge connecting x_1 and x_2 .

Combining these, the phase polynomial is $p(x) = x_1 + 2x_1 x_2$. The state corresponding to the diagram is therefore given by:

$$\begin{aligned} \sum_{x \in A} i^{x_1 + 2x_1 x_2} |x\rangle &= \sum_{x_1, x_2 \in \mathbb{Z}_2} i^{x_1} (-1)^{x_1 x_2} |x_1 x_2 (1 \oplus x_1)(x_1 \oplus x_2)\rangle \\ &= |0010\rangle + |0111\rangle + i|1001\rangle - i|1100\rangle \end{aligned}$$

It is then quick to check that applying the procedure in Lemma 3.8 for constructing a diagram from a state and a set of free variables gives back the original diagram.

Instead, we will show how to construct the diagram corresponding to the same state with a different set of free variables $F = \{x_2, x_3\}$. To do this, we first rewrite the affine space and the phase polynomial in terms of the new free variables x_2 and x_3 , and then apply the procedure for obtaining diagrams.

Choosing x_3 to be free instead of x_1 , we rearrange the first equation of (3) and then substitute it into the second to get:

$$x_1 = 1 \oplus x_3 \qquad x_4 = 1 \oplus x_2 \oplus x_3 \qquad (4)$$

Substituting into the phase polynomial yields $p(x) = (1 \oplus x_3) + 2(1 \oplus x_3)x_2$ where \oplus denotes addition modulo 2. Yet we want the phase polynomial to be computed modulo 4, since $i^4 = 1$. Now, as $y \bmod 2 = y^2 \bmod 4$ for all $y \in \mathbb{Z}$, and $b^2 = b$ for all $b \in \mathbb{Z}_2$, this can be rewritten to:

$$p(x) = (1 \oplus x_3) + 2(1 \oplus x_3)x_2 = (1 + x_3)^2 + 2(1 + x_3)^2 x_2 = 1 + 2x_2 + 3x_3 + 2x_2x_3 \pmod{4}$$

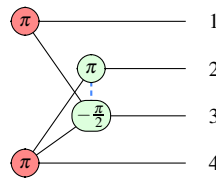
We thus have $r_2 = 2$, $r_3 = 3$, and $s_{23} = 1$. The constant term in the phase polynomial is irrelevant since we are ignoring global scalars. Up to scalar factor, the full state is

$$\sum_{x_2, x_3 \in \mathbb{Z}_2} i^{2x_2 + 3x_3 + 2x_2x_3} |(1 \oplus x_3)x_2x_3(1 \oplus x_2 \oplus x_3)\rangle.$$

To construct the diagram corresponding to this state and set of free variables:

- We already have the equations for the dependent variables in terms of $F = \{x_2, x_3\}$ in (4).
- Place a green spider with phase $r_2 \frac{\pi}{2} = \pi$ for qubit 2 and a green spider with phase $r_3 \frac{\pi}{2} = \frac{3\pi}{2}$ (or, equivalently, $-\frac{\pi}{2}$) for qubit 3. Each of the spiders is connected to one output wire.
- Place a red spider with phase π for qubit 1 and a red spider with phase π for qubit 4 since the equations for both x_1 and x_4 contain a constant term. Again, each of the spiders is connected to one output wire.
- Variable x_1 depends on x_3 , so draw a plain wire between the spiders for qubits 1 and 3. Variable x_4 depends on both x_2 and x_3 , so draw plain wires between the spiders for qubits 2 and 4, as well as between 3 and 4.
- As $s_{23} = 1$, draw a Hadamard edge connecting the green spiders corresponding to x_2 and x_3 .

This yields the following diagram:



Quantum Linear Optics via String Diagrams

Giovanni de Felice and Bob Coecke

Quantinuum – Quantum Compositional Intelligence
17 Beaumont street, OX1 2NA Oxford, UK

We establish a formal bridge between qubit-based and photonic quantum computing. We do this by defining a functor from the ZX calculus to linear optical circuits. In the process we provide a compositional theory of quantum linear optics which allows to reason about events involving multiple photons such as those required to perform linear-optical and fusion-based quantum computing.

1 Introduction

Quantum optics has pioneered experimental tests of entanglement [1], nonlocality [2], teleportation [3], quantum-key distribution [4], and quantum advantage [5]. These experiments ultimately rely on the ability to process coherent states of photons in *linear optical* devices, an intractable task for classical computers [6]. Recently, the potential of using linear optics for quantum computing has encouraged the development of both hardware [7] and software [8, 9] for photonic technologies. The first proposal was formulated by Knill, Laflamme and Millburn in 2001 [10]. Qubits are encoded in pairs of optical modes and quantum computing may be performed using only linear optical elements and photon detectors. Several improvements to the original scheme have been proposed in the literature [11, 12, 13]. Fusion measurements were introduced by Browne and Rudolf [14]. They form the basic ingredient of a recent proposal to achieve fault-tolerant quantum computation with photonic qubits [15].

String diagrams provide an intuitive language for quantum processes [16, 17, 18, 19, 20] and are implicitly employed in quantum software packages such as tket [21], PyZX [22], lambeq [23], DisCoPy [24], Quanhoven [25]. On the one hand, Coecke and Duncan [26] introduced the ZX calculus, a graphical language for reasoning about qubit quantum computing, with applications in circuit-based [27], measurement-based [28], and fault tolerant [29] quantum computing. The axioms of this calculus feature a bialgebra structure governing the Z and X qubit bases. On the other hand, Vicary and Fiore used the symmetric (or bosonic) Fock space to study the quantum harmonic oscillator, and discovered a different bialgebra structure on this infinite dimensional Hilbert space [30, 31]. These two foundational works are hardly ever related in the literature, possibly because of the difference in state space cardinality. However, it is well-known that photons in linear optics behave as quantum harmonic oscillators. Given the developments in linear-optical quantum computing, a formal bridge should be established between qubit-based and photonic QC. This would allow the construction of reliable software for compiling quantum computations into photonic circuits.

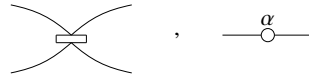
In this paper, we provide such a bridge by defining a functor from the ZX calculus to linear optics. In the process, we unify several results on the structure and combinatorics of quantum optical experiments. We start by studying the category of linear optical circuits, with their classical interpretation in terms of matrices or weighted paths (Sections 2). We then use the work of Vicary [30] to derive a functorial model for bosonic linear optics. Our first contribution is an explicit proof that this model is equivalent to the model based on matrix permanents of Aaronson and Arkhipov [6] (Section 3). Second, we introduce a graphical calculus QPath which allows to compute the amplitudes of linear optical events involving multiple photons, by rewriting diagrams to normal form (Section 4). Finally, we construct a functor from

the ZX calculus to QPath and use it to describe the basic protocols used in linear-optical and fusion-based quantum computing (Section 5).

Related work Graphical approaches of linear optics are widespread in the literature. Notable examples are the matchgates introduced by Valiant [32], corresponding to fermionic linear optics [33], whose amplitudes are computed by finding the perfect matchings of a graph. Graph-theoretic methods are also widely used in bosonic linear optics [34, 35]. There are strong links between linear optics and categorical logic. Blute et al. [36] studied Fock space as exponential modality for linear logic. The fermionic version of the Fock space has been studied in [37], it forms the W core of the ZW calculus introduced by Coecke, Kissinger and Hadzihasanovic [38, 39, 40]. More recently, there has been work on a diagrammatic calculus for reasoning about polarising beam splitters for quantum control [41], an informal essay describing bosonic linear optics with category theory [42], and a complete rewriting system for the single photon semantics of linear optical circuits [43]. The ZX calculus has also been used to describe the fault-tolerant aspects of fusion-based quantum computing [44].

2 Linear optical circuits

Linear optical circuits are generated by two basic physical gates. The *beam splitter* $\text{BS} : a \otimes a \rightarrow a \otimes a$ acts on a pair of optical modes, and may be implemented using prisms or half-silvered mirrors. The *phase shift* $\text{S}(\alpha) : a \rightarrow a$ acts on a single mode and has a single parameter $\alpha \in [0, 2\pi]$. We depict them:



Linear optical circuits are obtained from these gates by composing them vertically and horizontally. They form a set \mathbf{LO} , which has the structure of a free monoidal category, i.e. circuits can be composed in sequence or in parallel.

Definition 2.1. *The classical interpretation of \mathbf{LO} is given by a monoidal functor $\mathcal{U} : \mathbf{LO} \rightarrow \mathbf{Mat}_{\oplus}$ into the category of matrices over the complex numbers, where \oplus is the direct sum of vector spaces. On objects \mathcal{U} is defined by $\mathcal{U}(a) = \mathbb{C}$. On arrows we have:*

$$\mathcal{U}(\text{S}(\alpha)) = (e^{i\alpha})$$

$$\mathcal{U}(\text{BS}) = \frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix}$$

where we use one standard interpretation of the beam splitter [45].

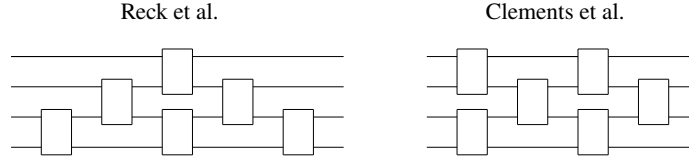
The Mach-Zehnder interferometer is obtained as the following composition:



The classical interpretation of this diagram is then given by:

$$\text{MZI}(\alpha, \beta) = ie^{i\alpha} \begin{pmatrix} -e^{i\beta} \sin(\alpha) & \cos(\alpha) \\ e^{i\beta} \cos(\alpha) & \sin(\alpha) \end{pmatrix}$$

MZIs may be used to parametrize any unitary map on m modes. They are the basic building blocks of integrated nanophotonic circuits currently being produced [7]. The first architecture for a universal multiport interferometer was proposed by Reck et al. [46] and consists of a mesh of MZIs. It was later simplified by Clements et al. into a grid-like architecture, reducing the depth from $d = 2m - 3$ to $d = m$ and thus the probability of photon loss [47].



Using one of these architectures, we have a parametrized circuit $c(\theta) : m \rightarrow m \in \mathbf{LO}$, where the parameters θ correspond to phases α, β of the Mach-Zehnder interferometers making up the chip. As shown in both [46] and [47], for any unitary $U : \mathbb{C}^m \rightarrow \mathbb{C}^m$, there is a configuration of parameters θ such that $\mathcal{U}(c(\theta)) = U$. We may restate their results in our notation.

Proposition 2.1 (Universality). [46, 47] *For any $m \times m$ unitary U , there is a circuit $c : m \rightarrow m$ in \mathbf{LO} such that $\mathcal{U}(c) = U$.*

In classical light experiments, we can measure the energy or *intensity* of an electromagnetic wave $E = E_0 e^{i(kx - \omega t)}$ where k is the wavenumber, ω is the angular frequency and E_0 is called the amplitude [48]. The intensity is then given by the quadratic quantity $I = \frac{c}{2} \epsilon_0 E_0^2$ where ϵ_0 is the permittivity of free space and c is the speed of light. The intensity is thus proportional to the Born rule $I \propto \|E\|^2$. Using the Born rule and the classical interpretation of \mathbf{LO} , we may compute the output distribution of a photonic chip $c \in \mathbf{LO}$ with m spatial modes, when the input is a classical or *incoherent* beam of light. Suppose the input intensities of light are $I \in \mathbb{R}^m$. One may assume $\sum_{i=1}^m I_i = 1$. Then the intensities J at the output of an interferometer $c \in \mathbf{LO}$ are given by:

$$J = \|\mathcal{U}(c)\|^2 I$$

where juxtaposition denotes matrix multiplication and the norm squared $\|\cdot\|^2$ is applied entry-wise. Note that $\|\mathcal{U}(c)\|^2$ is a doubly stochastic matrix since $\mathcal{U}(c)$ is unitary.

Example 2.1 (Classical light). *The intensities at the output of the beam splitter BS on any normalised input I are $J = (\frac{1}{2}, \frac{1}{2})$ since:*

$$\|\text{BS}\|^2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

The Mach-Zehnder interferometer yields the following stochastic matrix:

$$\|\text{MZI}(\alpha, \beta)\|^2 = \begin{pmatrix} \sin(\alpha)^2 & \cos(\alpha)^2 \\ \cos(\alpha)^2 & \sin(\alpha)^2 \end{pmatrix}$$

The reflection and transmission coefficients for light intensities are given by $R = \sin(\alpha)^2$ and $T = \cos(\alpha)^2$ with $R + T = 1$. Thus, if we input a beam of incoherent light on the left leg $I = (1, 0)$, we will observe the distribution $J = (R, T)$ in the output.

We have seen that linear optical circuits have a classical interpretation as complex-valued matrices. We now give a graph-theoretic interpretation of these circuits, using a syntactic category for counting paths. The classical **Path** calculus has the following generators:

$$\text{---} \begin{array}{c} \diagup \\ \diagdown \end{array} \text{---} , \text{---} \circ \text{---} , \text{---} \begin{array}{c} \diagdown \\ \diagup \end{array} \text{---} , \text{---} \circ \text{---} , \text{---} \times \text{---} , \text{---} \overset{r}{\circ} \text{---} \quad (1)$$

denoted respectively $\delta, \epsilon, \mu, \eta, \sigma$ and r . **Path** diagrams are obtained by composing these generators horizontally or vertically. Two **Path** diagrams are equal if we can rewrite from one to the other using the rules defined in Figure (1).

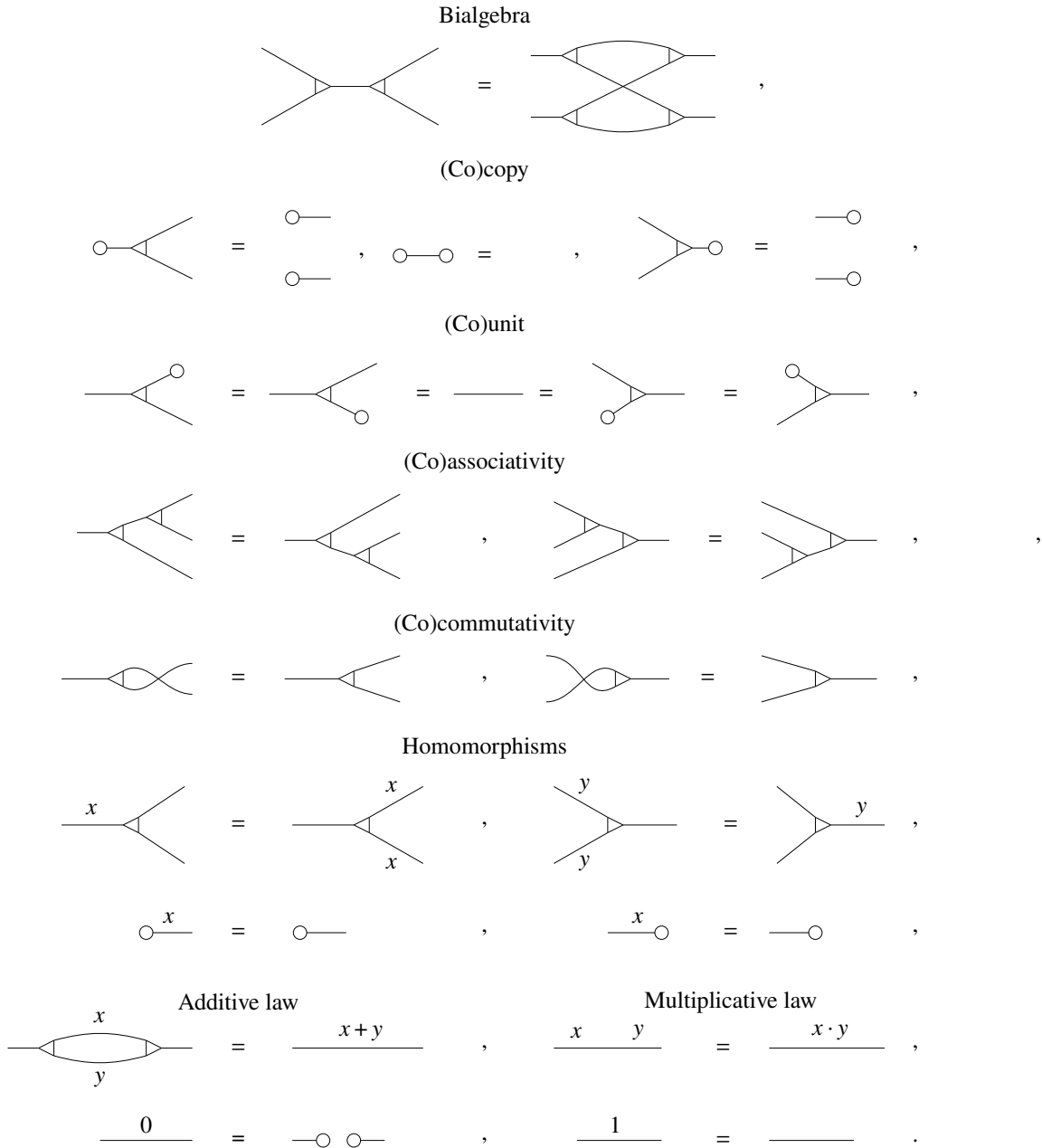


Figure 1: Axioms of the **Path** calculus

In categorical terms, we may define **Path** as the PROP generated by a bialgebra $(\delta, \epsilon, \mu, \eta)$ together with endomorphisms $r : 1 \rightarrow 1$ with a semiring structure $r \in \mathbb{S}$. Throughout this paper we fix $\mathbb{S} = \mathbb{C}$ although our main results can be generalised to any semiring. This calculus is folklore in category theory and was first studied by Pirashvili [49]. Bonchi, Sobocinski and Zanasi used it to model signal flow

graphs [50]. We can interpret **Path** in the monoidal category of matrices with direct sum.

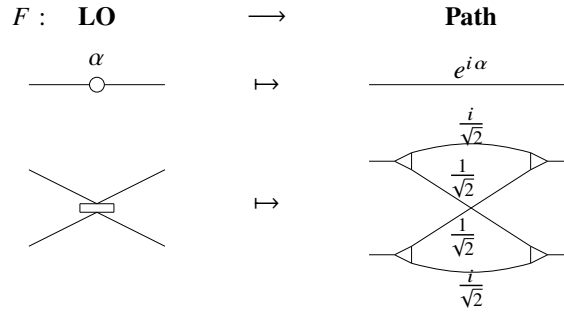
Proposition 2.2. *There is a monoidal functor $C : \mathbf{Path} \rightarrow \mathbf{Mat}_\oplus$.*

Proof. C is given on objects by $C(a) = 1$ and on the generators (1) by:

$$C(\delta) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad C(\epsilon) = () , \quad C(\mu) = (1 \ 1) , \quad C(\eta) = () , \quad C(r) = (r) , \quad C(\sigma) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

where $C(\epsilon) = () : 1 \rightarrow 0$ and $C(\eta) = () : 0 \rightarrow 1$ are the unique morphisms of that type in \mathbf{Mat}_\oplus . It is easy to check that all the relations in Figure 1 are satisfied by C . \square

Moreover, there is a functor turning linear optical gates into **Path** diagrams, representing their underlying matrix:

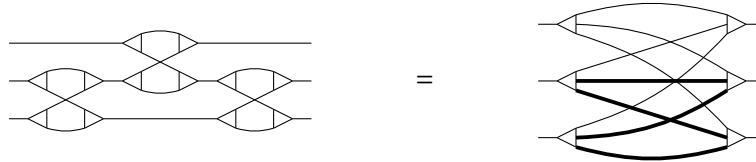


Proposition 2.3. *The classical interpretation of linear optics factors through the Path calculus, i.e. the functor $F : \mathbf{LO} \rightarrow \mathbf{Path}$ defined above satisfies $\mathcal{U} = C \circ F$.*

The rewrite rules of **Path** allow to reduce any diagram to a normal form, which carries the same data as a weighted bipartite graph. This normal form can be reached by the following (pseudo) algorithm:

1. remove all possible instances of $\eta : 0 \rightarrow 1$ and $\epsilon : 1 \rightarrow 0$ by using the (co)unit and (co)copy laws repeatedly.
2. apply the bialgebra law, together with homomorphism and multiplicative laws, until all instances of the comonoid δ precede all instances of the monoid μ ,
3. apply the additive rule to contract parallel edges.

As an example, the following equation holds in **Path**, the normal form procedure going from left to right.



where the thick wires carry the endomorphism $2 : 1 \rightarrow 1$. Computation of the weights on the resulting graph is equivalent to the block-diagonal matrix multiplication defined by C . This is stated formally as the following result.

Proposition 2.4 (Completeness). *The axioms of Path are complete for \mathbf{Mat}_\oplus , i.e. $C : \mathbf{Path} \rightarrow \mathbf{Mat}_\oplus$ is a monoidal equivalence.*

Proof. The normalisation procedure is described above, see also [50, Proposition 1]. \square

3 Fock space and permanents

Processing bosonic particles, such as photons, with linear optical devices gives rise to statistics that are hard to simulate classically [6]. In this section we give an interpretation of linear optical circuits, derived from [30], in terms of free and symmetric Fock space functors $\mathcal{F}, \mathcal{B} : \mathbf{Mat}_\oplus \rightarrow \mathbf{Vect}_\otimes$. We show that this characterisation is equivalent to the model introduced in [6].

Consider a box containing particles. Assume that the space of states of a single particle is given by a Hilbert space H . The *free* Fock space is defined as follows:

$$\mathcal{F}(H) = \bigoplus_{n=0}^{\infty} H^{\otimes n}$$

where \otimes is the usual tensor product and \oplus the direct sum. $\mathcal{F}(H)$ describes the state space of a given number of *distinguishable* particles indexed by n . Given a basis X of *modes* such that $H = \mathbb{C}X$, we have that:

$$\mathcal{F}(\mathbb{C}X) \simeq l^2(X^*)$$

where $\mathbb{C}X$ denotes the free vector space space with basis X , X^* is the set of lists over X and l^2 is the canonical Hilbert space construction defined in [51]. Thus for n particles in m modes we have $\mathcal{F}_n(\mathbb{C}^m) = (\mathbb{C}^m)^{\otimes n} \simeq \mathbb{C}([m]^n)$ the basis states $[m]^n$ are given by lists of length n using m distinct symbols.

Proposition 3.1. *The free Fock space can be extended to a functor $\mathcal{F} : \mathbf{Mat}_\oplus \rightarrow \mathbf{Vect}_\otimes$ defined on the n -particle sector by $\mathcal{F}_n(A) = A^{\otimes n}$ for matrices A .*

Proof. This follows by functoriality of tensor \otimes and biproduct \oplus . □

Now suppose that the particles in the box are *indistinguishable*. The state space of the system will then be described by the symmetric or *bosonic* Fock space, defined as follows:

$$\mathcal{B}(H) = \bigoplus_{n=0}^{\infty} H^{\hat{\otimes} n}$$

where $\hat{\otimes}$ is the quotient of the tensor product by the equivalence relation $x \hat{\otimes} y = y \hat{\otimes} x$, which ensures that the bosons are indistinguishable. One may show that $\mathcal{B}(\mathbb{C}X) \simeq l^2(\mathbb{N}^X)$, i.e. the bosonic Fock space over a set of modes X is spanned by the basis states of occupation numbers. The n -particle sector of the bosonic Fock space $\mathcal{B}_n(H)$ is the n -th component in the direct sum above. When $H = \mathbb{C}^m$ has dimension m , we have n indistinguishable particles in m possible modes. The basis states of $\mathcal{B}_n(H)$ are given by:

$$\Phi_{m,n} = \{ (s_1, \dots, s_m) \mid \sum_{i=1}^m s_i = n, s_i \in \mathbb{N} \} \subseteq \mathbb{N}^m$$

Note that $|\Phi_{m,n}| = \binom{m+n-1}{n}$ and $\mathcal{B}_n(H) = H^{\hat{\otimes} n} = \mathbb{C}(\Phi_{m,n})$. Let us compare the basis states for distinguishable and bosonic particles. There is a family of linear maps $\alpha_H : \mathcal{F}(H) \rightarrow \mathcal{B}(H)$ defined on the basis states of the n -particle sector $X \in [m]^n$ by:

$$\alpha(X) = \sqrt{\frac{n!}{\prod_{j=1}^m a(X)_j!}} |a(X)\rangle$$

where $a : [m]^n \rightarrow \Phi_{m,n}$ is defined by $a(X)_j = |\{i | X_i = j\}|$ for $j \in [m]$. Note that the normalisation factor is equal to the size of the pre-image $a^{-1}(a(X))$. Let us write the map α^\dagger explicitly:

$$\alpha^\dagger |I\rangle = \sqrt{\frac{N_I}{n!}} \sum_{X \in a^{-1}(I)} |X\rangle$$

where $N_I = \prod_{j=1}^m I_j!$. We can now use α to define the action of \mathcal{B} on arrows.

Proposition 3.2. [30] *The bosonic Fock space can be extended to a strong monoidal functor $\mathcal{B} : \mathbf{Mat}_\oplus \rightarrow \mathbf{Vect}_\otimes$ defined on arrows $A : m \rightarrow k$ by:*

$$\mathcal{B}_n(A) = A^{\hat{\otimes} n} = \alpha A^{\otimes n} \alpha^\dagger$$

and satisfying $\mathcal{B}(A \oplus B) = \mathcal{B}(A) \otimes \mathcal{B}(B)$.

Proof. Functoriality follows from naturality of $\alpha^\dagger \alpha$ [30, Lemma 6.6]. \mathcal{B} is moreover strong monoidal:

$$\mathcal{B}(\mathbb{C}X \oplus \mathbb{C}Y) \simeq \mathcal{B}(\mathbb{C}(X+Y)) \simeq l^2(\mathbb{N}^{X+Y}) \simeq l^2(\mathbb{N}^X \times \mathbb{N}^Y) \simeq \mathcal{B}(\mathbb{C}X) \otimes \mathcal{B}(\mathbb{C}Y)$$

□

We can use the bosonic Fock space to define a functorial model for linear optics.

Definition 3.1 (Functorial model). *The functorial interpretation of linear optics is given by the composition $\mathcal{B} : \mathbf{LO} \xrightarrow{\mathcal{U}} \mathbf{Mat}_\oplus \xrightarrow{\mathcal{B}} \mathbf{Vect}_\otimes$. Given a chip $c : m \rightarrow m \in \mathbf{LO}$, the probability of observing output state $J \in \Phi_{m,n}$ on input $I \in \Phi_{m,n}$ is given by:*

$$P_c^{\mathcal{B}}(J|I) = \|\langle J | \mathcal{B}(c) | I \rangle\|^2 = \|\langle J | \alpha \mathcal{U}(c)^{\otimes n} \alpha^\dagger | I \rangle\|^2$$

Aaronson and Arkhipov [6] introduced a formal model for linear optics based on matrix permanents.

Definition 3.2 (Permanent model [6]). *Given a chip $c : m \rightarrow m \in \mathbf{LO}$, the probability of observing output state $J \in \Phi_{m,n}$ on input $I \in \Phi_{m,n}$ is given by:*

$$P_c(J|I) = \frac{1}{N_I N_J} \|\text{Perm}(\mathcal{U}(c)_{I,J})\|^2$$

where $N_S = \prod_{j=1}^m S_j!$, Perm denotes the matrix permanent, and $U_{I,J}$ is the $n \times n$ matrix obtained from an $m \times m$ matrix U as follows. We first construct the $m \times n$ matrix U_J by taking J_j copies of the j th column of U for each $j \leq m$. Then we construct $U_{I,J}$ by taking I_i copies of the i th row of U_J .

We give an explicit proof that the models introduced above are equivalent, although the argument can be traced back to Fock [52].

Theorem 3.1. *The functorial model of linear optics is equivalent to the permanent model. Explicitly, for any $m \times m$ unitary U and basis states $I, J \in \Phi_{m,n}$*

$$\langle J | \mathcal{B}(U) | I \rangle = \frac{\text{Perm}(U_{I,J})}{\sqrt{N_I N_J}}$$

Proof. We start by expanding the left-hand side:

$$\begin{aligned}
\langle J | \mathcal{B}(U) | I \rangle &= \langle J | U^{\otimes n} | I \rangle = \langle J | \alpha U^{\otimes n} \alpha^\dagger | I \rangle = (\alpha^\dagger | J \rangle)^\dagger U^{\otimes n} (\alpha^\dagger | I \rangle) \\
&= \left(\sqrt{\frac{N_J}{n!}} \sum_{Y \in a^{-1}(J)} \langle Y | \right) U^{\otimes n} \left(\sqrt{\frac{N_I}{n!}} \sum_{X \in a^{-1}(I)} | X \rangle \right) \\
&= \frac{\sqrt{N_J N_I}}{n!} |a^{-1}(I)| \sum_{Y \in a^{-1}(J)} \langle Y | U^{\otimes n} | \hat{X} \rangle \\
&= \frac{\sqrt{N_J N_I}}{n!} \frac{n!}{N_I N_J} \sum_{\sigma \in S_n} \prod_{i=1}^n U_{\hat{X}_i, \hat{Y}_{\sigma(i)}} \\
&= \frac{1}{\sqrt{N_I N_J}} \text{Perm}(U_{I,J})
\end{aligned}$$

where $\hat{X} \in a^{-1}(I)$ and $\hat{Y} \in a^{-1}(J)$ are any chosen representatives. Note that this choice is irrelevant since we sum over all permutations, and so in particular we can set $(U_{I,J})_{ij} = U_{\hat{X}_i, \hat{Y}_j}$, yielding the last step. \square

Example 3.1 (Hong-Ou-Mandel). Consider the matrix of the beam splitter:

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix}$$

Suppose we input one boson in each port $I = (1, 1)$. There are three possible outcomes $J = (2, 0), (1, 1), (0, 2)$. We may determine the amplitudes of the different outcomes by computing permanents:

$$\text{Perm} \begin{pmatrix} i & i \\ 1 & 1 \end{pmatrix} = 2i \quad \text{Perm} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix} = 0 \quad \text{Perm} \begin{pmatrix} 1 & 1 \\ i & i \end{pmatrix} = 2i$$

The component for outcome $(1, 1)$ is 0. We deduce that the probability of observing one boson in each output port is 0. Thus interference ensures that the bosons bunch together at the output of the device, a phenomenon known as the Hong-Ou-Mandel effect.

4 Quantum paths and matchings

In the previous section we have shown that bosonic linear optics can be formulated equivalently in terms of Fock space and permanents. Aaronson and Arkhipov [6] used the second definition to show that sampling from a linear optical chip with bosonic particles is classically hard: if a classical computer can compute an additive approximation of matrix permanents then the polynomial hierarchy collapses. While this computational definition is useful for proving complexity results, we want to develop a diagrammatic syntax for programming linear optical circuits. We do this by developing a quantised calculus **QPath** which allows us to compute the amplitudes of linear optical events involving multiple photons, using simple rewrite rules. In order to quantise the **Path** calculus, we add creation and annihilation of particles as generators.

$$\mathbf{QPath} = \mathbf{Path} + \left\{ \begin{array}{c} |n\rangle \\ \bullet \text{---} \end{array} , \begin{array}{c} \text{---} \bullet \\ |n\rangle \end{array} \right\}_{n \in \mathbb{N}^+}$$

This yields a free monoidal category where we can represent linear optical processes with state preparations (creation) and post-selection (annihilation). Before developing a calculus around the **QPath**

generators, the first thing to note is that **QPath** is equivalent to **Path** if we interpret it classically, i.e. functors **QPath** \rightarrow **Mat**_⊕ are in bijective correspondence with functors **Path** \rightarrow **Mat**_⊕. In fact, black and white nodes are necessarily equal in **Mat**_⊕, since the unit 0 is both a terminal and an initial object. In order to interpret black nodes, representing modes occupied by photons, we need to use the bosonic Fock space functor.

The quantum interpretation $\mathcal{B} : \mathbf{QPath} \rightarrow \mathbf{Hilb}_\otimes$ is obtained on the Path generators (1) by composing $C : \mathbf{Path} \rightarrow \mathbf{Mat}_\oplus$ with the bosonic Fock space functor $\mathcal{B} : \mathbf{Mat}_\oplus \rightarrow \mathbf{Hilb}_\otimes$. The generating object a of **QPath** is mapped to the free Hilbert space $l^2(\mathbb{N})$. The comonoid $\delta : 1 \rightarrow 2$ is mapped as follows:

$$\text{---} \triangleleft \text{---} \quad \mapsto \quad \mathcal{B}(\delta) : |n\rangle \quad \mapsto \quad \sum_{k=0}^n \binom{n}{k}^{\frac{1}{2}} |k\rangle |n-k\rangle,$$

while the monoid μ is mapped to the dagger $\mathcal{B}(\mu) = \mathcal{B}(\delta)^\dagger$. White nodes are mapped to $|0\rangle, \langle 0|$, indicating that the mode is empty. Endomorphisms $r : 1 \rightarrow 1$ in **QPath** are interpreted as follows:

$$\text{---} \xrightarrow{r} \text{---} \quad \mapsto \quad \mathcal{B}(r) : |n\rangle \quad \mapsto \quad r^n |n\rangle$$

Finally, the black nodes in **QPath** are mapped respectively to $|n\rangle$ and $\langle n|$, indicating that the mode is occupied by n particles. Hadzihasanovic [40] showed directly that $(\mu, |0\rangle, \delta, \langle 0|)$ forms a bialgebra. In fact all the axioms that hold in the classical interpretation C also hold in the bosonic interpretation \mathcal{B} since it is defined by functor-composition. However, black nodes allow to express some processes which were not available in the classical semantics, as we will see below.

The axioms of **QPath** include all the axioms of **Path**, given in Figure 1. The only additional rules we will need to reason with black nodes are the following:

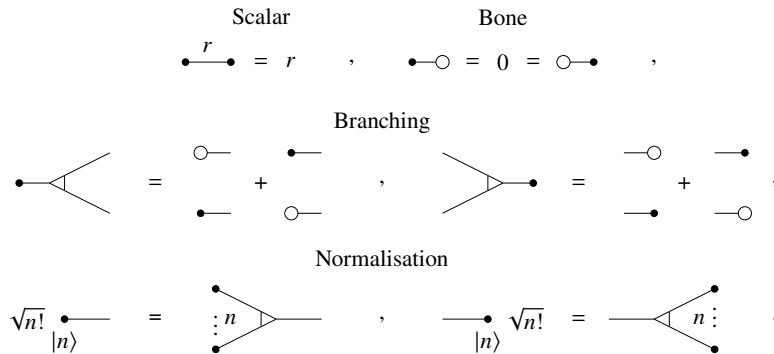


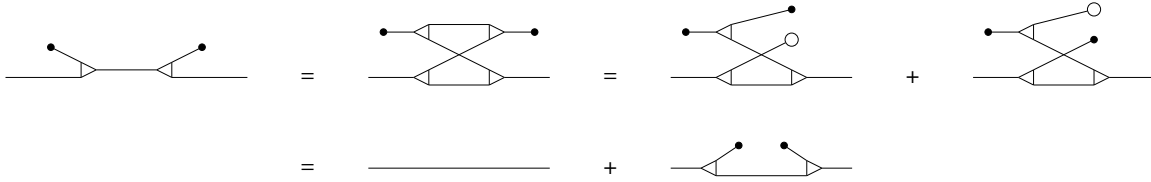
Figure 2: Additional axioms for the **QPath** calculus

It is easy to show that the axioms above are sound for the bosonic interpretation \mathcal{B} .

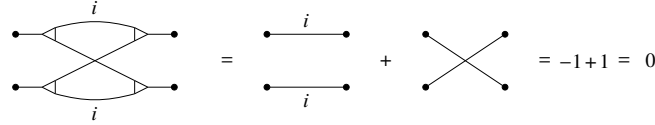
Example 4.1 (Creation/Annihilation). *The creation and annihilation operators on single modes have the following representation as **QPath** diagrams.*



We recover the commuting relations for these operators using the branching law:

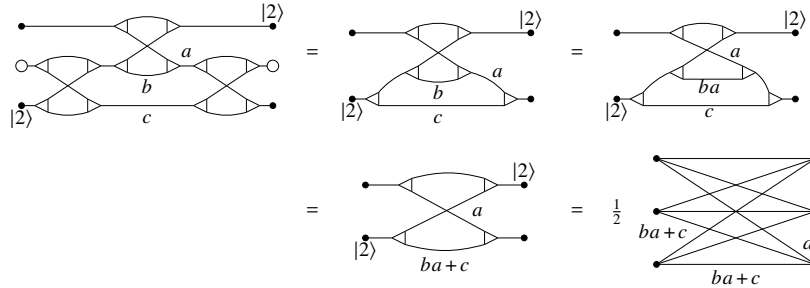


Example 4.2 (Hong-Ou-Mandel). We compute the amplitude of the beam splitter BS on input/output $I = (1, 1) = J$:



and we recover the zero amplitude for this event.

We interpret a *closed* diagram $d : 0 \rightarrow 0 \in \mathbf{QPath}$ as an event where particle creations are matched to particle annihilations. Given a linear optical circuit $c : m \rightarrow m \in \mathbf{LO}$ together with a pair of states $I, J \in \Phi_{m,n}$ of occupation numbers, we may construct a closed diagram $d = \langle J | F(c) | I \rangle \in \mathbf{QPath}$, corresponding to the event that we observe output J when we input I in a chip c . Using only the **Path** axioms together with the normalisation rule, we can rewrite d as in the following example.



where we use the following syntactic sugar: $\langle \cdot \rangle := \bullet \cdot \cdot$, $\cdot \rangle := \cdot \bullet$. At the end of the rewriting process, we obtain a weighted bipartite graph. Let us denote this graph by $G_d = (N, E)$ where N is the set of nodes and $E \subseteq N^2$ is the set of edges, together with $w : E \rightarrow \mathbb{C}$ an assignment of complex weights to every edge. Note that G is an undirected graph, i.e. $(i, j) \in E \implies (j, i) \in E$.

Proposition 4.1 (Normal form). Any closed diagram $d \in \mathbf{QPath}$ can be reduced to a pair (G_d, N_d) , where G_d is weighted bipartite graph and N_d is a normalisation factor, using the axioms in Figure 1 and the normalisation law.

Proof. The normal form procedure exemplified above is the same as for **Path**, with the addition of the use of the normalisation law which determines N_d . □

Once $d \in \mathbf{QPath}$ has been reduced to a weighted bipartite graph, we may further reduce it down to a scalar value by using the branching and scalar laws. Most terms obtained by branching will cancel out because of the first scalar law. The remaining terms are found to be in one-to-one correspondence with the *perfect matchings* of G_d . Recall that a matching for a graph G is a subset of the edges $M \subseteq E$ such that no node is contained in two edges of M . A perfect matching is a matching M such that every node is contained in an edge of M .

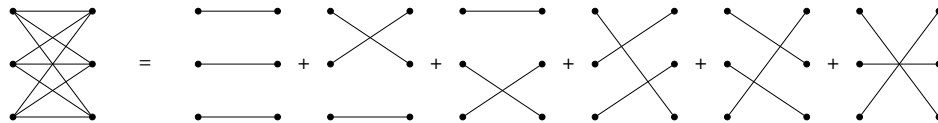
Theorem 4.1 (Matchings). *For closed diagrams $d : 0 \rightarrow 0 \in \mathbf{QPath}$, the rewrite rules of \mathbf{QPath} are complete for the bosonic interpretation $\mathcal{B} : \mathbf{QPath} \rightarrow \mathbf{Hilb}_{\otimes}$, which moreover satisfies:*

$$\mathcal{B}(d) = N_d \sum_M \prod_{e \in M} w_e \tag{2}$$

where M ranges over the perfect matchings of the graph G_d .

Proof. We need to show that if two closed diagrams d, d' have the same interpretation $\mathcal{B}(d) = \mathcal{B}(d') \in \mathbb{C}$, then we can rewrite from d to d' using the axioms of \mathbf{QPath} . To see this, note that for any closed diagram d , the branching law turns the graph G_d into a sum of n^n terms, where n is the number of photon preparations. We can cancel most of these terms using the “bone” law, which leaves us with $n!$ terms corresponding to the perfect matchings of G_d : each photon preparation is matched to a photon annihilation. Finally we reduce each of the terms to a complex value using the scalar law. It is a standard result in graph theory that the sum of weights of perfect matchings of a graph is equal to the permanent of its adjacency matrix, yielding (2). Therefore we can use the axioms of \mathbf{QPath} to reduce both d and d' to the same scalar value in \mathbb{C} . Since all the rules of \mathbf{QPath} are invertible we can rewrite from d to d' , yielding completeness. We do not currently know if the rules are complete also for “open” diagrams. \square

Example 4.3. *For a generic event d with three photons in \mathbf{QPath} , the normal form procedure gives us a weighted bipartite graph G_d with input and output of size 3, or equivalently we have a 3×3 adjacency matrix of weights. Using the branching law, we reduce the diagram to the following sum:*

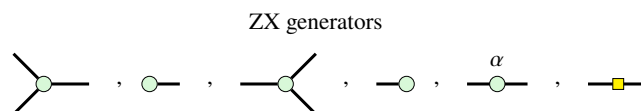


equivalently, we have just split the graph into its perfect matchings. Now we use the scalar laws to reduce each term to a complex number. Equivalently, we multiply the weights assigned to the edges on each matching. Finally we sum those terms to obtain the amplitude. Equivalently, we have computed the permanent of the adjacency matrix of G_d .

5 Linear-optical quantum computing

Our aim in this section is to describe how linear optics is used for qubit quantum computation. We will do this by giving a complete mapping from the \mathbf{ZX} calculus to \mathbf{QPath} . We start by introducing the \mathbf{ZX} calculus on qubits. The dual-rail encoding allows to encode a logical qubit as a photon in a pair of spatial modes. We show how all single qubit unitaries may be applied using simple linear optical devices. We describe fusion measurements as diagrams in \mathbf{QPath} and show how they can be used, along with polarising beam splitters, to construct Bell states and more general cluster states.

ZX calculus. The \mathbf{ZX} calculus is a graphical language for reasoning about qubit quantum computation. It has strong links with both circuit-based and measurement-based models of quantum computing [28]. The \mathbf{ZX} calculus is generated by the following basic operations:



We will also use the following syntactic sugar for X states and phases:

$$\bullet \text{---} := \circ \text{---} \square \text{---} \quad , \quad \overset{\alpha}{\bullet} \text{---} := \text{---} \square \circ \text{---} \overset{\alpha}{\bullet}$$

In this section we are not really interested in the rewrite rules for ZX diagrams, rather in their interpretation as linear maps between qubits. We will give this interpretation as we map each ZX generator to post-selected optical circuits in **QPath**, and refer to [53, 20] for more in-depth discussions.

Dual rail qubits. The dual-rail encoding can be thought of as a translation between polarized and spatial modes of photons. The polarization states of a single photon are spanned by the basis states $|H\rangle, |V\rangle$ for horizontal and vertical polarization, and thus naturally form a *qubit*. The dual-rail encoding consists in encoding a polarised mode of light as a pair of spatial modes in **LO** under the mapping $|H\rangle \mapsto |0, 1\rangle, |V\rangle \mapsto |1, 0\rangle$. The Z basis of a dual rail qubit may be expressed as a pair of **QPath** diagrams:

$$\bullet \text{---} \quad , \quad \overset{\pi}{\bullet} \text{---} \quad \begin{array}{c} \circ \text{---} \\ \bullet \text{---} \end{array} \quad , \quad \begin{array}{c} \bullet \text{---} \\ \circ \text{---} \end{array} \quad \{ |H\rangle, |V\rangle \}$$

The Z and X effects correspond to the following diagrams:

$$\text{---} \bullet \quad , \quad \text{---} \circ \quad \begin{array}{c} \circ \text{---} \\ \bullet \text{---} \end{array} \quad , \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} \text{---} \quad \{ \langle H|, \langle H| + \langle V| \}$$

The Z effect may be implemented by post-selecting a photon detector, the X effect by precomposition with a beam splitter. Z phases on dual-rail qubits are obtained as follows:

$$\overset{\alpha}{\circ} \text{---} \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad \begin{array}{l} |H\rangle \mapsto |H\rangle \\ |V\rangle \mapsto e^{i\alpha} |V\rangle \end{array}$$

The rotations from the Z basis to the X and Y bases are given by the beam splitters BS_H and BS respectively, defined as follows:

$$\begin{array}{c} \text{---} \square \text{---} \\ \overset{\pi/2}{\bullet} \text{---} \end{array} \quad \begin{array}{c} \diagup \\ \square \\ \diagdown \end{array} \quad H \quad = \quad \begin{array}{c} \diagup \\ \square \\ \diagdown \end{array} \quad \begin{array}{l} |H\rangle \mapsto |H\rangle + |V\rangle \\ |V\rangle \mapsto |H\rangle - |V\rangle \end{array}$$

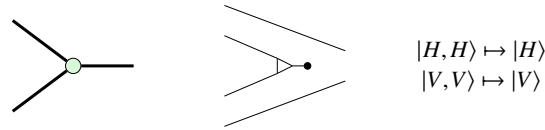
$$\begin{array}{c} \text{---} \square \text{---} \\ \overset{\pi/2}{\bullet} \text{---} \end{array} \quad \begin{array}{c} \diagup \\ \square \\ \diagdown \end{array} \quad = \quad \begin{array}{c} \diagup \\ \square \\ \diagdown \end{array} \quad \begin{array}{l} |H\rangle \mapsto i|H\rangle + |V\rangle \\ |V\rangle \mapsto |H\rangle + i|V\rangle \end{array}$$

where we use the following syntactic sugar: $\text{---} \text{---} := \text{---} \frac{-1}{\text{---}} \text{---}$, $\text{---} \text{---} := \text{---} \frac{i}{\text{---}} \text{---}$. We give the encoding up to scalar factor which does not affect the logic of the mapping. For example, the hadamard gate is technically $\frac{1}{\sqrt{2}}BS_H$. In conjunction with Z phases, we can use it to obtain all single qubit unitaries in dual rail encoding.

Example 5.1 (HOM). Using the beam splitters above, we obtain two versions of the Hong-Ou-Mandel effect which are depicted graphically as follows:

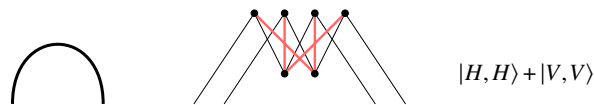
$$\begin{array}{c} \text{---} \square \text{---} \\ \text{---} \square \text{---} \end{array} \quad = \quad 0 \quad = \quad \begin{array}{c} \text{---} \square \text{---} \\ \text{---} \square \text{---} \end{array}$$

Fusion measurements. Fusion measurements are Bell measurements on dual-rail qubits. They correspond to the linear map: $|H, H\rangle \mapsto |H\rangle$, $|V, V\rangle \mapsto |V\rangle$, $|H, V\rangle, |V, H\rangle \mapsto 0$, which is denoted as a green spider with two inputs and one output in **ZX**, and is obtained on dual-rail qubits as the following diagram in **QPath**.

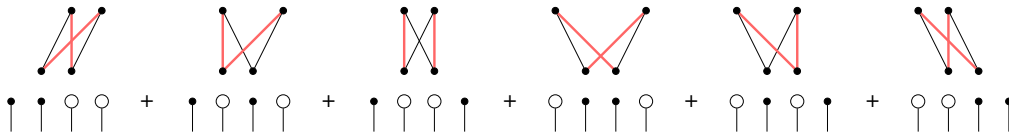


To see that this measures the Bell basis, note that there must be exactly one photon in the two middle modes. The input basis state in dual-rail encoding are $\{|0101\rangle, |0110\rangle, |1010\rangle, |1001\rangle\}$ and this condition is satisfied only by $|1010\rangle$ and $|0101\rangle$ which correspond respectively to $|H, H\rangle$ and $|V, V\rangle$.

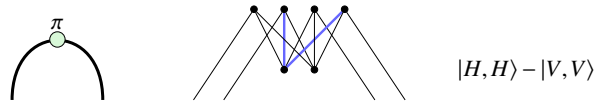
Bell states. We engineer a representation of dual-rail bell states as **QPath** diagrams.



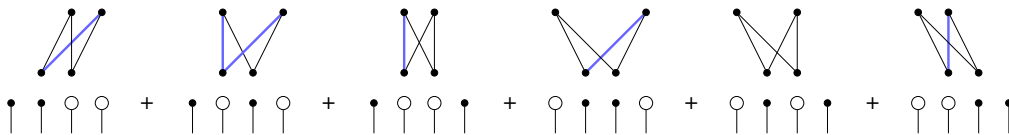
We can check that this diagram corresponds to the bell state $|H, H\rangle + |V, V\rangle$ by branching:



and using the Hong-Ou-Mandel effect (Example 5.1). Similarly, the Bell state $|HH\rangle - |VV\rangle$ may be represented using blue edes as follows:

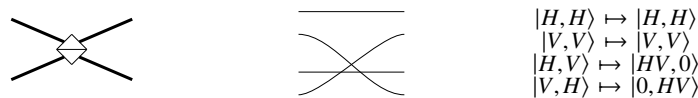


and we can check this using branching and HOM:

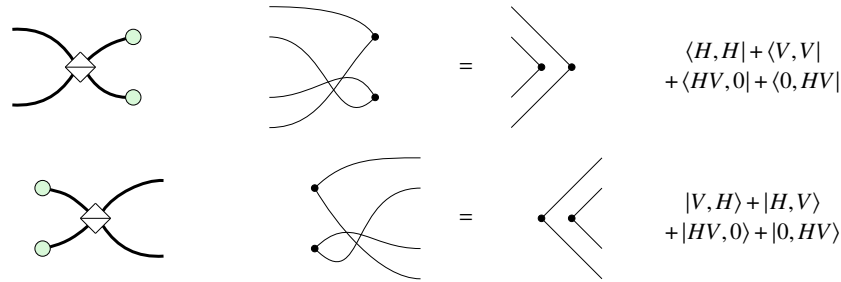


Note that there may be different equivalent representations of Bell states.

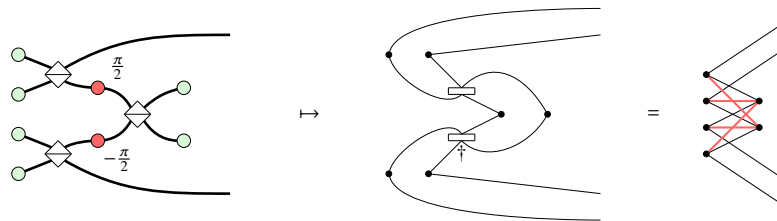
Polarising beam splitters. On bulk optics, the polarization states of photons can be acted upon using wave plates, polarizing beam splitters (PBSs) and photon counting measurements. Wave plates are simply X phase rotations, represented as red nodes in **ZX**. The PBS admits no description in **ZX**. It does however have a simple interpretation in **LO**:



In combination with X states and effects, polarising beam splitters can be used to perform post-selected fusion measurements and their transpose:

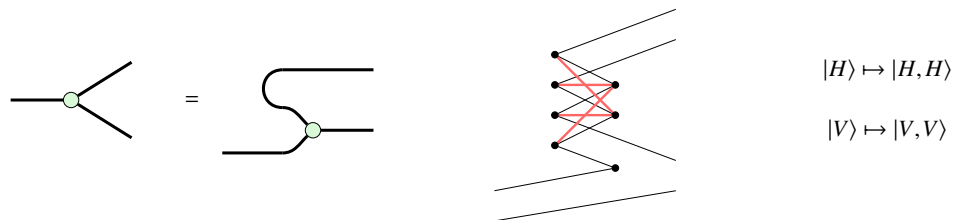


As an application, the linear-optical protocol for generating Bell states demonstrated in [13] may be described as a diagram using PBSs and ZX primitives:



We recover the diagram for the Bell state by reducing to normal form.

Spiders. The only missing ZX generator, which we need for a complete mapping $ZX \rightarrow QPath$, is the Z copy spider. We may readily deduce its representation using a known equality in ZX:



Similarly, we may turn the input leg into an output using a second Bell state. This yields a protocol for generating the dual-rail GHZ state using five ancillary photons. Note that the mapping is in no way unique, and we may obtain several equivalent protocols by further twisting the spider above. This however increases the number of ancillary photons needed. As first shown in [14], any cluster state can be obtained by performing additional fusion measurements.

Outlook

The theory in this paper is being implemented in DisCoPy [24], the Python library for monoidal categories. DisCoPy already has a number of tools for qubit quantum computing, including interfaces with tket [21], PyZX [22] and high-performance libraries for classical simulation. DisCoPy functors will allow to compile qubit circuits and cluster states into linear optical circuits for efficient simulation with Perceval [9] and future interfaces with photonic devices.

Acknowledgements

The authors would like to thank Richie Yeung, Harny Wang, Douglas Brown, Alex Cowtan and Anna Pearson also from Quantinuum in Oxford, Alexis Toumi in Paris, Amar Hazihasanovic in Estonia, Lee Rozema and Iris Agresti in Vienna and Terry Rudolph from PsiQuantum in California, for the many conversations on optics which led to this manuscript and the ones to come.

References

- [1] C. S. Wu and I. Shaknov. The Angular Correlation of Scattered Annihilation Radiation. *Physical Review*, 77(1):136–136, January 1950. doi:10.1103/PhysRev.77.136.
- [2] Alain Aspect, Jean Dalibard, and Gérard Roger. Experimental Test of Bell's Inequalities Using Time-Varying Analyzers. *Physical Review Letters*, 49(25):1804–1807, December 1982. doi:10.1103/PhysRevLett.49.1804.
- [3] D. Boschi, S. Branca, F. De Martini, L. Hardy, and S. Popescu. Experimental Realization of Teleporting an Unknown Pure Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels. *Physical Review Letters*, 80(6):1121–1125, February 1998. doi:10.1103/PhysRevLett.80.1121.
- [4] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe, and A. J. Shields. Gigahertz decoy quantum key distribution with 1 Mbit/s secure key rate. *Optics Express*, 16(23):18790–18797, November 2008. doi:10.1364/OE.16.018790.
- [5] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, December 2020. doi:10.1126/science.abe8770.
- [6] Scott Aaronson and Alex Arkhipov. The Computational Complexity of Linear Optics. *Proceedings of the forty-third annual ACM symposium on Theory of computing*, June 2011. doi:10.1145/1993636.1993682.
- [7] Giacomo Corrielli, Andrea Crespi, and Roberto Osellame. Femtosecond laser micromachining for integrated quantum photonics. *Nanophotonics*, 10(15):3789–3812, 2021. doi:10.1515/nanoph-2021-0419.
- [8] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry Fields: A Software Platform for Photonic Quantum Computing. *Quantum*, 3:129, March 2019. doi:10.22331/q-2019-03-11-129.
- [9] Nicolas Heurtel, Andreas Fyrrillas, Grégoire de Glinasty, Raphaël Le Bihan, Sébastien Malherbe, Marceau Pailhas, Boris Bourdoncle, Pierre-Emmanuel Emeriau, Rawad Mezher, Luka Music, Nadia Belabas, Benoît Valiron, Pascale Senellart, Shane Mansfield, and Jean Senellart. Perceval: A Software Platform for Discrete Variable Photonic Quantum Computing. *Quantum*, 7:931, February 2023. doi:10.22331/q-2023-02-21-931.
- [10] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, January 2001. doi:10.1038/35051009.
- [11] Michael A. Nielsen. Optical quantum computation using cluster states. *Physical Review Letters*, 93(4):040503, July 2004. doi:10.1103/PhysRevLett.93.040503.
- [12] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Review article: Linear optical quantum computing with photonic qubits. *Reviews of Modern Physics*, 79(1):135–174, January 2007. doi:10.1103/RevModPhys.79.135.
- [13] Qiang Zhang, Xiao-Hui Bao, Chao-Yang Lu, Xiao-Qi Zhou, Tao Yang, Terry Rudolph, and Jian-Wei Pan. Demonstration of efficient scheme for generation of "Event Ready" entangled photon pairs from single photon source. *Physical Review A*, 77(6):062316, June 2008. doi:10.1103/PhysRevA.77.062316.
- [14] Daniel E. Browne and Terry Rudolph. Resource-efficient linear optical quantum computation. *Physical Review Letters*, 95(1):010501, June 2005. doi:10.1103/PhysRevLett.95.010501.

- [15] Sara Bartolucci, Patrick Birchall, Hector Bombin, Hugo Cable, Chris Dawson, Mercedes Gimeno-Segovia, Eric Johnston, Konrad Kieling, Naomi Nickerson, Mihir Pant, Fernando Pastawski, Terry Rudolph, and Chris Sparrow. Fusion-based quantum computation. *arXiv:2101.09310 [quant-ph]*, January 2021. doi:10.1038/s41467-023-36493-1.
- [16] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 415–425, March 2004. doi:10.1109/LICS.2004.1319636.
- [17] P. Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, 2007. doi:10.1016/j.entcs.2006.12.018.
- [18] B. Coecke, É. O. Paquette, and D. Pavlović. Classical and quantum structuralism. In S. Gay and I. Mackie, editors, *Semantic Techniques in Quantum Computation*, pages 29–69. Cambridge University Press, 2010. doi:10.48550/arXiv.0904.1997.
- [19] B. Coecke, R. Duncan, A. Kissinger, and Q. Wang. Strong complementarity and non-locality in categorical quantum mechanics. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2012. arXiv:1203.4988. doi:10.1109/LICS.2012.35.
- [20] B. Coecke and A. Kissinger. *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. doi:10.1017/9781316219317.
- [21] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. $\text{TKet}\angle$: A Retargetable Compiler for NISQ Devices. *Quantum Science and Technology*, 6(1):014003, January 2021. doi:10.1088/2058-9565/ab8e92.
- [22] Aleks Kissinger and John van de Wetering. PyZX: Large Scale Automated Diagrammatic Reasoning. *Electronic Proceedings in Theoretical Computer Science*, 318:229–241, May 2020. doi:10.4204/EPTCS.318.14.
- [23] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. Lambeq: An Efficient High-Level Python Library for Quantum NLP. doi:10.48550/arXiv.2110.04236.
- [24] Giovanni de Felice, Alexis Toumi, and Bob Coecke. DisCoPy: Monoidal Categories in Python. *Electronic Proceedings in Theoretical Computer Science*, 333:183–197, Jan 2021. doi:10.4204/EPTCS.333.13.
- [25] E. R. Miranda, R. Yeung, A. Pearson, K. Meichanetzidis, and B. Coecke. A quantum natural language processing approach to musical intelligence. *Quantum Computer Music: Foundations, Methods and Advanced Concepts.*, 2021. doi:10.1007/978-3-031-13909-3_13.
- [26] Bob Coecke and Ross Duncan. Interacting Quantum Observables. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 298–310, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-70583-3_25.
- [27] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020. doi:10.22331/q-2020-06-04-279.
- [28] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, March 2021. doi:10.22331/q-2021-03-25-421.
- [29] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020. doi:10.22331/q-2020-01-09-218.
- [30] Jamie Vicary. A categorical framework for the quantum harmonic oscillator. *International Journal of Theoretical Physics*, 47(12):3408–3447, December 2008. doi:10.1007/s10773-008-9772-4.
- [31] M. Fiore. An axiomatics and a combinatorial model of creation/annihilation operators. *arXiv preprint arXiv:1506.06402*, 2015. doi:10.48550/arXiv.1506.06402.
- [32] Leslie G. Valiant. Quantum computers that can be simulated classically in polynomial time. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 114–123, New York, NY, USA, July 2001. Association for Computing Machinery. doi:10.1145/380752.380785.

- [33] Barbara M. Terhal and David P. DiVincenzo. Classical simulation of noninteracting-fermion quantum circuits. *Physical Review A*, 65(3):032325, March 2002. doi:10.1103/PhysRevA.65.032325.
- [34] Mario Krenn, Xuemei Gu, and Anton Zeilinger. Quantum Experiments and Graphs: Multipartite States as Coherent Superpositions of Perfect Matchings. *Physical Review Letters*, 119(24):240403, December 2017. doi:10.1103/PhysRevLett.119.240403.
- [35] Stefan Ataman. A graphical method in quantum optics. *Journal of Physics Communications*, 2(3):035032, 2018-03. Publisher: IOP Publishing. doi:10.1088/2399-6528/aab50f.
- [36] R. F. Blute, Prakash Panangaden, and R. A. G. Seely. Fock Space: A Model of Linear Exponential Types, 1994.
- [37] Giovanni de Felice, Amar Hadzihasanovic, and Kang Feng Ng. A diagrammatic calculus of fermionic quantum circuits. *Logical Methods in Computer Science ; Volume 15*, page Issue 3 ; 18605974, 2019. doi:10.23638/LMCS-15(3:26)2019.
- [38] B. Coecke and A. Kissinger. The compositional structure of multipartite quantum entanglement. In *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 297–308. Springer, 2010. doi:10.1007/978-3-642-14162-1_25.
- [39] A. Hadzihasanovic. A Diagrammatic Axiomatisation for Qubit Entanglement. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '15, pages 573–584. IEEE, 2015. doi:10.1109/LICS.2015.59.
- [40] A. Hadzihasanovic. *The Algebra of Entanglement and the Geometry of Composition*. PhD thesis, University of Oxford, 2017. doi:10.48550/arXiv.1709.08086.
- [41] Alexandre Clément and Simon Perdrix. PBS-Calculus: A Graphical Language for Coherent Control of Quantum Computations. In Javier Esparza and Daniel Král, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.24.
- [42] Paul McCloud. The Category of Linear Optical Quantum Computing. *arXiv:2203.05958 [quant-ph]*, March 2022. doi:10.48550/arXiv.2203.05958.
- [43] Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix, and Benoît Valiron. LOv-calculus: A graphical language for linear optical quantum circuits. In Stefan Szeider, Robert Ganian and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.35.
- [44] Hector Bombin, Chris Dawson, Ryan V. Mishmash, Naomi Nickerson, Fernando Pastawski, and Sam Roberts. Logical blocks for fault-tolerant topological quantum computation. *PRX Quantum* 4, 020303, April 2023. doi:10.1103/PRXQuantum.4.020303.
- [45] Francois Henault. Quantum physics and the beam splitter mystery. *The Nature of Light: What are Photons? VI*, 9570:199–213, 2015. doi:10.1117/12.2186291.
- [46] Michael Reck, Anton Zeilinger, Herbert J. Bernstein, and Philip Bertani. Experimental realization of any discrete unitary operator. *Physical Review Letters*, 73(1):58–61, July 1994. doi:10.1103/PhysRevLett.73.58.
- [47] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, W. Steven Kolthammer, and Ian A. Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, December 2016. doi:10.1364/OPTICA.3.001460.
- [48] David J Griffiths. *Introduction to electrodynamics*. Prentice Hall New Jersey, 1962. doi:10.1119/1.4766311 .
- [49] Teimuraz Pirashvili. On the PROP corresponding to bialgebras, October 2001. doi:10.48550/arXiv.math/0110014.

- [50] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A Categorical Semantics of Signal Flow Graphs. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014 – Concurrency Theory*, Lecture Notes in Computer Science, pages 435–450, Berlin, Heidelberg, 2014. Springer. doi:10.1007/978-3-662-44584-6_30.
- [51] Chris Heunen. On the Functor ℓ^2 . In Bob Coecke, Luke Ong, and Prakash Panangaden, editors, *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science, pages 107–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-38164-5_8.
- [52] V. Fock. Konfigurationsraum und zweite Quantelung. *Zeitschrift für Physik*, 75(9):622–647, September 1932. doi:10.1007/BF01344458.
- [53] John van de Wetering. ZX-calculus for the working quantum computer scientist. *arXiv:2012.13966 [quant-ph]*, December 2020. doi:10.48550/arXiv.2012.13966.

Identification of Causal Influences in Quantum Processes

Isaac Friend

Aleks Kissinger

Department of Computer Science
University of Oxford

{isaac.friend,aleks.kissinger}@cs.ox.ac.uk

Though the topic of causal inference is typically considered in the context of classical statistical models, recent years have seen great interest in extending causal inference techniques to quantum and generalized theories. Causal identification is a type of causal inference problem concerned with recovering from observational data and qualitative assumptions the causal mechanisms generating the data, and hence the effects of hypothetical interventions. A major obstacle to a theory of causal identification in the quantum setting is the question of what should play the role of “observational data,” as any means of extracting data at a locus will almost certainly disturb the system. Hence, one might think *a priori* that quantum measurements are already too much like interventions, so that the problem of causal identification trivializes. This is not the case. Fixing a limited class of quantum instruments (namely the class of all projective measurements) to play the role of “observations,” we note that as in the classical setting, there exist scenarios for which causal identification is impossible. We then present sufficient conditions for quantum causal identification, starting with a quantum analogue of the well-known “front-door criterion” and finishing with a broader class of scenarios for which the effect of a single intervention is identifiable. These results emerge from generalizing the process-theoretic account of classical causal inference due to Jacobs, Kissinger, and Zanasi beyond the setting of Markov categories, and thereby treating the classical and quantum problems uniformly.

1 Introduction

The problem of causal inference is to deduce from statistical correlations among variables something about the causal mechanisms responsible for those correlations, where a causal mechanism is a process that answers *interventional* queries. Although the majority of the work in the field of causal inference has focused on classical, statistical models, it is interesting to consider causal inference problems in the quantum setting as well, where quantum systems play the role of classical random variables. One can ask, for example, whether it is possible for agents confronted with a recurring scenario involving a pair of quantum systems to deduce using only certain limited operations whether the agents are in a common-cause-type situation (e.g., accessing two parts of a quantum entangled state) or a cause-effect-type situation (e.g., accessing the same system at two points in time). Ried et al. [16] presented a solution to such an inference problem for specific scenarios involving two quantum systems, and raised the problem of inference with larger collections of systems. Essentially what is sought is quantum generalization of some of the theory of statistical causal inference, which has systematized much of the business of combining qualitative knowledge of “causal structure” with quantitative data to characterize causal influences between variables. This article accomplishes such generalization, using the logical conception of causality presented in [12] to reveal the common process-theoretic underpinnings of causal inference in both ordinary stochastic and quantum settings.

A theory of quantum causal inference requires first a mathematical model of quantum causal scenarios. Here, we will take a minimal notion of a quantum causal model consisting of a “circuit with

holes,” i.e., a directed acyclic string diagram wherein some wires have gaps allowing agents to apply local processes. The circuit can be seen as a second-order process, or *comb* [3], which maps local non-deterministic processes to probabilities.

This notion of causal model is relatively weak in that unlike the one studied in [1] and [2], it doesn’t seem to admit a relation of “complete common cause” whereby a single intervenable quantum system can act as the sole source of correlations between two systems in its future. As a complete common cause can be pictured in the classical setting as “copying” a random variable and using it as input to two or more subsequent stochastic maps, it is difficult and somewhat subtle to make sense of a “complete quantum common cause” in the absence of a physically meaningful process of cloning or broadcasting quantum systems. Hence, it is interesting to see how much traction we can get on causal inference for a class of models that don’t admit the explicit general representation of complete common causes. We will show here that, in the case of the particular problem of *quantum causal identification*, we can get relatively far without such a representation. We also recover, from an abstract perspective, results in classical statistical causal inference.

Causal identification, in the classical case, refers to the problem of identifying the effects of (often hypothetical) interventions on the basis of purely observational data [15]. In contrast to related problems such as causal discovery, here the hypothesized causal structure of events—represented, e.g., by a directed acyclic graph depicting the possible directions of causal influence between random variables—is known in advance, but not the exact conditional probability distributions (or functional dependencies) governing the influence of individual variables on each other. Even with the causal structure given in advance, this problem can be highly non-trivial in the presence of confounding variables [15] or selection bias [8].

In generalizing to quantum causal identification, one needs to fix a notion that stands in the place of “observation,” as it is impossible to extract any data from a quantum system without causing a disturbance, which in some sense is already an active intervention. Here, we fix the class of processes playing the role of “observations” as local projective measurements, whereas “interventions” can be arbitrary quantum instruments. The latter includes, for example, the process of discarding the incoming state of a system and preparing a fixed new state, while the former does not.

While we do not intend to argue here that these notions of “observation” and “intervention” are fully conceptually justified, we will give strong evidence instead that this kind of quantum causal identification problem is interesting: we note that the problem can be impossible, then show that it becomes possible when a causal structure satisfies certain criteria.

By analogy to the classical case, causal identification is defined to be impossible when a causal structure admits a pair of models that behave identically with respect to projective measurements, but differently under arbitrary interventions. Simple such pairs of models were mentioned in [16], and we give an example. Our first new result is a quantum version of the front-door criterion for causal identifiability [14]. This result is then generalized to a sufficient condition for identifiability that implies the quantum analogues of multiple sufficient conditions in the statistical causal modeling literature, including some cases covered by Galles and Pearl in [11] and by Tian and Pearl in [18]. The statements and proofs here invoke diagrammatic technology presented in [5] and previously applied to causal inference by Jacobs, Kissinger, and Zanasi [12], who indicated the possibility, realized in the present article, of “import[ing] results from classical causal reasoning to the quantum case” by changing the concrete process theory in which abstract causal diagrams are modeled.

Potential consequences of the work lie in multiple areas. With respect to applicable quantum information science, the present work first of all describes how to identify certain interventional quantities in quantum networks of certain shapes, without full tomography. In fact, our results indicate that the abstract causal structure of a collection of quantum processes can sometimes be used to characterize those

processes completely, even with limited interactions. As Ried et al. explained, quantum causal inference schemes with limited operations “promise extensive applications in experiments exhibiting quantum effects” [16]. Our means of inference in scenarios involving unobserved common causes might apply specifically to the problem of detecting non-Markovianity in quantum information processing [17].

This work may also have consequences for the theory of non-quantum statistical causal inference. The process-theoretic presentation here, unifying classical and quantum causal identification, uncovers the basic structures and procedures—comb factorization, informationally complete sets of states and effects, and process tomography—that underpin causal inference but are often masked by the details of classical probability theory. The isolation of these rudiments should not only help guide the further development of theories of causal inference for quantum and other special kinds of processes, but also motivate continued research in ordinary statistical causal modeling using the logical and compositional techniques of theoretical computer science.

Finally, we hope this work will contribute to the program aimed at answering questions in the foundations of quantum physics by viewing them through the lens of causal modeling and inference [19]. In order to draw foundational conclusions from our results, those pursuing such ideas will have to assess the implications of the fact that projective measurement as “passive observation” defines a close quantum analogue of the classical problem of causal identification with latent variable models.

2 Preliminaries

To treat classical probability and quantum theory on the same footing, we will use the language of *process theories* [5] throughout. Process theories have been defined in slightly varying ways in the literature. Our definition follows.

Definition 1. A *process theory* is a symmetric monoidal category $(\mathcal{C}, \otimes, I)$.

The concrete classical and quantum process theories of causal models studied in this work are each equipped with a distinguished family of *discarding* morphisms $d_A : A \rightarrow I$ for each object A , satisfying $d_{A \otimes B} = d_A \otimes d_B$ and $d_I = 1_I$.

To give a physical or computational interpretation to process theories, it is typical to refer to generic morphisms $f : A \rightarrow B$ as *processes*, morphisms of the form $\rho : I \rightarrow A$ as *states*, and morphisms of the form $\pi : A \rightarrow I$ as *effects*. Morphisms of the form $\lambda : I \rightarrow I$ are called *numbers* or *scalars*. Objects are also called *system-types*.

Throughout the paper, we will adopt *string diagram* notation, where processes are depicted as boxes and objects as wires. We depict discarding using a black dot.

$$\begin{array}{ccc}
 f : A \rightarrow B & \rightsquigarrow & \begin{array}{c} |B \\ \boxed{f} \\ |A \end{array} & \quad & \rho : I \rightarrow A & \rightsquigarrow & \begin{array}{c} |A \\ \nabla \\ \rho \end{array} & \quad & d_A : A \rightarrow I & \rightsquigarrow & \begin{array}{c} \bullet \\ |A \end{array} \\
 & & & & \pi : A \rightarrow I & \rightsquigarrow & \begin{array}{c} \triangle \\ \pi \\ |A \end{array} & & & &
 \end{array}$$

Note that the discarding maps in a process theory are not required *a priori* to satisfy any equations aside from the basic compatibility with \otimes . They play an important role, however, in identifying certain families of well-behaved maps within a process theory. The most important such condition is the following.

Definition 2. A map $f : A \rightarrow B$ is called *causal* if $d_B \circ f = d_A$, or diagrammatically:

$$\begin{array}{c} \bullet \\ | \\ B \\ \square \\ f \\ | \\ A \end{array} = \begin{array}{c} \bullet \\ | \\ A \end{array} \tag{1}$$

Intuitively, causality captures the fact that the only influence a map can have is on its “future,” i.e., its output. If the output is discarded, then the actual causal process that took place is irrelevant.

Our main examples of process theories are $\mathbf{Mat}[\mathbb{R}_+]$ and \mathbf{CPM} , which contain (finite-dimensional) classical probability theory and quantum theory, respectively.

Example 1. The process theory $\mathbf{Mat}[\mathbb{R}_+]$ has as objects natural numbers and as morphisms $M : m \rightarrow n$ the $n \times m$ matrices whose entries are non-negative real numbers. The monoidal product is given by tensor product of matrices (a.k.a. Kronecker product), whose unit is the 1×1 matrix $(1) : 1 \rightarrow 1$. Discarding maps $d_n : n \rightarrow 1$ are the $1 \times n$ matrices (i.e. row vectors) consisting of all 1’s. Consequently, causal states are column vectors of positive numbers whose entries sum to 1 (i.e., probability distributions), and causal processes are matrices whose columns each sum to 1 (i.e., stochastic maps, equivalent to conditional probability distributions with $P(i|j) := M_{ij}$).

Example 2. The process theory \mathbf{CPM} has as objects finite-dimensional Hilbert spaces $\mathcal{H}, \mathcal{K}, \dots$ and as morphisms completely positive maps $\Phi : L(\mathcal{H}) \rightarrow L(\mathcal{K})$, where $L(\mathcal{H})$ is the algebra of operators $\mathcal{H} \rightarrow \mathcal{H}$. The monoidal product is again given by tensor product, whose unit is the identity map on $L(\mathbb{C}) \cong \mathbb{C}$. States $\rho : \mathbb{C} \rightarrow L(\mathcal{H})$ are fixed by a single positive operator $\rho(1) \in L(\mathcal{H})$ and causal states correspond to trace-1 positive operators. More generally, causal processes are the trace-preserving completely positive maps.

We will furthermore find it convenient to assume that each process theory has a (self-dual) compact structure, meaning that every object A is equipped with a pair of maps $\cup_A : I \rightarrow A \otimes A$ and $\cap_A : A \otimes A \rightarrow I$, called “cups” and “caps” respectively, satisfying the so-called *yanking equations*, which are depicted in string diagram notation as follows:

$$\begin{array}{c} \cup \\ \cup \\ \cup \end{array} = \begin{array}{c} | \\ | \\ | \end{array} = \begin{array}{c} \cap \\ \cap \\ \cap \end{array}$$

This structure enables us easily to represent higher-order maps as first order ones. For example, we can represent a process that takes processes of type $A \rightarrow A'$ and produces processes of type $B \rightarrow B'$ as a normal, first-order process $f : B \otimes A' \rightarrow A \otimes B'$. We then indicate its higher-order interpretation by drawing f as a box with a “hole” in it, and use cups and caps to define “plugging” another box into that hole:

$$\begin{array}{c} |A| \\ \square \\ f \\ |B| \\ |A' \end{array} \rightsquigarrow \begin{array}{c} B' \\ | \\ A' \\ \square \\ f \\ | \\ A \\ | \\ B' \end{array} \tag{2}$$

$$\begin{array}{c} B' \\ | \\ A' \\ \square \\ g \\ | \\ A \\ \square \\ f \\ | \\ B' \end{array} := \begin{array}{c} B' \\ | \\ A' \\ \square \\ g \\ | \\ A \\ \square \\ f \\ | \\ B' \\ | \\ A' \end{array} \tag{3}$$

In [12], the authors furthermore assumed the structure of a CDU category—a minor variation on the notion of a Markov category [10]—which captures an abstract notion of probabilistic maps by assuming every object carries a “copying” (a.k.a. “broadcasting”) map [6]. In particular, this structure allows one to capture causal models based on Bayesian networks as certain functors between CDU categories.

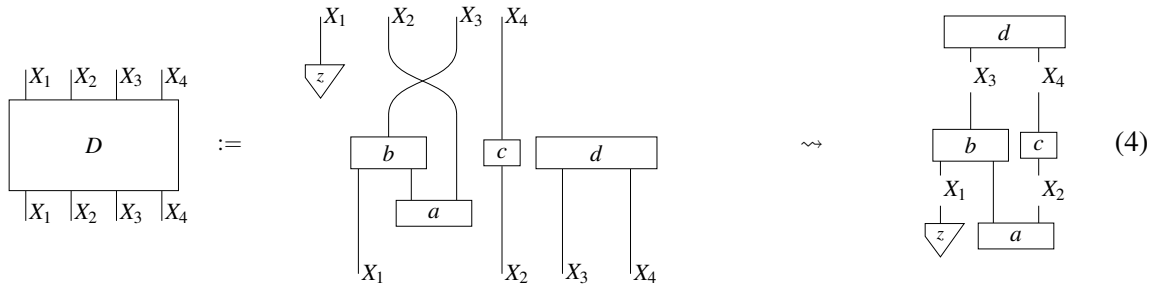
The famous no-cloning/no-broadcasting theorems of quantum theory, however, rule out a Markov-like structure in the category **CPM** of quantum maps. Hence, we adopt a weaker notion of causal model, consisting of a formal string diagram (i.e., a morphism in the free category over a signature) and an interpretation of that diagram into a concrete process theory (of, e.g., probabilistic or quantum maps).

3 Interventional causal models

A causal model consists of two parts: (i) a formal string diagram capturing our causal hypotheses, and (ii) an associated interpretation in a concrete process theory (i.e., $\mathbf{Mat}[\mathbb{R}_+]$ or **CPM**). We will also use the word “model” to refer just to (ii): the interpretation in the concrete process theory gives a model of (i) in a logical sense.

We define a formal string diagram as a morphism of a particular form in the free symmetric monoidal category $\mathbf{Free}(\Sigma)$ over some signature Σ . For a fixed set of objects $\{X_1, \dots, X_n\}$ in Σ , we call a diagram $D : X_1 \otimes \dots \otimes X_n \rightarrow X_1 \otimes \dots \otimes X_n$ a *circuit with holes* if it is a morphism in the free symmetric monoidal category and furthermore has the property that joining each input X_i to its corresponding output X_i yields another morphism in the free SMC (i.e., it doesn’t introduce a directed cycle).

The intuition is that each of the input/output pairs is a “hole” in the diagram, which we call an *intervention locus*, or simply *locus* (plural loci), where a local process can be plugged in. For example:



We require a locus’s input and output system-types to be identical, partly in order to accommodate the special “trivial intervention,” which joins a locus’s input and output with an identity wire. More broadly, statistical causal inference often involves considering a pair of instances of a single variable, with, e.g., one being observed and the other set by intervention. What is being represented is the same causal relation at two different “times,” before and after intervention.

We can now introduce a notion of causal model that is similar in spirit to that of [12], but no longer relies on the CDU structure needed to capture Bayesian networks.

Definition 3. For any process theory \mathcal{C} , an interventional causal model consists of a pair (D, Φ) where D is a circuit-with-holes in $\mathbf{Free}(\Sigma)$, Φ is a causal process in \mathcal{C} , and there exists a symmetric monoidal functor $F : \mathbf{Free}(\Sigma) \rightarrow \mathcal{C}$ such that $F(D) = \Phi$.

When $\mathcal{C} = \mathbf{Mat}[\mathbb{R}^+]$ we call (D, Φ) a *classical interventional causal model*, whereas when $\mathcal{C} = \mathbf{CPM}$, we call it a *quantum interventional causal model*.

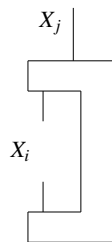
The shape of the abstract diagram D containing loci X_i and X_j may prohibit models in which interventions at X_i can causally affect events at X_j . Call locus X_j a *descendant* of locus X_i in abstract diagram D if and only if, when every input other than X_i and X_j is joined to its corresponding output, the resulting circuit has a path from input X_i (the wire leaving locus X_i) to output X_j (the wire arriving at locus X_j) along which every traversal of a box is in the upward direction. Thus in the example depicted in 4, X_4

is a descendant of X_2 , but not of X_3 . If locus X_j is not a descendant of locus X_i in abstract diagram D , then for any model of D , for every fixed set of causal processes plugged into the loci other than X_i and X_j , plugging a causal process into locus X_i yields a process, with only input/output pair X_j , that does not depend on the choice of causal process at X_i . In other words, X_j being a non-descendant of X_i in D captures the hypothesis that interventions at locus X_i cannot possibly affect events at X_j , and the problem of identifying the causal influence of X_i on X_j is uninteresting. Note finally that if X_i is a descendant of X_j , then X_j is not a descendant of X_i , and the uninteresting identification scenario obtains. We therefore lose nothing by focusing henceforth on the case wherein X_i is not a descendant of X_j .

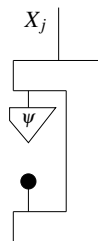
Inferring the ‘‘causal influence’’ of one locus on another will mean inferring, from whatever data are available, the value of a certain process determined by the causal model. That process, called an *interventional channel* between the two loci, encodes the quantitative causal relation between the loci according to the model at hand.

Definition 4. In a classical or quantum interventional causal model with loci X_1, \dots, X_n , the *interventional channel from X_i to X_j* , where X_i is a non-descendant of X_j , is the process obtained by filling in all loci other than X_i and X_j with identity interventions, and inputting a normalized, i.e., causal, state to the wire leaving locus X_j .

The interventional channel—whose definition as above using an unspecified causal state is made possible by the assumption that X_i is a non-descendant of X_j —is a process of the form



which maps (possibly non-deterministic) intervention outcome— $f : X_i \rightarrow X_i$ at locus X_i to the state on system X_j resulting from the combination of intervention f at X_i and trivial (identity) interventions at all loci other than X_i and X_j . In particular, the interventional channel gives the consequence for X_j of forcibly setting the state leaving X_i to ψ :



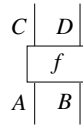
Thus the interventional channel yields what in ordinary statistical causal inference is called an ‘‘interventional distribution’’ of X_j due to ‘‘surgical intervention’’ at X_i . Moreover, one can compose the interventional channel with arbitrary causal processes $X_i \rightarrow X_i$ to evaluate the influences of so-called soft interventions [7], for which the state leaving a locus depends on the incoming state.

Thus the shift in focus from distributions to channels, in line with a general trend toward channel-based accounts of probabilistic reasoning [13, 4] and suggested for the present work by the difficulties of defining and reasoning with quantum analogues of conditional probability distributions, has definite advantages for a theory of causal inference. The single process called the interventional channel supports uniform reasoning about the consequences of all kinds of interventions, including soft interventions,

which are likely to be the norm in applications to quantum information processing, where the interventions under consideration may be, e.g., coherent quantum processes. The channel-based approach is both conceptually clarifying in its application to classical causal models, and especially suited for the most pertinent problems of quantum causal inference.

One expects that an interventional causal model’s data-generating process should be completely discoverable from the results of various interventions [9]. That is, there should exist a set of instruments for which the outcome statistics suffice to determine the data-generating process. This property is guaranteed for the interventional causal models in the present article by a key commonality between the classical and quantum processes studied in this work: they can be completely specified by the numbers that result when they are locally composed with states and effects.

Proposition 1. *The theories $\mathbf{Mat}[\mathbb{R}_+]$ and \mathbf{CPM} have local process tomography: any process*



is determined by numbers



where $i, j, k,$ and l index certain finite sets of states or effects on the appropriate system-types. In quantum tomography literature, the appropriate sets are called “informationally complete.”

We call the set of numbers in equation (5) the *generalized matrix elements* associated with a process f . A local process tomography protocol for causal—i.e., probability-preserving—maps in $\mathbf{Mat}[\mathbb{R}_+]$ and \mathbf{CPM} uses observed probabilities of combinations of measurement outcomes conditioned on combinations of causal state preparations. In the quantum case, though one cannot obtain all of the generalized matrix elements using a single choice of measurement basis, it is always possible to obtain them from the measurement statistics of multiple projective measurements at the outputs (along with independent state preparations at the inputs).

Local process tomography for comb-shaped quantum processes, which corresponds to Ried et al.’s [16] “causal tomography,” is mathematically just the same as local process tomography for ordinary first-order processes, but in the physical implementation, the measurement realizing an effect at a locus precedes temporally the preparation of the state leaving that locus. Thus local process tomography for a classical or quantum interventional model typically relies on probabilities that result from filling intervention loci with maps of the form



where i and j index informationally complete sets of effects and causal states. To implement all these maps in an experiment and thereby learn corresponding probabilities requires the ability to record an observation labeled i and then prepare the system in a new state labeled j , where j does not depend on i .

In contrast, what we will call observational data arise, for instance, when the only outcomes of this form that can be implemented are those satisfying $i = j$: the state to be fed forward from a locus is determined by the observation outcome, and so, for simplicity, we label them with the same value of a single index. This article is concerned with what can be inferred circumstances like these. The general problem of causal identification is to use qualitative assumptions about the causal scenario to compute quantitative causal influences given statistics from only a highly restricted set of interventions. Usually the allowed interventions are “passive observations,” which non-deterministically implement certain maps of the aforementioned “observational” form, and thereby teach the observer certain limited sets of probabilities. There is no quantum instrument representing a procedure appropriately called passive observation. For the purposes of this paper, the quantum interventions allowed as “observations” are exactly the projective measurements, which include identity processes (totally uninformative measurements) as well as instruments whose outcomes take the form of (6) for $i = j$.

The class of projective measurement instruments is closely related to a criterion, called *informational symmetry*, whereby Ried et al. characterized certain interventions in both classical and quantum causal scenarios as mere observations [16]. Informational symmetry, however, depends on both the intervening process and the prior state, whereas we desire a criterion applying only to the intervening process itself.

To apply our proofs of sufficient conditions for identifiability to the classical stochastic setting, we need not characterize completely a classical stochastic analogue of the quantum class of observation outcomes, but only posit that classical observation outcomes include identity matrices and matrices that have all zero entries except 1 in a single position on the main diagonal. (When classical probability theory is viewed as a sub-theory of quantum theory, the non-identity classical observation outcomes just described are in fact identified with outcomes of maximally informative projective measurement in a fixed basis.) The latter kind of matrix represents an outcome of what is normally called “observing a random variable.” By marginalization, identity interventions in the classical setting can be simulated from the probabilities of such projections onto pure causal states (point distributions). Thus our proofs of classical identifiability really appeal to no intervention procedures other than ordinary maximally informative classical observation. When we say an inference in the classical setting is impossible with only observational data accessible, “observational data” means probabilities of the classical outcomes just described.

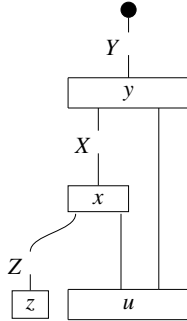
The question of identifiability of the causal influence of one locus on another is whether the qualitative causal assumptions encoded in the abstract string diagram D are strong enough that together with observational data for a causal scenario represented by an unknown model of D , they determine the value of the interventional channel derived from the unknown model. A classical or quantum interventional channel, respectively, from one locus to another will be called identifiable from an abstract string diagram if for any *positive* stochastic or quantum model of the string diagram, the interventional channel can be computed from the probabilities of arbitrary combinations of observation outcomes at all intervention loci of the model. Positivity is defined as follows:

Definition 5. A positive stochastic or quantum interventional model is a model whose composition with any non-zero state and any non-zero effect gives a strictly positive number.

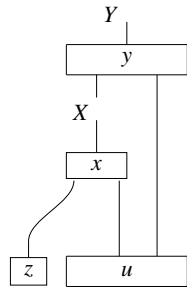
The states and effects composed with a model may in particular be products of those implemented at individual intervention loci. For a positive model, therefore, any combination of observational outcomes occurs with non-zero probability. The positivity condition in our process-theoretic account serves the same purpose as the common requirement in ordinary causal modeling that a probabilistic causal model induce a strictly positive joint distribution on all variables. Positivity ensures that all relevant conditional probabilities are defined, and that detecting an arbitrary state at a locus after intervening at another

locus is at least possible—if it were not, asking for the corresponding interventional probability would make no sense. The definition of identifiability from an abstract string diagram captures the notion that the assumptions of no direct influence between loci disconnected in the abstract diagram—equivalent to assumptions of absence of certain arrows in a directed acyclic graph representing a classical [15] or quantum [9] causal structure—suffice for inference: in any model satisfying at least the constraints implied by the string diagram, the quantity in question can be deduced from observational outcome statistics.

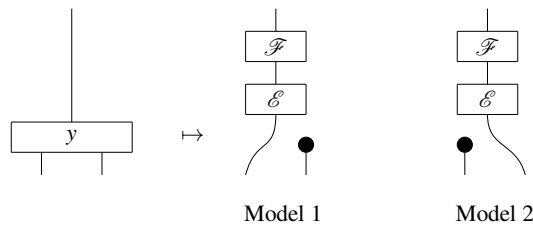
Circumscribing the class of allowed interventions raises the question of whether the restrictions are strong enough to rule out schemes like causal tomography that would always allow causal identification. The answer is affirmative, as Ried et al. [16] noted, and is evident from string diagrams like



for which the interventional channel



from X to Y is not identifiable. Two models yielding different interventional channels but identical observational outcome statistics are constructed via functorial interpretation according to Definition 3: in both models, u is interpreted as the Bell state $|\Psi^+\rangle\langle\Psi^+| = \frac{1}{2}(|0\rangle + |1\rangle)(\langle 0| + \langle 1|)$ on two qubits, z as a fixed quantum state with full support (i.e., a state whose composition with any non-zero effect is non-zero, corresponding to an operator of full rank), and x as the quantum map that discards its left-hand input and outputs its right-hand input unchanged. In the first model, y is interpreted as the map that discards its right-hand input and applies to its left-hand input a projective measurement followed by a depolarizing channel with parameter λ . In the second model, y is interpreted as the map that discards its left-hand input and applies to its right-hand input the same projective measurement followed by the same depolarizing channel as in the first model. Thus the interpretations of y in the two models are



where

$$\begin{aligned}\mathcal{E}(\rho) &= |0\rangle\langle 0|\rho|0\rangle\langle 0| + |1\rangle\langle 1|\rho|1\rangle\langle 1| \\ \mathcal{F}(\sigma) &= (1 - \lambda)\sigma + \lambda\left(\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|\right).\end{aligned}$$

These models are positive because z has full support, the reduced state arriving at X is maximally mixed, and the state arriving at Y includes a maximally mixed state with weight λ no matter what outcome has occurred at X . They produce identical outcome statistics for projective measurements at the loci. The interventional channel is therefore not identifiable from the abstract string diagram, because it cannot be computed for *every* positive model. The theory behind the two-locus quantum identification schemes of [16], however, might sometimes help in three-locus situations—perhaps some scenarios that involve coherence or entanglement and also include an “instrumental variable,” which would correspond here to the locus Z .

Because Z does not influence X or Y in these models, this example is essentially equivalent to those two-variable scenarios in [16] for which the desired interventional channel was noted to be unidentifiable. We show three variables to detach our example from the two-variable case of the “quantum advantage.” Moreover, we explicitly note that identification is impossible even for positive models of the string diagram.

4 Front-door scenarios

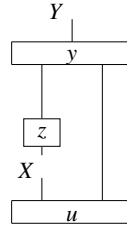
It is generally impossible to tell from observational data whether two correlated random variables, one of which is known *not* to be a descendant of the other—i.e., one of which comes “before” the other—stand in a cause-effect relation or are instead descendants of an unobserved common cause. If, however, there is a third observed variable or set of variables along the possible path of causal influence between the first two, the “front-door criterion” for causal identifiability implies that such inference may be possible. The operative sufficient condition has a quantum analogue, captured along with the classical version by the following result, which is derived for both process theories simultaneously, using the fact that the theories’ scalars are real numbers.

In this and the following section, each system represented by an uppercase letter may be a composite of multiple smaller systems, and similarly each box may be a composite of smaller boxes. Thus, a single locus in one of our diagrams might correspond to a list of several classical variables or quantum laboratories [9] occupying several nodes of a more traditional causal diagram. What we call an intervention at a locus would then correspond to (possibly choreographed/non-local) intervention at all those nodes.

Proposition 2. *For quantum or stochastic models of a string diagram*



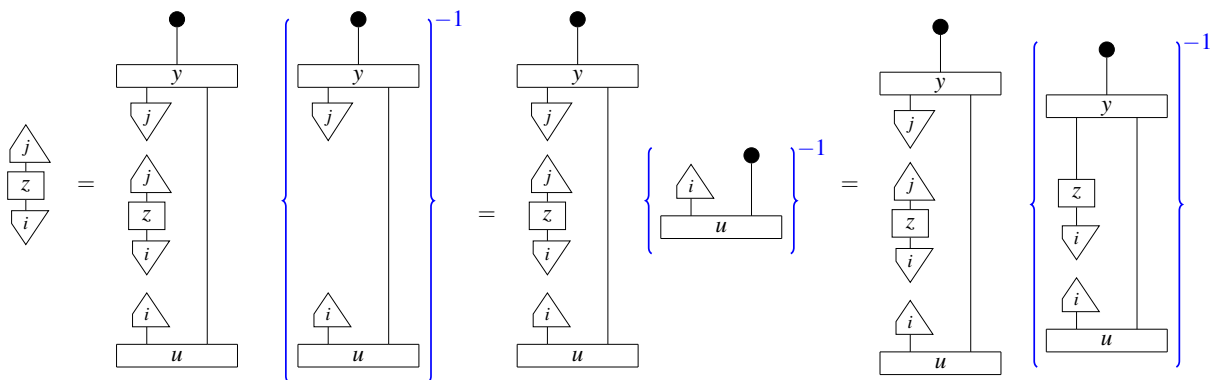
the interventional channel



from X to Y is identifiable.

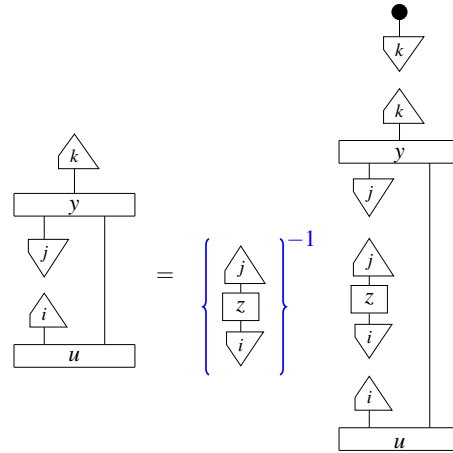
Each proof of identifiability from here on consists in demonstrating how to compute the causal quantity of interest—usually an interventional channel—from certain component processes of the model, specified by their generalized matrix elements, which are ultimately computed from probabilities of local observation outcomes. Each proof can be read in either the classical or the quantum process theory. An observation outcome consisting of an effect labeled, say, i , followed by a state with the same label, is to be understood in the classical case as a row vector with 1 in the i th position followed by the transpose of that vector. The composite map is the matrix product of the two. In the quantum case, a state/effect pair represents a single outcome of a non-degenerate projective measurement. The state labeled i is the i th measurement eigenstate, whereas the effect is the CPM obtained by tracing the input together with the i th eigenstate.

Proof. First, we compute the process z , determined by its generalized matrix elements, which we obtain by introducing a non-zero scalar factor and its inverse (where the inverse is indicated by a diagram inside $\{-\}^{-1}$), then using the causality equation (1) to transform into the following quantity:



Note that the scalars being inverted are indeed non-zero, by positivity of the whole interventional model. Furthermore, the rightmost diagram above consists of quantities that can be computed purely from projective measurements at all of the loci (including the identity/trivial measurement at Z).

Once we have computed the generalized matrix elements of z , we can use them to compute those of another factor of the model by a procedure we call “adjusting for z .”



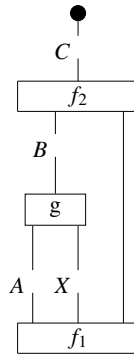
Finally, we compose the two processes at Z , leaving the X input and output, to obtain the interventional channel. □

Thus, in the quantum just as in the classical case, observation at a locus Z lying on the path between X and Y “blocks” that path and allows control of the confounding influence of u .

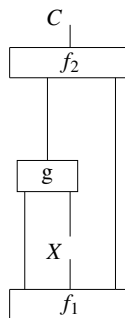
5 A more general case of a single intervention

The identification criterion of Proposition 2 can be generalized, using the same proof technique, to a quantum version of Jacobs, Kissinger, and Zanasi’s Theorem 8.1 [12].

Proposition 3. *For quantum or stochastic models of a string diagram*

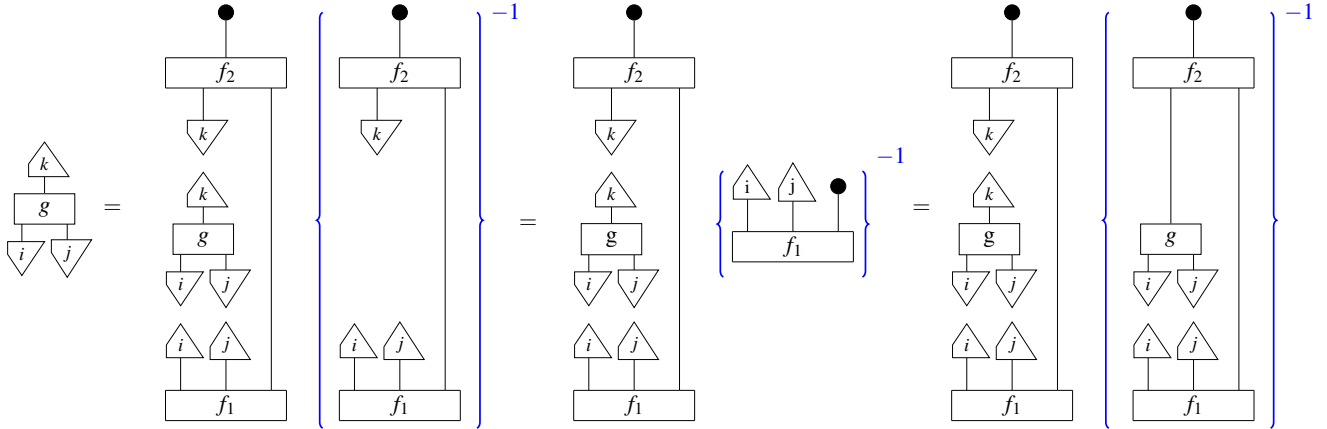


the interventional channel

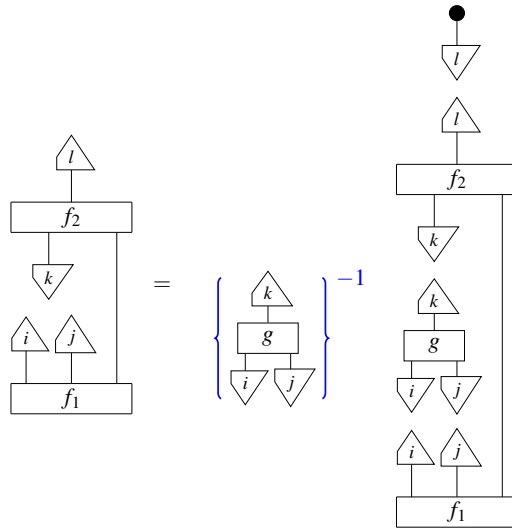


from X to C is identifiable.

Proof. First, we compute the generalized matrix elements of g , similarly to before:



Once we have the generalized matrix elements for g , we can again generate those of the outer comb by adjusting for g :



Finally, we compose the two processes at A and B , leaving the X input and output, to obtain the interventional channel. □

6 Conclusion

Functorial causal models, combined with string diagrammatic language, promise continued developments on multiple fronts of quantum and classical causal inference. Since our comb factorization roughly amounts to Tian and Pearl’s c -component factorization [18], we expect to be able to deal with more complicated quantum scenarios than the ones presented here, by porting classical identification protocols based on c -component factorization through our process-theoretic formalism to the category of quantum models. Moreover, in both the quantum and classical settings, understanding causal inference as invoking a process theory’s property of local process tomography unlocks the potential for immediately applying the abstract techniques of this article to inference with data from more general instruments than projective measurement or classical passive observation. While on the classical side both sorts of

generalization—to more complicated network shapes and to other data-collection instruments—may first lead simply to more efficient presentations of existing theory, all developments on the quantum side will constitute new domain knowledge.

Because graphs representing causal structure in other literature are often taken to encode stronger assumptions about complete common causes than are expressible in our framework, some identifiability conditions based on tests of such graphs do not have analogues in terms of the diagrams used in this article; our front-door criterion might be considered only a limited analogue of the classical set of sufficient conditions known by that name. Our diagrams' lack of explicit representation of completeness of common causes is valuable in allowing us to discern which classical graphical criteria do not involve considerations of independence of multiple variables conditioned on observed complete common causes, and to derive the quantum analogues of those criteria without a treatment of quantum complete common causes. Future work, however, will extend the framework here to incorporate assumptions of completeness of observed common causes, with a view to unified process-theoretic description of those parts of classical and quantum causal inference that rely on such assumptions.

Acknowledgements. The authors would like to thank Rob Spekkens, Jon Barrett, Frederick Eberhardt, and Sally Shrapnel for useful discussions about causal structures, observation, common causes, and control of confounding. AK acknowledges support of Grant No. 61466 and No. 62312 from the John Templeton Foundation as part of the QISS project. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation. IF acknowledges support from a Future of Humanity Institute DPhil Scholarship.

References

- [1] John-Mark A. Allen, Jonathan Barrett, Dominic C. Horsman, Ciarán M. Lee & Robert W. Spekkens (2017): *Quantum Common Causes and Quantum Causal Models*. *Phys. Rev. X* 7(3), p. 031021, doi:10.1103/PhysRevX.7.031021.
- [2] Jonathan Barrett, Robin Lorenz & Ognian Oreshkov (2019): *Quantum Causal Models*, doi:10.48550/ARXIV.1906.10726.
- [3] G. Chiribella, G. M. D'Ariano & P. Perinotti (2008): *Quantum Circuit Architecture*. *Phys. Rev. Lett.* 101(6), p. 060401, doi:10.1103/PhysRevLett.101.060401. Publisher: American Physical Society.
- [4] Kenta Cho & Bart Jacobs (2019): *Disintegration and Bayesian inversion via string diagrams*. *Mathematical Structures in Computer Science* 29(7), pp. 938–971, doi:10.1017/S0960129518000488. Publisher: Cambridge University Press.
- [5] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.
- [6] Bob Coecke & Robert W. Spekkens (2012): *Picturing classical and quantum Bayesian inference*. *Synthese* 186(3), pp. 651–696, doi:10.1007/s11229-011-9917-5.
- [7] Juan Correa & Elias Bareinboim (2020): *A Calculus for Stochastic Interventions: Causal Effect Identification and Surrogate Experiments*. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(06), pp. 10093–10100, doi:10.1609/aaai.v34i06.6567.
- [8] Juan D. Correa, Jin Tian & Elias Bareinboim (2019): *Identification of Causal Effects in the Presence of Selection Bias*. *Proceedings of the AAAI Conference on Artificial Intelligence* 33(01), pp. 2744–2751, doi:10.1609/aaai.v33i01.33012744. Section: AAAI Technical Track: Knowledge Representation and Reasoning.

- [9] Fabio Costa & Sally Shrapnel (2016): *Quantum causal modelling*. *New Journal of Physics* 18(6), p. 063032, doi:10.1088/1367-2630/18/6/063032. Publisher: IOP Publishing.
- [10] Tobias Fritz (2020): *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics*. *Advances in Mathematics* 370, p. 107239, doi:10.1016/j.aim.2020.107239.
- [11] David Galles & Judea Pearl (1995): *Testing Identifiability of Causal Effects*. In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 185–195, doi:10.5555/2074158.2074180. Event-place: Montréal, Qué, Canada.
- [12] Bart Jacobs, Aleks Kissinger & Fabio Zanasi (2019): *Causal Inference by String Diagram Surgery*. In Mikołaj Bojańczyk & Alex Simpson, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 313–329, doi:10.1007/978-3-030-17127-8_18.
- [13] Bart Jacobs & Fabio Zanasi (2020): *The Logical Essentials of Bayesian Reasoning*, p. 295–332. Cambridge University Press, doi:10.1017/9781108770750.010.
- [14] Judea Pearl (1995): *Causal diagrams for empirical research*. *Biometrika* 82(4), pp. 669–688, doi:10.1093/biomet/82.4.669.
- [15] Judea Pearl (2009): *Causality*, 2nd edition. Cambridge University Press, doi:10.1017/CBO9780511803161.
- [16] Katja Ried, Megan Agnew, Lydia Vermeyden, Dominik Janzing, Robert W. Spekkens & Kevin J. Resch (2015): *A quantum advantage for inferring causal structure*. *Nature Physics* 11(5), pp. 414–420, doi:10.1038/nphys3266.
- [17] Ángel Rivas, Susana F. Huelga & Martin B. Plenio (2014): *Quantum non-Markovianity: characterization, quantification and detection*. *Reports on Progress in Physics* 77(9), p. 094001, doi:10.1088/0034-4885/77/9/094001. Publisher: IOP Publishing.
- [18] Jin Tian & Judea Pearl (2002): *A General Identification Condition for Causal Effects*. In: *Eighteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, USA, p. 567–573.
- [19] Christopher J. Wood & Robert W. Spekkens (2015): *The lesson of causal discovery algorithms for quantum correlations: causal explanations of Bell-inequality violations require fine-tuning*. *New Journal of Physics* 17(3), p. 033002, doi:10.1088/1367-2630/17/3/033002. Publisher: IOP Publishing.

Architecture-Aware Synthesis of Phase Polynomials for NISQ Devices

Arienne Meijer-van de Griend^{1,*} Ross Duncan^{1,2,†}

¹ *Quantinuum*

Terrington House, 13-15 Hills Road, Cambridge CB2 1NL, United Kingdom

² *Department of Computer and Information Sciences*

University of Strathclyde

26 Richmond Street, Glasgow, United Kingdom

We propose a new algorithm to synthesise quantum circuits for phase polynomials, which takes into account the qubit connectivity of the quantum computer. We focus on the architectures of currently available NISQ devices. Our algorithm generates circuits with a smaller CNOT depth than the algorithms currently used in `Staq` and `t|ket>`, while improving the runtime with respect the former.

1 Introduction

Many current quantum computing architectures have restricted qubit connectivity, meaning that interactions between qubits are only possible when the physical qubits are adjacent in a certain graph, henceforth called the *architecture*, defined by the design of the quantum hardware. Traditional compiling techniques for quantum circuits work around this limitation by inserting additional SWAP gates into the circuit to move the logical qubits into a location where the desired interaction is physically possible, a process called *routing* or *mapping* [6, 16, 19, 17]. This typically increases the depth and gate count of the circuit by a multiplicative factor between 1.5 and 3 [6]. However, recent work by Kissinger and Meijer-van de Griend [11] has shown that for pure CNOT circuits it is possible to compile a circuit directly to an architecture without dramatically increasing the number of CNOT gates. Their approach was to use a higher-level representation of the desired unitary transform and (re)synthesise the corresponding circuit in an architecture-aware manner.

In this paper, we consider another class of high-level constructs called *phase polynomials*, which give rise to circuits containing only CNOT and $R_Z(\theta)$ gates. The current state-of-the-art algorithm for phase polynomial synthesis is the GraySynth algorithm [1]. Unlike other algorithms for phase polynomial synthesis [3], GraySynth attempts to minimise the number of 2-qubit gates. Unfortunately, GraySynth assumes unrestricted qubit connectivity. This limitation was removed by Nash et al. [13], by adding qubit permutation subcircuits whenever a sequence of CNOTs required by GraySynth is not permitted by the architecture. Nevertheless, the algorithm still relies on the same recursive strategy as GraySynth, which might be suboptimal for sparse architectures.

In this paper we propose a new algorithm for the architecture-aware synthesis of phase polynomial circuits. The algorithm has been tuned for the relatively sparse connectivity graphs of current quantum computers.

*ariannemeijer@gmail.com

†ross.duncan@quantinuum.com

We compare our algorithm against two compilers that are able to natively synthesise phase polynomials: $t|\text{ket}\rangle$ [15] and Staq [2]. We compare the different methods based on the final CNOT count, final CNOT depth, and their runtime. These figures of merit are appropriate for noisy-intermediate scale quantum (NISQ) devices [14], since the single-qubit gates of such devices typically have error rates an order of magnitude less than that of the two qubit gates. By minimising the CNOT count we are minimising the exposure of our computation to gate error, including crosstalk; by minimising depth we reduce its exposure to decoherence.

We show that for sufficiently sparse quantum computer architectures and sufficiently large phase polynomials, our algorithm outperforms the algorithm from Nash et al. [13] that is used in Staq [2] as well as the decomposition and routing strategies from $t|\text{ket}\rangle$ [6]. Our algorithm relies on finding non-cutting vertices in the connectivity graph, and does not require computing any Steiner trees; we find that in most cases our algorithm has reduced runtime compared to that of Nash et al.

In section 2, we introduce phase polynomials and existing methods for their synthesis, both with and without architecture-awareness. Our new algorithm is described in section 3 and our experimental results can be found in section 4. Throughout the paper we will assume some familiarity with the ZX-calculus [4], which we use as notation. For the uninitiated, Cowtan et al. [7] give a short introduction to the calculus, including the phase gadget notation; Coecke and Kissinger provide a complete treatment [5].

Notation We use bold face letters x, y , to denote vectors, and the corresponding regular weight letters x_i, y_j to denote their components.

2 Phase polynomial synthesis

Following Amy et al. [1], we define the phase polynomial via the sum-over-paths formalism [8].

Definition 2.1. Let C be a circuit consisting of only CNOT and $R_Z(\theta)$ gates; then its corresponding unitary matrix U_C has a *sum-over-paths* form,

$$U_C = \sum_{x \in \mathbb{F}_2^n} e^{2\pi i f(x)} |Ax\rangle \langle x| \quad (1)$$

consisting of a *phase polynomial*

$$f(x) = \sum_{y \in \mathbb{F}_2^n} \hat{f}(y) \cdot (x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n) \quad (2)$$

with *Fourier coefficients* $\hat{f}(y) \in \mathbb{R}$, and a *basis transform* $A \in GL(n, \mathbb{F}_2)$. When no confusion will arise we refer to the pair (f, A) as the phase polynomial of C .

Note that parity functions – henceforth just called *parities* – of the form $x \mapsto (x_1 y_1 \oplus \cdots \oplus x_n y_n)$ as in Equation 2, can be identified with the bit string y ; these are the basis of the space of phase polynomials. Those parities for which $\hat{f}(y) \neq 0$ are called the *support* of f .

Every circuit over $\{\text{CNOT}, R_Z(\theta)\}$ has a canonical sum-over-paths form, which we now sketch. First, we associate a parity to each “wire segment” of the circuit as follows: the inputs of the circuit are labelled x_1, \dots, x_n respectively; the output of an R_Z gate has the same parity as its input; and a CNOT gate with parities p_1 and p_2 on its control and target inputs has output parities p_1 and $p_1 \oplus p_2$, respectively. Second, the coefficients $\hat{f}(y)$ are computed by summing all the angles θ occurring in R_Z gates labelled by the

parity y . Finally, the linear transform A is defined by the mapping $x \mapsto x'$ where x' are the final labels of circuit outputs. We refer the reader to Amy et al. [1] for more details.

The task of *phase polynomial synthesis* is the reverse: given (f, A) we must find the circuit C . This amounts to constructing a parity labelled CNOT circuit such that every y in the support of f occurs as a label on some wire, adding an $R_Z(\hat{f}(y))$ gate on that wire, and extending that circuit so that the desired output parities for A are achieved. Since $f(x)$ is a sum, and addition is commutative, the order in which the parities are achieved is irrelevant; neither does it matter on which qubits these parities occur. To obtain the required final parities, additional CNOTs are added to the circuit. Since the new parity is the sum of the parities of both the control and the target qubit, applying a CNOT gate can therefore be seen as an elementary row operation on the matrix $x \mapsto x'$. If the desired parities for each qubit are known, Gaussian elimination can produce a CNOT sequence to achieve those parities [12, 11, 13]. This method suffices to synthesise the matrix A of the phase polynomial [3, 1, 13]; note, however, that this second phase is totally independent of the earlier synthesis of the parities required for $f(x)$.

Architecture agnostic synthesis. Phase polynomials may be synthesised via the *phase gadget* construct of the ZX-calculus [7]. Since our algorithm can be intuitively described using phase gadgets, we will briefly explain this method.

Definition 2.2. In ZX-calculus notation we denote the R_Z gate with phase α and CNOT gate as:

$$\boxed{Z(\alpha)} \simeq \text{---} \bigcirc(\alpha) \text{---} \quad \text{---} \bigoplus \text{---} \simeq \text{---} \bigcirc \text{---} \text{---} \bigcirc \text{---}$$

In a phase polynomial (f, A) , each term in $f(x)$ defines an operator $e^{-i\frac{\alpha}{2}Z^{\otimes n}}$, which we represent by the *phase gadget* $\Phi_n(\alpha)$:

$$\Phi_n(\alpha) := \begin{array}{c} \text{---} \bigcirc \text{---} \\ \text{---} \bigcirc \text{---} \\ \vdots \\ \text{---} \bigcirc \text{---} \end{array} \text{---} \bigcirc(\alpha) \text{---}$$

where $\alpha = \hat{f}(y)$ and the gadget is connected to qubit i iff $x_i y_i = 1$.

Lemma 2.3. We have the following law for decomposition of phase gadgets [7].

$$\begin{array}{c} \text{---} \bigcirc(\alpha) \text{---} \\ \text{---} \bigcirc \text{---} \\ \text{---} \bigcirc \text{---} \\ \vdots \\ \text{---} \bigcirc \text{---} \end{array} = \begin{array}{c} \text{---} \bigcirc(\alpha) \text{---} \\ \text{---} \bigcirc \text{---} \\ \text{---} \bigcirc \text{---} \\ \vdots \\ \text{---} \bigcirc \text{---} \end{array} \quad (3)$$

$$\begin{array}{c} \text{---} \bigcirc(\alpha) \text{---} \\ \text{---} \bigcirc \text{---} \\ \vdots \\ \text{---} \end{array} = \text{---} \bigcirc(\alpha) \text{---} \quad (4)$$

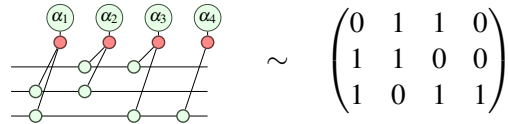
Lemma 2.3 serves as a recursive definition of the phase gadget, and demonstrates how the gadget may be realised as two ladders of CNOTs and an R_Z gate. Cowtan et al. [7] showed how to synthesise phase gadgets in reduced depth using a balanced tree of CNOTs, however if the gadgets are synthesised singly, and their ordering is not taken into account, the circuit may still be suboptimal even after local optimisation.

A consequence of Lemma 2.3 is that phase gadgets stabilise CNOT circuits in the following sense. Let C_{ij} be a CNOT gate with control qubit i and target qubit j ; then for all phase gadgets $\Phi(\alpha)$ there exists $\Phi'(\alpha)$ such that $C_{ij}\Phi(\alpha) = \Phi'(\alpha)C_{ij}$. Φ' is identical to Φ except that Φ' is connected qubit i iff Φ is connected to exactly one of i and j .

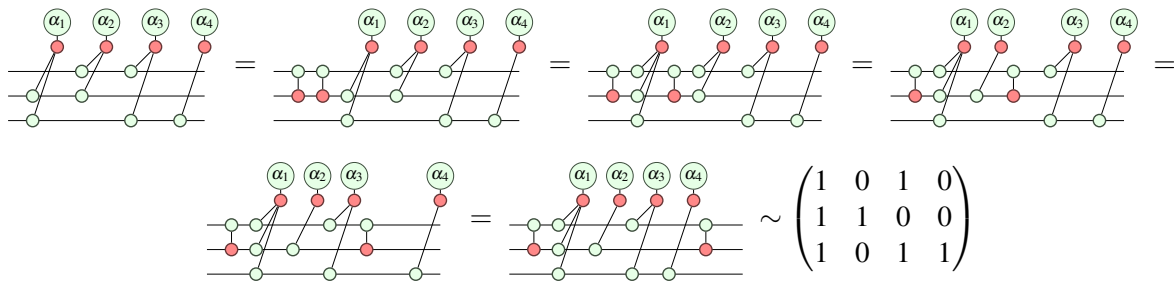
This observation leads to an improvement in the algorithm. If we view the sequence of phase gadgets as a binary matrix whose rows are the qubits and whose columns are the corresponding parities y in the

support of f , then commuting C_{ij} through the entire circuit is an elementary row operation, namely adding row j to row i . Therefore, by conjugating the circuit with CNOTs, we may obtain a column containing a single 1. At that point, the desired parity (corresponding to the column in the matrix) is achieved on the qubit corresponding to the row with the 1. The R_Z gate can then be placed, and the column can be removed from the matrix.

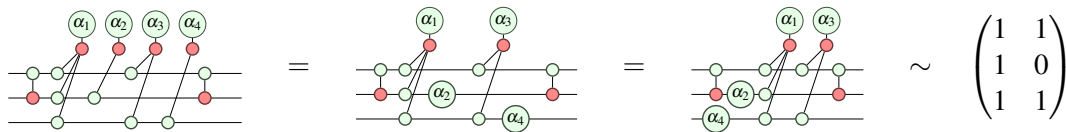
For example, the 3 qubit phase polynomial, $(f(x), I)$, specified by $f(x) = \alpha_1(x_2 \oplus x_3) + \alpha_2(x_1 \oplus x_2) + \alpha_3(x_1 \oplus x_3) + \alpha_4 x_3$, can be represented in a ZX-diagram and corresponding binary matrix as:



Conjugating the first and second qubits with two CNOTs, and applying Eq. 3 we obtain the following rewrite sequence and final matrix:



The second and last columns of the matrix contain only a single 1, so we can use Equation 4 to place a R_Z gate:



Note the equation relies on the fact that R_Z gates commute with phase gadgets.

The matrix representation reduces the task of phase polynomial synthesis to finding the order in which to reduce the columns, and which qubit should remain a 1 in the matrix for each column. Amy et al. [1] proposed a heuristic algorithm called *GraySynth* based on Gray codes. The main idea is to pick the qubit q participating in most parities and then achieving all parities containing q in order of Gray codes [9] on qubit q . As a result, many CNOTs will have the same target qubit. This algorithm has been implemented as part of *Staq* [2] in combination with SWAP-based routing.

Unfortunately, *GraySynth* does not accommodate qubit connectivity restrictions, making it less useful for NISQ devices. A naive solution is to apply a generic qubit routing routine to the synthesised circuit, however this will almost always increase the size of the circuit. Luckily, there is no need to be so naive.

Architecture-aware synthesis. It is possible to define synthesis algorithms which produce circuits that immediately satisfy the constraints imposed by the quantum computer. Several algorithms such *architecture-aware synthesis* algorithms for CNOT circuits and phase polynomials have recently been proposed [11, 13]. While SWAP-based methods respect the original structure of the circuit at the level of individual gates, architecture-aware synthesis preserves only the overall unitary, and this additional

freedom allows the architectural constraints to inform the choice of which gates to generate. This concept has also been used in the Staq compiler [2], which uses the algorithms described in this section.

Kissinger et al. [11] and Nash et al. [13] independently modified the Gaussian elimination algorithm sketched above to synthesise routed CNOT circuits. They used Steiner trees to determine paths on the connectivity graph across which to simulate one or more CNOT gates. Nash et al. [13] showed that their method scales well with respect to the size and the density of the connectivity graph of the quantum computer. Kissinger et al. [11] showed that for circuits consisting only of CNOT gates their method outperformed current state-of-the-art SWAP-based methods. Wu et al. [18] have recently improved these methods with an adaptation relying on Steiner trees and non-cutting vertices.

This constrained version of Gaussian elimination, called *Steiner-Gauss*, can be used in any synthesis algorithm by replacing the original Gaussian elimination such that it routes (part of) the synthesised circuit. In particular, this can be used in the T-par algorithm [3] and in GraySynth it can be used to synthesise the matrix A .

Nash et al. [13] also proposed an adaptation of the GraySynth algorithm we called *Steiner-GraySynth*. They replaced the step in the original GraySynth algorithm that generates a small sequence of CNOTs with a step that emulates this sequence with routed CNOTs. This emulation is created using a Steiner tree over the connectivity graph with the phase qubit as root and the other qubits participating in the sequence of CNOTs as nodes. Then, a CNOT is placed for every Steiner-node in the tree and one for every edge in the Steiner tree.

For phase polynomial synthesis, this algorithm performs better than naive routing [13]. However, following GraySynth, it will place many CNOT gates with the same target qubit. If this qubit is poorly connected in the architecture, a large CNOT overhead will result. Furthermore, it requires the construction of a Steiner tree in order to route the CNOT gates. The minimal Steiner tree problem is NP-hard[10], so finding the true optimum is not feasible, but it can be approximated in polynomial time using the all-pairs shortest paths and building a spanning tree between them.

3 New natively routed heuristic algorithm

In this section, we describe a natively routed algorithm that attempts to take the architecture into account. It uses a novel heuristic which works well for sparse architecture graphs.

Pseudo-code for the algorithm is shown in Figure 1 and its sub-procedures are listed in Appendix A. A full worked example is presented in Appendix B; for ease of comparison this example is the same one treated by Amy et al. [1] using the GraySynth algorithm.

In the following, the architecture graph – that is, the connectivity map of the physical qubits – is denoted G . The phase polynomial to be synthesised, (f, A) , is represented as two binary matrices, P and A , where the columns of P are the corresponding parities y in the support of f , as explained in Section 2. By construction, the columns in P are unique and no column y has all values set to 0.

Preprocessing. The algorithm starts by synthesising phase gadgets of the form specified by Equation 4. This will remove trivial columns in P and placing their corresponding R_Z phase gates. A column y is trivial if it has exactly one index j such that $y_j = 1$. The phase gate R_Z is placed on the qubit corresponding to j and its phase α is equal to $\hat{f}(y)$. This makes sure that every column y in P contains at least two elements with value 1. Hence, each column requires at least one CNOT in order to be synthesised by Lemma 2.3.

For example, consider the phase polynomial from Section 2, we can use Equation 4 to remove the

global variables*G*, the architecture graph*Circuit*, An initially empty circuit with $|G.\text{vertices}|$ qubits*A*, The basis transform of the phase polynomial*P*, The matrix describing the support of *f**ZPhases*, The list of Z phases $\hat{f}(y)$ belonging to each parity *y* in *f***end global variables****function** BASERECURSIONSTEP(*Cols*, *Qubits*)**if** *Qubits* non-empty and *Cols* non-empty **then***H* \leftarrow InducedSubgraph(*G*, *Qubits*)*Rows* \leftarrow NonCuttingVertices(*H*)*ChosenRow* \leftarrow $\text{argmax}_{r \in \text{Rows}} \max_{x \in \mathbb{F}_2} |\{c \in \text{Cols} \text{ where } P_{r,c} = x\}|$ *Cols0*, *Cols1* \leftarrow SplitColsOnRow(*Cols*, *ChosenRow*)BaseRecursionStep(*Cols0*, *Qubits* \setminus {*ChosenRow*})OnesRecursionStep(*Cols1*, *Qubits*, *ChosenRow*)**end if****end function****function** ONESRECURSIONSTEP(*Cols*, *Qubits*, *ChosenRow*)**if** *Cols* non-empty **then***Neighbours* \leftarrow {*q* \in *Qubits* where *q* \sim *ChosenRow* in *G*}*n* \leftarrow $\text{argmax}_{q \in \text{Neighbours}} |\{c \in \text{Cols} \text{ where } P_{q,c} = 1\}|$ **if** $|\{c \in \text{Cols} \text{ where } P_{n,c} = 1\}| > 0$ **then**PlaceCNOT (*ChosenRow*, *n*)*Cols* \leftarrow ReduceColumns(*Cols*)**else**PlaceCNOT (*n*, *ChosenRow*)PlaceCNOT (*ChosenRow*, *n*)**end if***Cols0*, *Cols1* \leftarrow SplitColsOnRow(*Cols*, *ChosenRow*)BaseRecursionStep(*Cols0*, *Qubits* \setminus {*ChosenRow*})OnesRecursionStep(*Cols1*, *Qubits*, *ChosenRow*)**end if****end function****algorithm** ROUTEDPHASEPOLYSYNTH*Columns* \leftarrow ReduceColumns({0, ..., $|P.\text{columns}|$ })BaseRecursionStep(*Columns*, *G*.vertices)*Circuit*.AddGates(SteinerGauss($A * P'^{-1}$))**end algorithm**

Figure 1: Algorithm for synthesising phase polynomials in an architecture aware manner. The subroutines not defined here are in listed in Appendix A.

fourth column (corresponding to α_4) and synthesise the phase gate $R_Z(\alpha_4)$ on qubit 3:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \sim \begin{array}{c} \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \\ \text{---} \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \text{---} \end{array} = \begin{array}{c} \alpha_1 \quad \alpha_2 \quad \alpha_3 \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \alpha_4 \text{---} \text{---} \end{array} \sim \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Base recursion step. Similar to GraySynth, we want to synthesise the phase gadgets in the phase polynomial in an order that requires the least amount of CNOT gates. However, we do not want to synthesise the phase gadgets such that many phase gates are placed on the same qubit. Instead, we pick one qubit and attempt to remove its row from P . However, we cannot pick just any row to remove from P because it might still be needed to synthesise other phase gadgets due to the connectivity constraints. Thus, we pick a non-cutting vertex $i \in G$ such that row P_i has either the most ones or the most zeroes. A *non-cutting vertex* is a vertex in G that can be removed from G without disconnecting the remaining graph. Like GraySynth, we split P into two matrices, P^0 and P^1 , such that column P_j is a column in P^0 iff $P_{i,j} = 0$ and P_j is a column in P^1 otherwise. Since all entries in row P_i^0 are equal to 0, we do not need this row any more and we can remove it from P^0 , and because i is non-cutting, its removal leaves the graph connected. Then, we use the base recursion step on the sub-matrix P^0 (excluding row i) with the sub-graph of G where vertex i has been removed. The matrix P^1 is treated by a different recursive procedure using the full graph G , described below.

Continuing the example above, suppose we are targeting the architecture $G : x_1 \Leftrightarrow x_2 \Leftrightarrow x_3$. We can pick either x_1 or x_3 as they are both non-cutting and have the same number of ones and zeroes; we will make the arbitrary choice of x_1 . This choice yields our new P^0 and P^1 :

$$P = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad P^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad P^1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

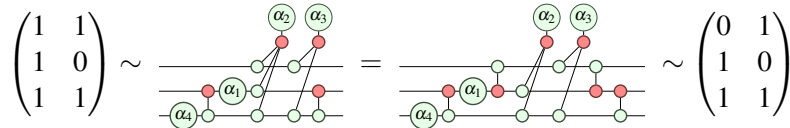
Note that P^0 corresponds to the phase gadget α_1 , and P^1 corresponds to the phase gadgets α_2 and α_3 . Recursing on P^0 will eventually place the CNOT $C_{3,2}$ and $R_Z(\alpha_1)$ gate on qubit x_2 , as shown below.

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \sim \begin{array}{c} \alpha_1 \quad \alpha_2 \quad \alpha_3 \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \alpha_4 \text{---} \text{---} \end{array} = \begin{array}{c} \alpha_1 \quad \alpha_2 \quad \alpha_3 \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \alpha_4 \text{---} \text{---} \end{array} = \begin{array}{c} \alpha_2 \quad \alpha_3 \\ \text{---} \text{---} \\ \text{---} \text{---} \\ \alpha_4 \quad \alpha_1 \text{---} \text{---} \end{array} \sim \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Bear in mind that the recursion on P^0 may add CNOTs to the circuit, performing a row operation on the global P matrix. For our recursion scheme to be valid we require that the row P_i^1 remains equal to 1. Initially, this holds by the construction of P^1 . Since row i has been removed from P^0 , no gate involving qubit i will be added by recursion on P^0 , and hence the i th row of P^1 will be unchanged. Moreover, P^1 does not contain any trivial columns because for every column j in P^1 there will always be another row $k \neq i$ such that $P_{k,j}^1 = 1$.

Ones recursion step. The recursion step for P^1 attempts to remove as many ones from row i as possible such that it can be removed. This can be achieved by placing CNOTs in the circuit, however we are restricted by the connectivity graph. Therefore, we pick a neighbour vertex $n \in G$ such that row P_n^1 has most ones. Picking the row P_n^1 with most ones will ensure that most ones are removed. Then, we can conjugate with CNOT $C_{i,n}$, and update P by adding row n to row i , as explained in Section 2. This

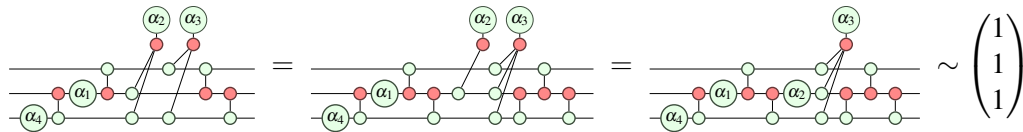
might introduce trivial columns in P^1 (note that $P^0 = \emptyset$ by the recursion), which are removed like in the preprocessing step. Thus, in the example circuit:



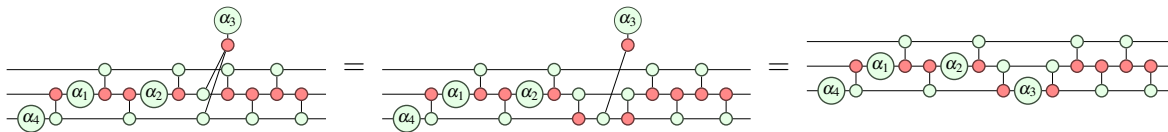
However, if every entry of row P_n^1 is 0, conjugation with $C_{i,n}$ will have no effect. In this situation, we first apply the opposite CNOT, $C_{n,i}$ and then $C_{i,n}$ as before. This effectively swaps the rows i and n , so there is no need to reduce the circuit. Nevertheless, this ensures that every entry in row P_i^1 is 0.

After placing the CNOT gate(s), we have modified row P_i^1 and we can split P^1 into two matrices, $P^{1,0}$ and $P^{1,1}$, and recurse upon these two as in the base recursion step.

In our example $P^{1,0} \sim \{\alpha_2\}$ and $P^{1,1} \sim \{\alpha_3\}$. We use the base recursion on $P^{1,0}$ and pick x_3 arbitrarily. Note that we only consider the sub-matrix $P^{1,0}$ to count the number of ones and zeroes. Then, we split $P^{1,0}$ into \emptyset and $\{\alpha_2\}$, respectively. In the ones recursion step, we pick neighbour x_2 and place CNOT $C_{3,2}$ and $R_Z(\alpha_2)$ on qubit x_2 .



Afterwards, we use the ones recursion step twice on the remaining row, placing two CNOTs, $C_{2,1}$ and $C_{3,2}$, and placing the final phase gate $R_Z(\alpha_3)$ on qubit x_3



Post-processing. Lastly, we need to synthesise the basis transform A . Because the CNOT gates in the circuit, obtained by synthesising the phase gadgets, change the parities on each qubit, we need to undo these changes. Let P' be the basis transform corresponding to the final parities of the synthesised circuit, then we can undo this transform and apply the desired transform A by synthesising $A \cdot P'^{-1}$ using Steiner-Gauss as explained in Section 2.

In our synthesis example, P'^{-1} is equivalent to the CNOTs that were commuted to the end of the ZX-diagram. Incidentally, $P'^{-1} = A \cdot P'^{-1}$ because of our choice $A = I$, thus the desired linear transformation is already achieved. Moreover, these trailing CNOTs are already routed, however resynthesising them might remove a few redundant CNOTs for the final circuit.

Termination and correctness. Our algorithm terminates and is correct if the recursion converges and it synthesises the desired phase polynomial (f, A) while satisfying the connectivity constraints imposed by the architecture.

At each recursion step, the matrix P is split into P^0 and P^1 . In the case of P^0 , the base recursion step will effectively remove a row from P^0 . In the case of P^1 , the ones recursion step will place one or two CNOT gates. This will either remove a column from P^1 or, when splitting P^1 into $P^{1,0}$ and $P^{1,1}$, make sure that $P^{1,0} \neq \emptyset$. The recursion finishes when P is empty. Hence, the recursion converges and the algorithm terminates.

By construction, the matrix P describes the remaining phase gadgets to be synthesised (initially the parities y in the support of f). This remains the case while synthesising because placing a CNOT updates P with an elementary row addition as explained in Section 2. Moreover, a column is only removed from P iff the phase gadget is trivial, i.e. it is of the form described by Equation 4. Consequently, the phase gates are placed at the right parity by Lemma 2.3. Lastly, the basis transform A is obtained as described in the previous paragraph. Thus, the algorithm has synthesised the desired phase polynomial once it has terminated.

Additionally, all CNOT gates that are added have the property that the control and target qubits are neighbours in the connectivity graph G , thus satisfying the connectivity constraints imposed by the architecture.

Hence, our algorithm terminates and when it does the desired phase polynomial has been synthesised in an architecture-aware manner.

4 Results and discussion

To verify the quality of our algorithm, we generated random phase polynomials and synthesised them for two different real quantum computers: Rigetti’s 16 qubit Aspen device and IBM’s 20 qubit Singapore device¹. We compare the average CNOT count, CNOT depth and runtime (in seconds) of our proposed algorithm with Staq [2] and $t|ket\rangle$ [15]. To the best of our knowledge, $t|ket\rangle$ and Staq are the only compilers that can synthesise and route phase polynomials from an abstract representation².

For each architecture, we randomly generated phase polynomials until we had 20 distinct ones with 1, 5, 10, 50, 100, 500, and 1000 phase gadgets in each. The phase gadgets were sampled uniformly across the parameter space. Figure 2 shows how each algorithm scales with respect to the number of phase gadgets on the two quantum computer architectures. Each point in the chart is the average of the 20 phase polynomials of that size.

We used pytket version 0.4.3³. We described our phase polynomials in terms of $t|ket\rangle$ ’s abstract representation for phase gadgets (PauliExpBox) which $t|ket\rangle$ synthesises and then routes using swaps [15]. While routing, we allowed $t|ket\rangle$ to also find an optimal qubit placement.

For Staq, we used version 1.0. We chose to use the Steiner tree option because this results in a much lower CNOT count and depth. Unfortunately, we were unable to use this option in combination with optimal qubit placement because this took too long for large phase polynomials (≥ 50 phase gadgets)⁴.

Note that both $t|ket\rangle$ and Staq are implemented in C++, while our algorithm was written in python 3.6, putting it at a significant runtime disadvantage. All experiments were run on a 2017 MacBook Pro with an Intel Core i5 2.3 GHz and 8 GB 2133 MHz RAM. We used pytket to calculate the CNOT count and CNOT depth of all circuits (including Staq).

We observe that for very small phase polynomials (1 phase gadget), $t|ket\rangle$ is the best method, but it does not scale well in CNOT count and depth for larger, more realistic phase polynomials (see Figure 2). This shows that naive synthesis combined with clever routing is not competitive with architecture-aware synthesis methods.

Between five and 100 phase gadgets, Staq has the lowest average CNOT count. For larger phase

¹Qubit-scaling and gadget-scaling results for synthetic architectures can be found in Appendix C

²The source code to replicate our results, including the raw experimental data, can be found on <https://github.com/CQCL/architecture-aware-phasepoly-synth>

³This pytket version will be released for the general public soon

⁴Staq results with placement for small phase polynomials can be found in Figure 5 of Appendix C

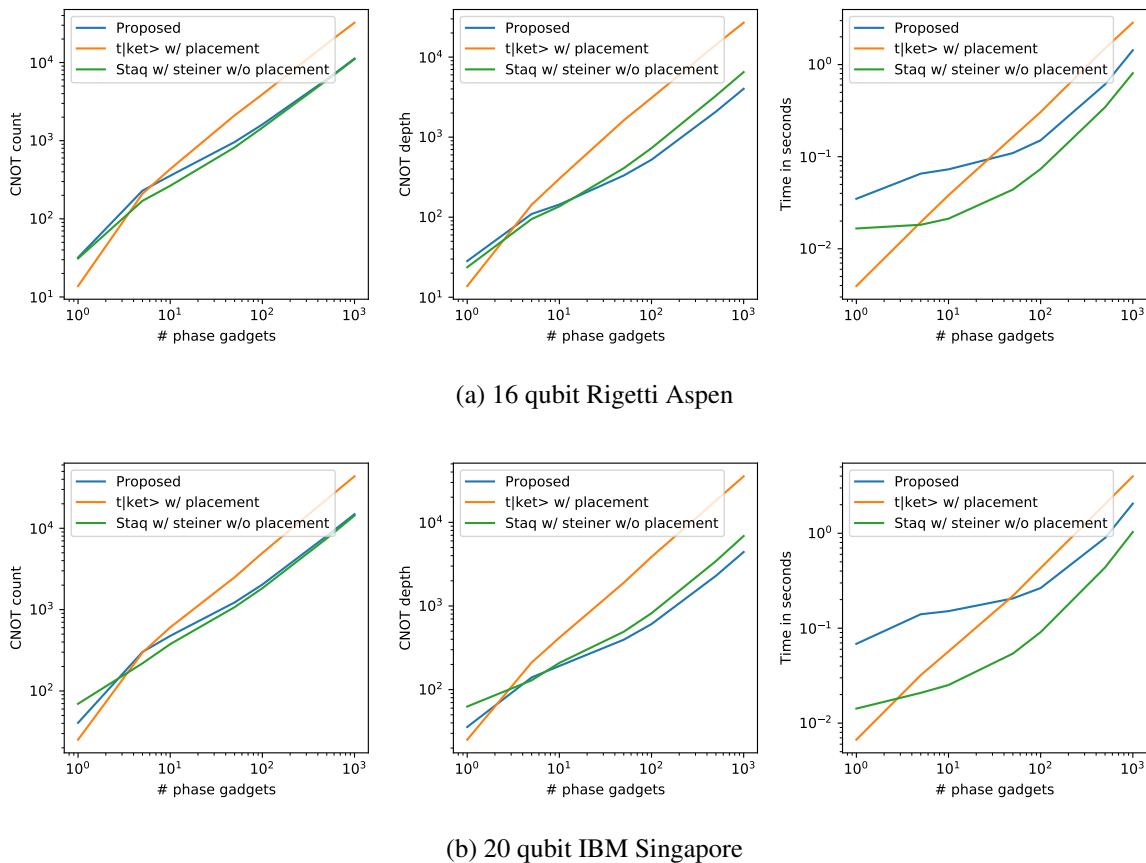


Figure 2: Plots showing the scaling of the CNOT count, CNOT depth and runtime with respect to the number of phase gadgets on the 16 qubit Rigetti Aspen architecture and the 20 qubit IBMQ Singapore device. The exact data can be found in Table 1 in Appendix C.

polynomials, Staq’s CNOT count performance is equal to the proposed algorithm. However, the CNOT depth is consistently better when synthesised with the proposed algorithm for phase polynomials with more than 10 gadgets. This means that it is better at parallelising CNOT gates than Staq.

With respect to runtime, we observe that for phase polynomials with 5-1000 phase gadgets, Staq is the fastest synthesis algorithm. The proposed algorithm is faster at synthesising than $t|ket\rangle$ for phase polynomials 50-1000 gadgets on both architectures. We do note that both Staq and the proposed algorithm does not scale linearly with respect to the number of phase gadgets, thus it might not be faster than $t|ket\rangle$ for phase polynomials with more phase gadgets than we have tested.

5 Conclusion and Future Work

In this paper, we introduced one of the first successful algorithms for architecture-aware synthesis of phase polynomials. We showed that this algorithm performs comparable or better than current state-of-the-art compilers for current NISQ devices without compromising the runtime of the algorithm.

Although our algorithm is very promising, it should still be adjusted to better fit the specification of the device that it is synthesising for. For example, the choice of placing the qubits affects the size of the synthesised circuit because the connectivity graph of a quantum computer is generally not regular. Similarly, the current algorithm improves CNOT depth, but it might do so in a way that increases the crosstalk between parallel gates.

And, lastly, our algorithm can only synthesise phase polynomials. This means that circuits containing rotations over X and Y need to be split into subcircuits to use our algorithm. It will be much more beneficial if our algorithm can be extended to also synthesise the generalised version of phase gadgets, called *Pauli exponentials*.

Acknowledgements

The authors would like to thank Alex Cowtan, John van de Wetering, and Nicolas Heurtel for helpful discussions.

References

- [1] 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.
- [2] Matthew Amy & Vlad Gheorghiu (2020): *staq—A full-stack quantum processing toolkit*. *Quantum Science and Technology* 5(3), p. 034016, doi:10.1088/2058-9565/ab9359.
- [3] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T-Depth Optimization of Clifford+T Circuits Via Matroid Partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953.
- [4] Bob Coecke (2010): *Quantum picturalism*. *Contemporary Physics* 51(1), pp. 59–83, doi:10.1080/00107510903257624.
- [5] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.
- [6] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons & Seyon Sivarajah (2019): *On the Qubit Routing Problem*. In Wim van Dam & Laura Mancinska, editors: *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019), Leibniz International Proceedings in Informatics (LIPIcs)* 135, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 5:1–5:32, doi:10.4230/LIPIcs.TQC.2019.5. Available at <http://drops.dagstuhl.de/opus/volltexte/2019/10397>.
- [7] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons & Seyon Sivarajah (2020): *Phase Gadget Synthesis for Shallow Circuits*. *Electronic Proceedings in Theoretical Computer Science* 318, pp. 213–228, doi:10.4204/eptcs.318.13.
- [8] Christopher M. Dawson, Andrew P. Hines, Duncan Mortimer, Henry L. Haselgrove, Michael A. Nielsen & Tobias J. Osborne (2005): *Quantum Computing and Polynomial Equations over the Finite Field \mathbb{Z}_2* . *Quantum Info. Comput.* 5(2), p. 102–112, doi:10.26421/QIC5.2-2.
- [9] Frank Gray (1953): *Pulse Code Communication*.
- [10] Richard M. Karp (2010): *Reducibility Among Combinatorial Problems*, pp. 219–241. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-540-68279-0_8.
- [11] Aleks Kissinger & Arianne Meijer-van de Griend (2020): *Cnot circuit extraction for topologically-constrained quantum memories*. *Quantum information & computation* 20(7-8), pp. 581–596, doi:10.26421/QIC20.7-8-4.

- [12] Patel K.N., Markov I.L. & Hayes J.P. (2008): *Optimal synthesis of linear reversible circuits*. *Quantum Information and Computation* 8(3&4), pp. 282–294, doi:10.26421/qic8.3-4-4. Available at <https://cir.nii.ac.jp/crid/1360294647045719424>.
- [13] Beatrice Nash, Vlad Gheorghiu & Michele Mosca (2020): *Quantum circuit optimizations for NISQ architectures*. *Quantum Science and Technology* 5(2), p. 025010, doi:10.1088/2058-9565/ab79b1.
- [14] John Preskill (2018): *Quantum Computing in the NISQ era and beyond*. *Quantum* 2, p. 79, doi:10.22331/q-2018-08-06-79.
- [15] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2021): *T|ket>: A Retargetable Compiler for NISQ Devices*. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92.
- [16] Mathias Soeken, Giulia Meuli, Bruno Schmitt, Fereshte Mozafari, Heinz Riemer & Giovanni De Micheli (2020): *Boolean satisfiability in quantum compilation*. *Philosophical Transactions Of The Royal Society A-Mathematical Physical And Engineering Sciences* 378(2164), p. 20190161, doi:10.1098/rsta.2019.0161. Available at <http://infoscience.epfl.ch/record/275628>.
- [17] Robert Wille, Lukas Burgholzer & Alwin Zulehner (2019): *Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations*. In: *Proceedings of the 56th Annual Design Automation Conference 2019, DAC '19*, Association for Computing Machinery, New York, NY, USA, pp. 1–6, doi:10.1145/3316781.3317859.
- [18] Bujiao Wu, Xiaoyu He, Shuai Yang, Lifu Shou, Guojing Tian, Jialin Zhang & Xiaoming Sun (2023): *Optimization of CNOT circuits on limited-connectivity architecture*. *Phys. Rev. Res.* 5, p. 013065, doi:10.1103/PhysRevResearch.5.013065.
- [19] Alwin Zulehner, Alexandru Paler & Robert Wille (2019): *An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38(7), pp. 1226–1236, doi:10.1109/TCAD.2018.2846658.

A Subfunctions for the proposed synthesis algorithm

The subfunctions that we used in the pseudocode for our algorithm (Figure 1) are listed below.

```

function REDUCECOLUMNS(Columns)
  for all  $c \in \text{Columns}$  do
    if  $|\{q \in P.\text{rows} \text{ where } P_{q,c} = 1\}| = 1$  then
       $Qubit \leftarrow \text{argmax}_{q \in P.\text{rows}} P_{q,c}$ 
       $Circuit.AddGate(R_Z(ZPhases[c], Qubit))$ 
       $Columns \leftarrow Columns \setminus \{c\}$ 
    end if
  end for
  return Columns
end function

function PLACECNOT(Control, Target)
   $Circuit.AddGate(CNOT(\text{Control}, \text{Target}))$ 
   $P[\text{Control}] \leftarrow P[\text{Control}] + P[\text{Target}]$ 
end function

function SPLITCOLSONROW(Columns, Row)
   $Cols0 \leftarrow \{c \in \text{Columns} \text{ where } P_{Row,c} = 0\}$ 
   $Cols1 \leftarrow \{c \in \text{Columns} \text{ where } P_{Row,c} = 1\}$ 
  return Cols0, Cols1
end function

```

B Example synthesis

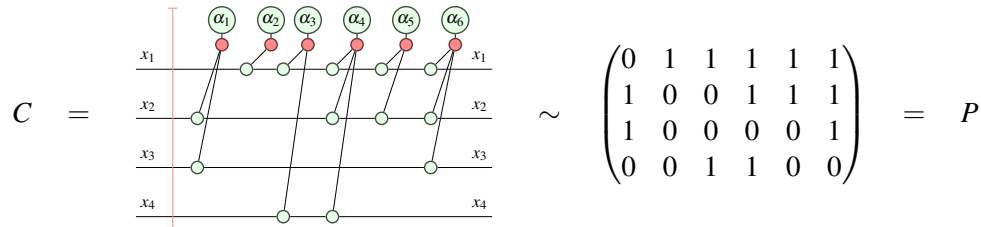
To get a better idea of the inner workings of the algorithm, we synthesise the following phase polynomial:

$$f(x) = \alpha_1(x_2 \oplus x_3) + \alpha_2(x_1) + \alpha_3(x_1 \oplus x_4) + \alpha_4(x_1 \oplus x_2 \oplus x_4) + \alpha_5(x_1 \oplus x_2) + \alpha_6(x_1 \oplus x_2 \oplus x_3)$$

$$A = I$$

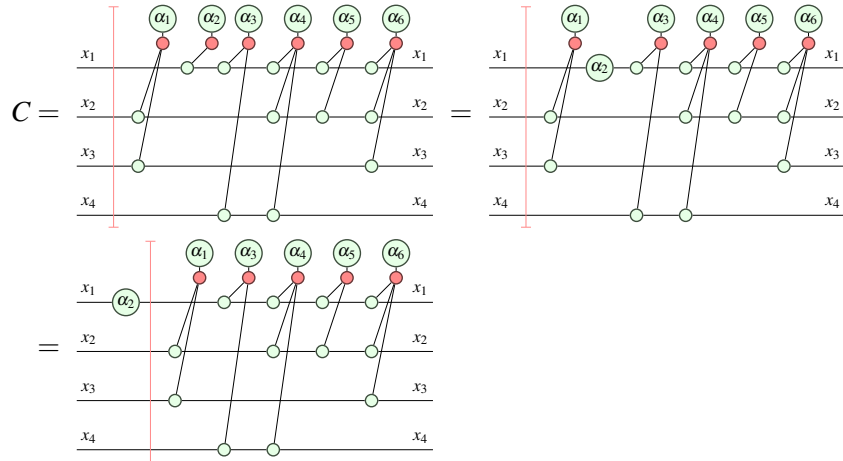
Note that this is the parameterised version of the example phase polynomial given by Amy et al.[1]. The connectivity graph we use for synthesis is a simple line architecture: $G : x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4$.

This phase polynomial corresponds to the following ZX-diagram C and matrix representation P .



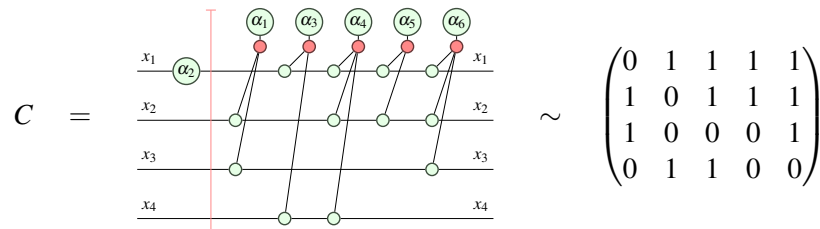
Note that the matrix P has a column for each phase gadget in the diagram and each row has a 1 iff the corresponding qubit is participating in the corresponding phase gadget (i.e. it has a green spider). We have added a red vertical line to the ZX-diagram to represent the *frontier*. This indicates the progress of our synthesis. The diagram on the left of the frontier has been synthesised, the diagram on the right of the frontier contains the phase polynomial to be synthesised. Additionally, while synthesising, we will rewrite the diagram C by adding gates to the frontier without changing the semantics of C .

Preprocessing. The first step in the algorithm is to check if any columns can be removed from the matrix. This is possible if the column contains exactly a single entry with the value 1. If this is the case, the phase gadget is only acting on a single qubit and it is equivalent to a Z phase gate which we can move to the other side of the frontier.



We describe this process as *placing a phase gate*.

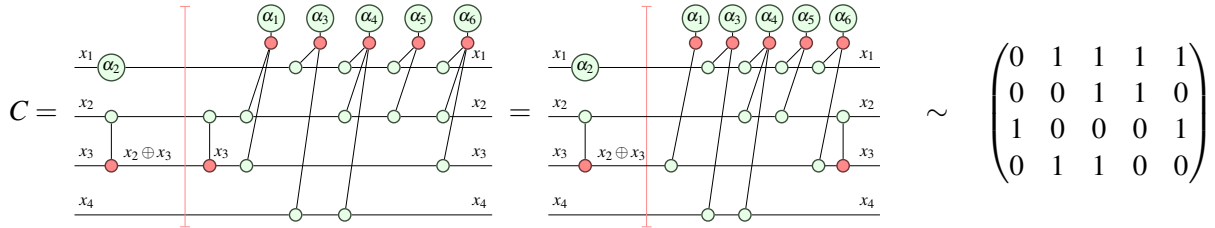
Once the phase gate $R_Z(\alpha_2)$ is placed on qubit x_1 , we have a phase gadget less, so we can remove the corresponding column from the matrix P .



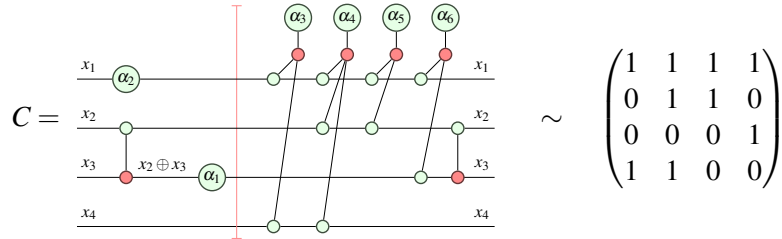
Main recursion. Now we can start the main recursion loop. We start with the base recursion step and calculate all non-cutting vertices of our graph G , which are $\{x_1, x_4\}$. We pick the row in P with either most ones or most zeroes, which is x_1 . We split the row into columns with zeroes $P^0 \sim \{\alpha_1\}$, and columns with ones $P^1 \sim \{\alpha_3, \alpha_4, \alpha_5, \alpha_6\}$. We recurse using the base recursion step on P^0 and the ones recursion step on P^1 .

In the base recursion step on P^0 , we have subgraph $G : x_2 \Leftrightarrow x_3 \Leftrightarrow x_4$, with non-cutting vertices $\{x_2, x_4\}$. We pick x_2 arbitrarily and split the matrix once more into $P^{0,0} \sim \emptyset$ and $P^{0,1} \sim \{\alpha_1\}$. This time, there are no columns with zeroes, so the base recursion step is trivial. Then, in the ones recursion step on $P^{0,1}$, we pick a neighbour of x_2 with the most ones, this is x_3 , and we place a CNOT gate, C_{x_2, x_3} , in front of the frontier. To keep the diagram equivalent to the previous diagrams, we add a second CNOT gate, C_{x_2, x_3} , after the frontier and commute it through the phase gadgets. By commuting the second CNOT gate through the gadgets, each control qubit will participate in the phase gadget iff either the control or the target qubit (exclusive) was participating before commuting the CNOT through, see Section 2 for a detailed explanation. This is the same as adding the target row to the control row in the matrix P (modulo

2). Observe that this also changes the columns in P^1 .

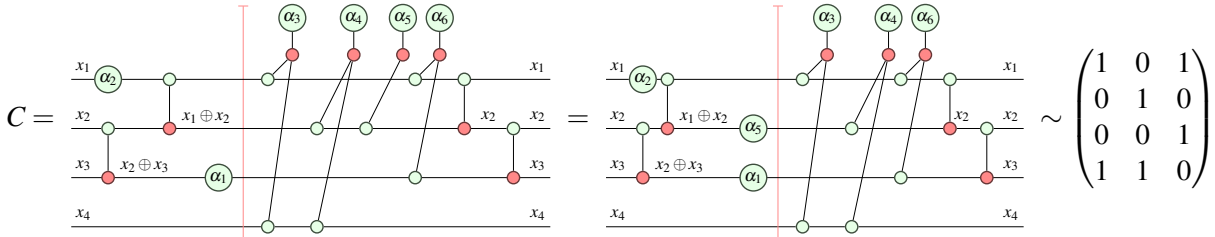


As a result, we can place a phase gate, $R_Z(\alpha_1)$, corresponding to α_1 on qubit x_3 .



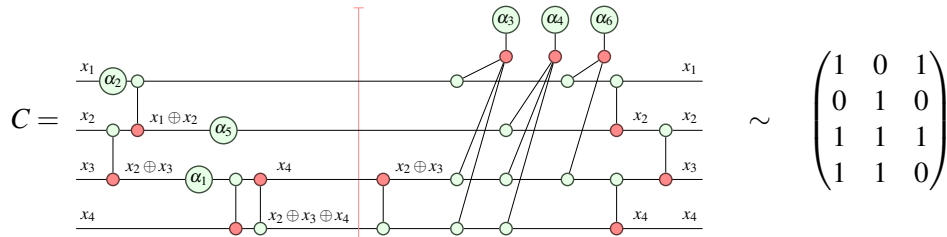
Note that placing the phase gate causes that $P^0 = \emptyset$ so splitting the row X_2 and recursing on $P^{0,0} \sim \emptyset$ and $P^{0,1} \sim \emptyset$ is trivial.

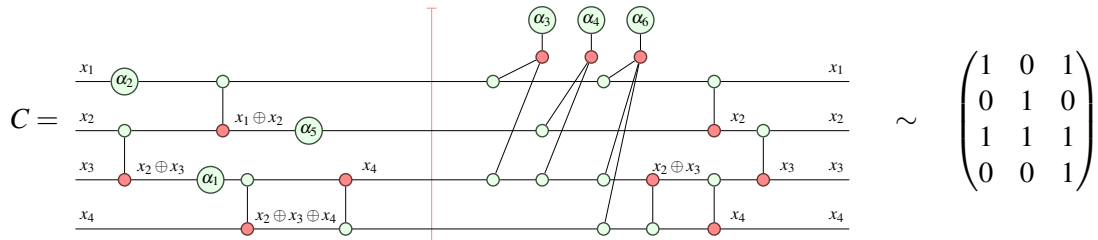
Now we are finished with the base recursion step on P^0 and continue with the ones recursion step on P^1 and the full graph G . We had chosen x_1 earlier, now we pick a neighbour, x_2 , and place the CNOT gate, C_{x_1, x_2} . This allows us to place a phase gate, $R_Z(\alpha_5)$, on qubit x_2 .



Again, we split row x_1 into columns with zeroes $P^{1,0} \sim \{\alpha_4\}$ and with ones $P^{1,1} \sim \{\alpha_3, \alpha_6\}$. We use the base recursion step on $P^{1,0}$ with the subgraph $G : x_2 \Leftrightarrow x_3 \Leftrightarrow x_4$ and we use the ones recursion step on $P^{1,1}$.

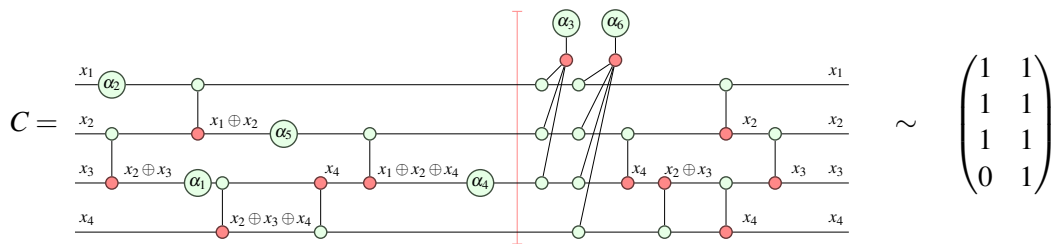
The subgraph $G : x_2 \Leftrightarrow x_3 \Leftrightarrow x_4$ has non-cutting vertices x_2 and x_4 . We pick x_4 arbitrarily and split the row into $P^{1,0,0} \sim \emptyset$ and $P^{1,0,1} \sim \{\alpha_4\}$. The base recursion step on $P^{1,0,0}$ is trivial. In the ones recursion step, we pick neighbour x_3 and place two CNOT gates, C_{x_3, x_4} , and C_{x_4, x_3} , because x_3 only has zeroes in $P^{1,0,1}$.



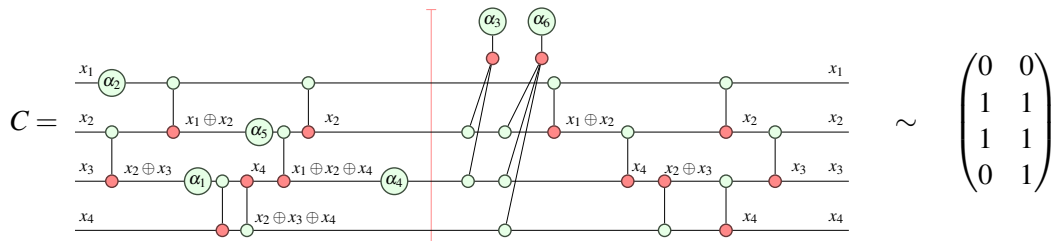


Now we split $P^{1,0,1}$ on row x_4 into $P^{1,0,1,0} \sim \{\alpha_4\}$ and $P^{1,0,1,1} \sim \emptyset$ and recurse as before, note that the latter case is trivial.

In the base recursion step on $P^{1,0,1,0}$, we are left with the subgraph $G : x_2 \Leftrightarrow x_3$. We pick row x_2 arbitrarily and split it into $P^{1,0,1,0,0} \sim \emptyset$ and $P^{1,0,1,0,1} \sim \{\alpha_4\}$. The base recursion step on $P^{1,0,1,0,0}$ is trivial and in the ones recursion step, we pick neighbour x_3 . Hence we can place a CNOT gate, C_{x_2,x_3} , and a phase gate, $R_Z(\alpha_4)$ on qubit x_3 .

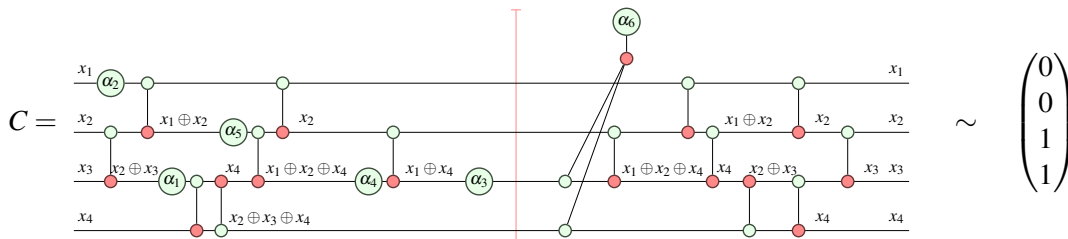


This finishes the recursion on $P^{1,0}$ and we can continue with the ones recursion step on $P^{1,1} \sim \{\alpha_3, \alpha_6\}$. Once more, we are back at the original graph $G : x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4$. We previously picked row x_1 and so we now pick neighbour x_2 . We place a CNOT gate, C_{x_1,x_2} , and split on row x_1 into $P^{1,1,0} \sim \{\alpha_3, \alpha_6\}$, and $P^{1,1,1} \sim \emptyset$.



In the base recursion step on $P^{1,1,0}$, we pick row x_2 and split $P^{1,1,0}$ into $P^{1,1,0,0} \sim \emptyset$, and $P^{1,1,0,1} \sim \{\alpha_3, \alpha_6\}$. The base recursion step on $P^{1,1,0,0}$ is trivial.

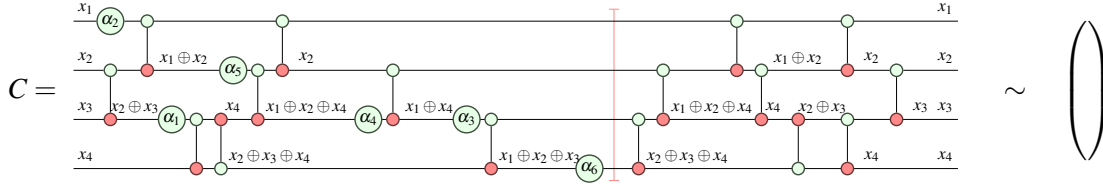
In the ones recursion step on $P^{1,1,0,1}$, we pick neighbour x_3 , and place a CNOT gate, C_{x_2,x_3} , and a phase gate, $R_Z(\alpha_3)$, on qubit x_3 .



We split $P^{1,1,0,1}$ on row x_2 , resulting in $P^{1,1,0,1,0} \sim \{\alpha_6\}$, and $P^{1,1,0,1,1} \sim \emptyset$ and we recurse as before.

In the base recursion on $P^{1,1,0,1,0}$, we are left with subgraph $G : x_3 \Leftrightarrow x_4$. We pick x_3 and split on it resulting in $P^{1,1,0,1,0,0} \sim \emptyset$ and $P^{1,1,0,1,0,1} \sim \{\alpha_6\}$. The base recursion step is trivial.

Finally, in the ones recursion step on $P^{1,1,0,1,0,1}$, we pick neighbour x_4 and place a CNOT gate, C_{x_3,x_4} and a phase gate, $R_Z(\alpha_6)$, on qubit x_4 .



Now we have synthesised every phase gadget in the support of f .

Post-processing. What remains is synthesising the basis transform $A = I$. At the frontier, the basis transform of the qubits is equal to the matrix P' ,

$$P' = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

as can be seen in the parity annotation of each qubit of the final circuit. This transform needs to be undone before the basis transform A can be applied.

As explained at the end of Section 3, this transformation is undone by the trailing CNOTs on the right of the frontier. I.e. the CNOTs on the right of the frontier apply the basis transform P'^{-1} . Although these CNOTs are already mapped, they could be optimised using an architecture-aware CNOT circuit synthesis technique, such as Steiner-Gauss. In case the matrix $A \neq I$, we can calculate the full transformation A' by undoing the existing linear transformation and then applying the desired transformation: $A' = A \cdot P'^{-1}$.

C Additional results

This appendix contains additional figures and tables to show the performance of the proposed algorithm with respect to the existing algorithms.

To show the scaling of our algorithm with respect to the number of qubits, the number of phase gadgets and the density of the device connectivity graph, we have run several experiments, generating 20 random phase polynomials per experimental setting. Since StaQ only supports a small selection of quantum computer architectures, we compare the proposed algorithm against an in-house implementation of Steiner-GraySynth for all synthetic architectures.

Figure 3 shows how our algorithm and the two baselines perform on a line, square and fully connected connectivity of various sizes given a phase polynomial with 100 phase gadgets. Similarly, Figure 4 shows how our algorithm and the two baselines perform on phase polynomials of various sizes given a 36 qubit line, square and unconstrained connectivity graph. In Figure 5, we show that, if StaQ is used with qubit placement optimisation, it can synthesise slightly smaller circuits than without qubit placement. However, this comes at an extreme runtime cost. The runtime of this option was long enough that it was not feasible for to run experiments with more than 50 and 100 gadgets (IBMQ Singapore and Rigetti

Aspen, respectively) because Staq would take more than two hours to synthesise a single circuit with 500 gadgets on Rigetti Aspen.

Lastly, the exact data that was visualised in each figure, Figure 2, 3, 4, and 5, is given in Table 1, 2, 3, and 4, respectively.

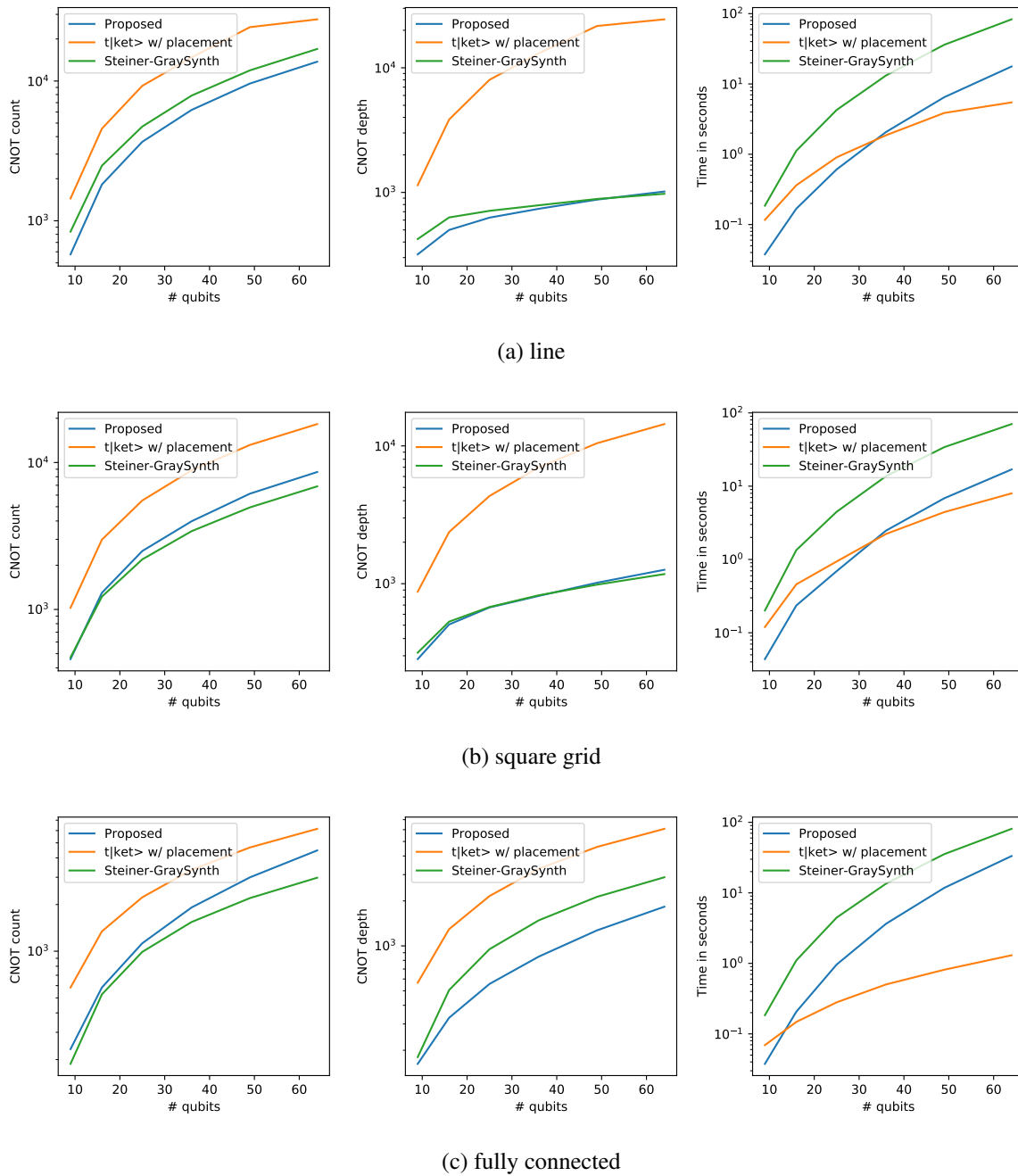
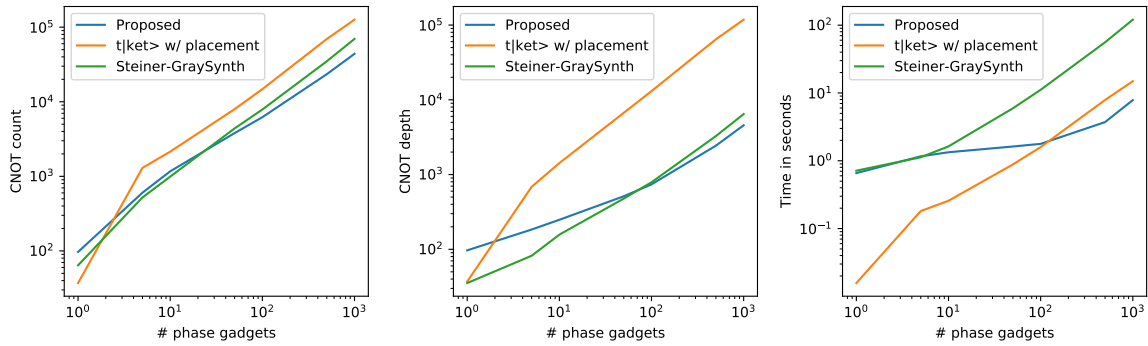
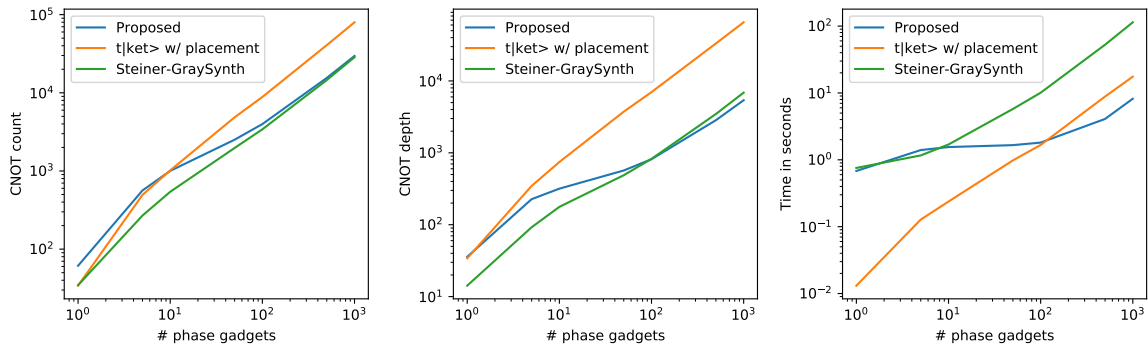


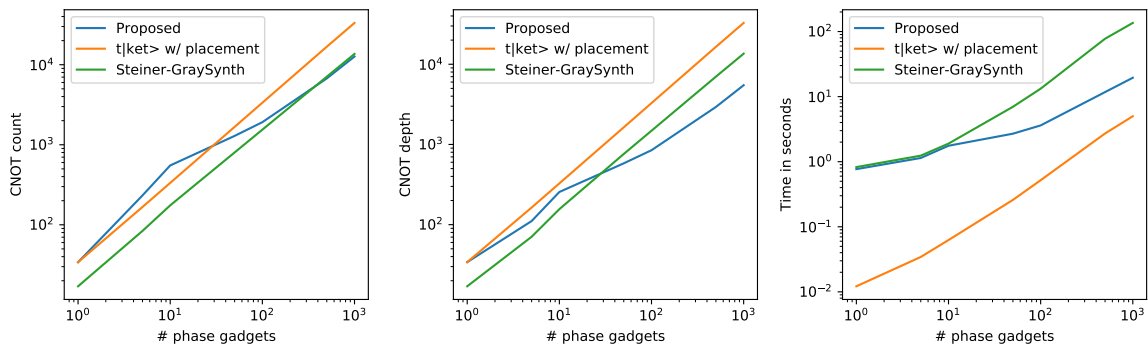
Figure 3: The influence of the number of qubits on the CNOT count, CNOT depth and runtime for architectures with different regular structures: line, square grid and fully connected. The exact data can be found in Table 2.



(a) line

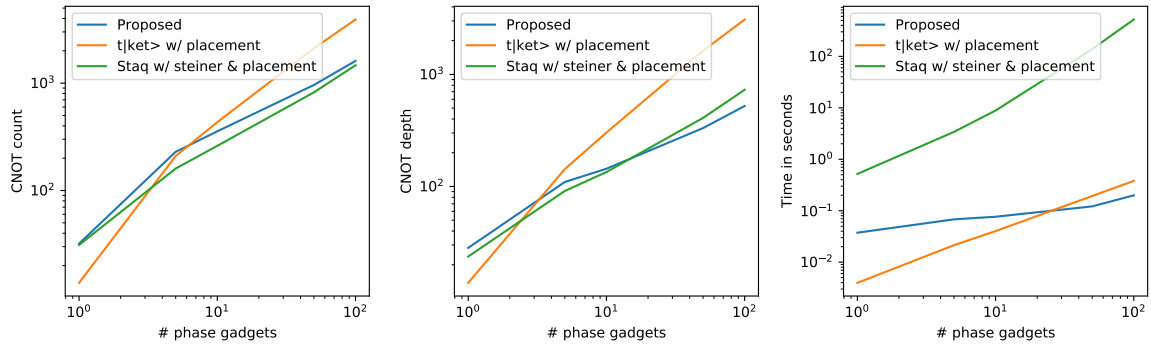


(b) square grid

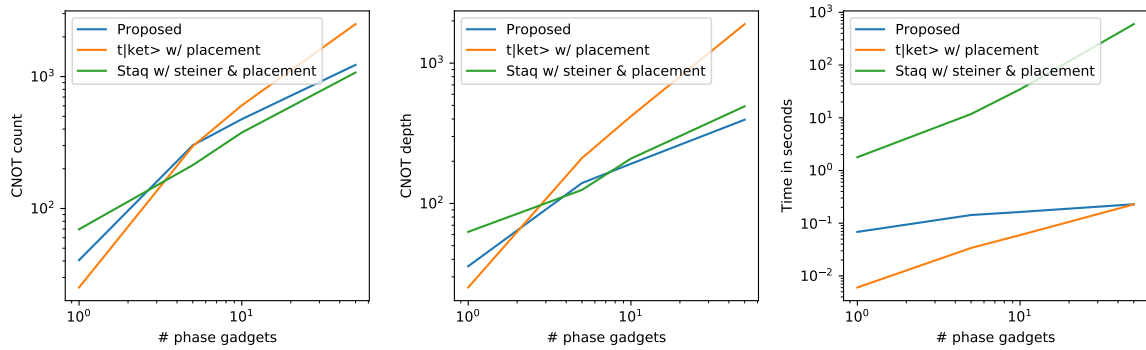


(c) fully connected

Figure 4: Plots showing the scaling of the CNOT count, CNOT depth and runtime with respect to the number of phase gadgets on a 36 qubit line, square grid, and unconstrained architecture. The exact data can be found in Table 3.



(a) 16 qubit Rigetti Aspen



(b) 20 qubit IBM Singapore

Figure 5: Plots showing the scaling of the CNOT count, CNOT depth and runtime with respect to the number of phase gadgets on the 16 qubit Rigetti Aspen architecture and the 20 qubit IBMQ Singapore device. The exact data can be found in Table 4.

$\#R_Z$	count	$t \text{ket}\rangle$ depth	time	count	Staq depth	time	count	Proposed depth	time
1	13.80	13.80	0.004s	31.10	23.70	0.017s	32.00	28.30	0.035s
5	209.55	142.95	0.020s	169.85	94.70	0.018s	229.30	109.70	0.066s
10	433.25	302.65	0.038s	264.80	136.45	0.021s	355.80	143.95	0.073s
50	2109.35	1622.45	0.164s	820.80	409.00	0.044s	961.05	331.90	0.110s
100	3917.60	3090.60	0.306s	1466.85	728.80	0.074s	1611.90	522.05	0.151s
500	17416.70	14274.65	1.506s	5928.60	3293.90	0.345s	6081.30	2082.25	0.611s
1000	32357.05	26896.25	2.851s	11043.40	6500.05	0.807s	11238.30	4018.20	1.431s

(a) Rigetti 16Q Aspen

$\#R_Z$	count	$t \text{ket}\rangle$ depth	time	count	Staq depth	time	count	Proposed depth	time
1	25.20	25.20	0.007s	69.60	62.75	0.014s	40.60	35.70	0.068s
5	296.70	210.60	0.032s	218.60	129.15	0.021s	301.60	139.75	0.140s
10	604.95	417.90	0.057s	376.50	208.60	0.025s	475.50	191.75	0.151s
50	2499.30	1897.70	0.219s	1073.25	492.40	0.054s	1226.00	395.20	0.206s
100	4969.60	3863.90	0.431s	1834.75	819.65	0.091s	2035.10	607.95	0.265s
500	22710.00	18205.90	2.033s	7498.85	3440.55	0.437s	8054.35	2293.45	0.893s
1000	43538.10	35533.30	3.983s	14309.70	6867.05	1.031s	14908.55	4422.70	2.054s

(b) IBMQ Singapore

Table 1: The average number of CNOT, CNOT depth and runtime for 20 circuits for synthesising phase polynomials with various sizes using $t|\text{ket}\rangle$, Staq (without qubit placement) and our proposed algorithm on Rigetti Aspen (Table 1a) and IBMQ Singapore (Table 1b). This data was visualised in Figure 2.

Qubits	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
9	1444.30	1142.75	0.117s	836.45	422.75	0.186s	575.40	318.65	0.038s
16	4565.50	3846.85	0.362s	2480.10	629.95	1.121s	1818.75	499.85	0.168s
25	9240.80	8004.30	0.905s	4724.60	711.55	4.269s	3673.35	627.50	0.609s
36	14761.20	13074.70	1.863s	7866.50	788.75	13.198s	6211.55	739.30	2.072s
49	24291.45	21651.35	3.867s	11920.00	886.60	36.099s	9592.70	875.40	6.484s
64	27632.10	24473.65	5.462s	16960.45	975.30	82.994s	13750.00	1017.30	17.706s

(a) Line

Qubits	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
9	1025.65	874.75	0.120s	472.15	316.10	0.201s	459.35	283.85	0.044s
16	2986.20	2372.30	0.458s	1222.50	530.50	1.342s	1299.00	505.95	0.236s
25	5514.35	4327.40	0.939s	2191.35	677.30	4.479s	2497.65	672.35	0.693s
36	8842.65	6993.00	2.227s	3409.85	824.35	13.679s	3982.25	815.60	2.475s
49	13171.00	10457.15	4.432s	4948.05	984.55	34.106s	6135.10	1016.65	6.843s
64	18259.70	14393.05	7.977s	6881.55	1174.30	70.358s	8615.05	1261.85	16.918s

(b) Square

Qubits	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
9	583.30	566.90	0.069s	187.75	180.05	0.184s	233.75	162.25	0.038s
16	1343.40	1294.60	0.148s	527.05	506.25	1.103s	583.85	330.15	0.206s
25	2227.70	2155.15	0.280s	990.10	949.85	4.463s	1125.55	555.80	0.963s
36	3351.10	3281.25	0.502s	1543.40	1483.15	13.501s	1915.45	847.85	3.642s
49	4668.50	4599.60	0.813s	2200.75	2131.25	35.417s	2994.10	1271.05	11.777s
64	6155.90	6076.20	1.303s	2978.95	2880.10	80.856s	4466.60	1829.50	33.257s

(c) Unconstrained

Table 2: The average number of CNOT, CNOT depth and runtime for 20 circuits for synthesising phase polynomials with 100 phase gadgets using t|ket), Nash and our proposed algorithm on synthetic qubit architectures of various sizes connected in a line (Table 2a), square (Table 2b) and fully connected (Table 2c). This data was visualised in Figure 3.

# R_Z	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
1	37.10	37.10	0.016s	64.15	35.40	0.714s	96.90	96.90	0.658s
5	1298.70	684.85	0.181s	521.10	82.00	1.126s	603.80	183.95	1.163s
10	2142.80	1424.30	0.257s	991.85	157.15	1.631s	1167.45	248.40	1.335s
50	7960.45	6700.05	0.884s	4391.70	477.30	5.943s	3800.50	512.50	1.624s
100	14761.20	13074.70	1.591s	7866.50	788.75	11.137s	6211.55	739.30	1.776s
500	69417.10	64493.35	7.885s	34883.95	3272.30	55.913s	23425.10	2435.75	3.718s
1000	126095.50	118035.55	14.913s	69584.50	6446.50	120.575s	43934.05	4564.45	7.853s

(a) 36 qubit line

# R_Z	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
1	34.10	34.10	0.013s	34.60	14.20	0.755s	61.30	35.80	0.681s
5	493.45	346.30	0.128s	268.80	92.15	1.164s	562.95	226.65	1.400s
10	1017.65	745.20	0.238s	541.70	177.15	1.695s	1002.75	317.45	1.552s
50	4859.05	3737.90	0.980s	1969.35	488.35	5.762s	2512.10	568.20	1.658s
100	8842.65	6993.00	1.669s	3409.85	824.35	10.080s	3982.25	815.60	1.809s
500	40734.55	33232.65	8.840s	14560.45	3469.65	52.656s	15492.85	2844.85	4.087s
1000	79821.90	65423.70	17.518s	28530.30	6876.90	113.956s	29621.30	5390.25	8.196s

(b) 36 qubit square

# R_Z	t ket)			Nash			Proposed		
	count	depth	time	count	depth	time	count	depth	time
1	34.10	34.10	0.012s	17.05	17.05	0.826s	34.10	34.10	0.769s
5	166.90	163.50	0.034s	83.80	70.65	1.239s	232.70	110.90	1.141s
10	333.80	326.95	0.063s	174.95	156.00	1.897s	551.40	255.60	1.759s
50	1673.00	1636.20	0.259s	801.60	761.30	6.981s	1290.25	581.80	2.695s
100	3351.10	3281.25	0.520s	1543.40	1483.15	13.252s	1915.45	847.85	3.611s
500	16781.80	16489.55	2.715s	7081.75	7006.85	77.596s	6741.75	2935.00	11.749s
1000	33291.10	32759.35	4.999s	13642.20	13550.90	135.732s	12660.40	5479.05	19.504s

(c) 36 qubit unconstrained

Table 3: The average number of CNOT, CNOT depth and runtime for 20 circuits for synthesising phase polynomials with various sizes using t|ket), Nash and our proposed algorithm on synthetic 36 qubit architectures connected in a line (Table 3a), square (Table 3b) and fully connected (Table 3c). This data was visualised in Figure 4.

# R_Z	$t ket\rangle$			Staq			Proposed		
	count	depth	time	count	depth	time	count	depth	time
1	13.80	13.80	0.004s	31.10	23.70	0.518s	32.00	28.30	0.037s
5	209.55	142.95	0.021s	160.20	91.20	3.407s	229.30	109.70	0.068s
10	433.25	302.65	0.040s	260.65	134.45	8.854s	355.80	143.95	0.076s
50	2109.35	1622.45	0.193s	820.80	409.00	138.448s	961.05	331.90	0.121s
100	3917.60	3090.60	0.380s	1466.85	728.80	521.102s	1611.90	522.05	0.199s

(a) Rigetti 16Q Aspen

# R_Z	$t ket\rangle$			Staq			Proposed		
	count	depth	time	count	depth	time	count	depth	time
1	25.20	25.20	0.006s	69.60	62.75	1.784s	40.60	35.70	0.068s
5	296.70	210.60	0.034s	213.30	124.95	11.770s	301.60	139.75	0.143s
10	604.95	417.90	0.060s	376.30	208.80	34.636s	475.50	191.75	0.163s
50	2499.30	1897.70	0.227s	1073.25	492.40	597.870s	1226.00	395.20	0.229s

(b) IBMQ Singapore

Table 4: The average number of CNOT, CNOT depth and runtime for 20 circuits for synthesising phase polynomials with various sizes using $t|ket\rangle$, Staq (with qubit placement) and our proposed algorithm on Rigetti Aspen (Table 4a) and IBMQ Singapore (Table 4b). This data was visualised in Figure 5.

Encoding High-level Quantum Programs as SZX-diagrams

Augustin Borgna

CNRS LORIA, Inria-MOCQUA,
Université de Lorraine
CNRS, LMF,
Université Paris-Saclay
agustin.borgna@loria.fr

Rafael Romero

CONICET, Instituto de Ciencias de la Computación
Universidad de Buenos Aires, Buenos Aires, Argentina
PEDECIBA,
Universidad de la República-MEC. Montevideo, Uruguay
lromero@dc.uba.ar

The Scalable ZX-calculus is a compact graphical language used to reason about linear maps between quantum states. These diagrams have multiple applications, but they frequently have to be constructed in a case-by-case basis. In this work we present a method to encode quantum programs implemented in a fragment of the linear dependently typed Proto-Quipper-D language as families of SZX-diagrams. We define a subset of translatable Proto-Quipper-D programs and show that our procedure is able to encode non-trivial algorithms as diagrams that grow linearly on the size of the program.

1 Introduction

The ZX calculus [18] has been used as intermediary representation language for quantum programs in optimization methods [12, 5, 3] and in the design of error correcting schemes [4]. The highly flexible representation of linear maps as open graphs with a complete formal rewriting system and the multiple extensions adapted to represent different sets of quantum primitives have proven useful in reasoning about the properties of quantum circuits.

Quantum operations are usually represented as quantum circuits composed by primitive gates operating over a fixed number of qubits. The ZX calculus has a close correspondence to this model and is similarly limited to representing operations at a single-qubit level. In this work we will focus on the Scalable ZX extension [7], which generalizes the ZX diagrams to work with arbitrary qubit registers using a compact representation. Previous work [6] has shown that the SZX calculus is capable of encoding nontrivial algorithms via the presentation of multiple hand-written examples. For an efficient usage as an intermediate representation language, we require an automated compilation method from quantum programming languages to SZX diagrams. While ZX diagrams can be directly obtained from a program compiled to a quantum circuit, to the best of our knowledge there is no efficient method leveraging the parametricity of the SZX calculus.

There exist several quantum programming languages capable of encoding high-level parametric programs [1, 11, 17]. Quipper [15] is a language for quantum computing capable of generating families of quantum operations indexed by parameters. These parameters need to be instantiated at compile time to generate concrete quantum circuit representations. Quipper has multiple formal specifications, in this work we focus on the linear dependently typed Proto-Quipper-D formalization [14, 13] to express high-level programs with integer parameters.

The contributions of this article are the following. We introduce a *list initialization* notation to represent multiple elements of a SZX diagram family composed in parallel. We formally define a fragment of Proto-Quipper-D programs that can be described as families of diagrams. Then we present a novel compilation method that encodes quantum programs as families of SZX diagrams and demonstrate the codification and translation of a nontrivial algorithm using our procedure.

In Section 2 we outline both languages and introduce the list initialization notation. In Section 3 we define the restricted Proto-Quipper-D fragment. In Section 4 we introduce the translation into SZX diagrams. Finally, in Section 5 we demonstrate an encoding of the Quantum Fourier Transform algorithm using our method. The proofs of the lemmas stated in this work can be found in Appendix E.

2 Background

We describe a quantum state as a system of n qubits corresponding to a vector in the \mathbb{C}^{2^n} Hilbert space. We may partition the set of qubits into multi-qubit *registers* representing logically related subsets. Quantum computations under the QRAM model correspond to compositions of unitary operators between these quantum states, called quantum gates. Additionally, the qubits may be initialized on a set state and measured.

High-level programs can be encoded in Quipper [15], a Haskell-like programming language for describing quantum computations. In this work we use a formalization of the language called Proto-Quipper-D [13] with support for linear and dependent types. Concrete quantum operations correspond to linear functions between quantum states, generated as a composition of primitive operations that can be described directly as a quantum circuit. Generic circuits may have additional parameters that must fixed at compilation time to produce the corresponding quantum circuit.

In Section 3 we describe a restricted fragment of the Proto-Quipper-D language containing the relevant operations for the work presented in this paper.

2.1 The Scalable ZX-calculus. The ZX calculus [18] is a formal graphical language that encodes linear maps between quantum states. Multiple extensions to the calculus have been proposed. We first present the base calculus with the grounded-ZX extension, denoted ZX_{\pm} [10], to allow us to encode quantum state measurement operations. A ZX_{\pm} diagram is generated by the following primitives, in addition to parallel and serial composition:

$$\begin{array}{ccccccc}
 n \vdots \begin{array}{c} \diagup \\ \alpha \\ \diagdown \end{array} \vdots m : n_1 \rightarrow m_1 & n \vdots \begin{array}{c} \diagdown \\ \alpha \\ \diagup \end{array} \vdots m : n_1 \rightarrow m_1 & \text{---} \square \text{---} : 1_1 \rightarrow 1_1 & \text{---} \dashv \text{---} : 1_1 \rightarrow 0_1 \\
 \text{---} : 1_1 \rightarrow 1_1 & (: 0_1 \rightarrow 2_1 &) : 2_1 \rightarrow 0_1 & \times : 2_1 \rightarrow 2_1 & \boxed{} : 0_1 \rightarrow 0_1
 \end{array}$$

where n_k represents the n -tensor of k -qubit registers, the green and red nodes are called Z and X spiders, $\alpha \in [0, 2\pi)$ is the phase of the spiders, and the yellow square is called the Hadamard node. These primitives allow us to encode any quantum operation, but they can become impractical when working with multiple qubit registers.

The SZX calculus [7, 6] is a *Scalable* extension to the ZX-calculus that generalizes the primitives to work with arbitrarily sized qubit registers. This facilitates the representation of diagrams with repeated structure in a compact manner. Carette et al. [6] show that the scalable and grounded extensions can be directly composed. We refer to the resulting SZX_{\pm} -calculus as SZX for simplicity. Bold wires in a SZX diagram are tagged with a non-negative integer representing the size of the qubit register they carry, and other generators are marked in bold to represent a parallel application over each qubit in the register. Bold spiders with multiplicity k are tagged with k -sized vectors of phases $\vec{\alpha} = \alpha_1 :: \dots :: \alpha_k$. The natural extension of the ZX generators correspond to the following primitives:

$$\begin{array}{cccc}
 n \vdots \begin{array}{c} \diagup \\ \vec{\alpha} \\ \diagdown \end{array} \vdots m : n_k \rightarrow m_k & n \vdots \begin{array}{c} \diagdown \\ \vec{\alpha} \\ \diagup \end{array} \vdots m : n_k \rightarrow m_k & \text{---} \square \text{---} : 1_k \rightarrow 1_k & \text{---} \dashv \text{---} : 1_k \rightarrow 0_0
 \end{array}$$

$$\text{---}^k : 1_k \rightarrow 1_k \quad k \left(\text{---}^{0_0} \rightarrow 2_k \quad \right) k : 2_k \rightarrow 0_0 \quad \begin{matrix} k & l \\ \diagdown & \diagup \\ & \end{matrix} : 1_k \otimes 1_l \rightarrow 1_l \otimes 1_k \quad \boxed{\text{---}} : 0_k \rightarrow 0_k$$

Wires of multiplicity zero are equivalent to the empty mapping. We may omit writing the wire multiplicity if it can be deduced by context.

The extension defines two additional generators; a *split* node to split registers into multiple wires, and a function arrow to apply arbitrary functions over a register. In this work we restrict the arrow functions to permutations $\sigma : [0 \dots k] \rightarrow [0 \dots k]$ that rearrange the order of the wires. Using the split node and the wire primitives can derive the rotated version, which we call a *gather*.

$$\begin{matrix} n+m & n \\ \swarrow & \searrow \\ & \text{---} \\ \nwarrow & \nearrow \\ m & m \end{matrix} : 1_{n+m} \rightarrow 1_n \otimes 1_m \quad \begin{matrix} n & n+m \\ \swarrow & \searrow \\ & \text{---} \\ \nwarrow & \nearrow \\ m & m \end{matrix} : 1_n \otimes 1_m \rightarrow 1_{n+m} \quad \text{---}^\sigma : 1_k \rightarrow 1_k$$

The rewriting rules of the calculus imply that a SZX diagrams can be considered as an open graph where only the topology of its nodes and edges matters. In the translation process we will make repeated use of the following reductions rules to simplify the diagrams:

$$\begin{matrix} n & n \\ \swarrow & \searrow \\ & \text{---} \\ \nwarrow & \nearrow \\ m & m \end{matrix} \begin{matrix} n & n+m \\ \swarrow & \searrow \\ & \text{---} \\ \nwarrow & \nearrow \\ m & m \end{matrix} \stackrel{(sg)}{=} \text{---}^{n+m} \quad \begin{matrix} n & n+m & n \\ \swarrow & \searrow & \swarrow \\ & \text{---} & \searrow \\ \nwarrow & \nearrow & \nearrow \\ m & m & m \end{matrix} \stackrel{(gs)}{=} \frac{n}{m}$$

We may also depict composition of gathers as single multi-legged generators. In an analogous manner, we will use a legless gather $\text{---}\ominus$ to terminate wires with cardinality zero. This could be encoded as the zero-multiplicity spider $\text{---}\ominus$, which represents the empty mapping.

Refer to Appendix A for a complete definition of the rewriting rules and the interpretation of the SZX calculus. Cf. [6] for a description of the calculus including the generalized arrow generators.

Carette et al. [6] showed that the SZX calculus can encode the repetition of a function $f : 1_n \rightarrow 1_n$ an arbitrary number of times $k \geq 1$ as follows:

$$\begin{matrix} & (k-1)n & \\ & \text{---} & \\ & \text{---}^{kn} & \text{---}^{kn} \\ & \text{---} & \\ n & & n \end{matrix} \boxed{f^k} \begin{matrix} & (k-1)n & \\ & \text{---} & \\ & \text{---}^{kn} & \text{---}^{kn} \\ & \text{---} & \\ n & & n \end{matrix} = \left(\text{---}^n \boxed{f} \text{---}^n \right)^k$$

where f^k corresponds to k parallel applications of f . With a simple modification this construction can be used to encode an accumulating map operation.

Lemma 2.1 Let $g : 1_n \otimes 1_s \rightarrow 1_m \otimes 1_s$ and $k \geq 1$, then

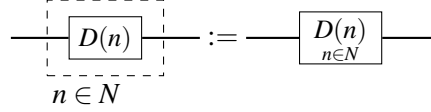
$$\begin{matrix} & kn & & (k-1)s & & km \\ & \text{---} & & \text{---} & & \text{---} \\ & \text{---}^{ks} & & \text{---}^{ks} & & \text{---} \\ & \text{---} & & \text{---} & & \text{---} \\ s & & & s & & s \end{matrix} \boxed{g^k} \begin{matrix} & kn & & (k-1)s & & km \\ & \text{---} & & \text{---} & & \text{---} \\ & \text{---}^{ks} & & \text{---}^{ks} & & \text{---} \\ & \text{---} & & \text{---} & & \text{---} \\ s & & & s & & s \end{matrix} = \begin{matrix} & kn & & n & & m & & km \\ & \text{---} & & \text{---} & & \text{---} & & \text{---} \\ & \text{---}^{ks} & & \text{---}^{ks} & & \text{---}^{ks} & & \text{---} \\ & \text{---} & & \text{---} & & \text{---} & & \text{---} \\ s & & & s & & s & & s \end{matrix} \boxed{g} \cdots \boxed{g}$$

As an example, given a list $N = [n_1, n_2, n_3]$ and a starting accumulator value x_0 , this construction would produce the mapping $([n_1, n_2, n_3], x_0) \mapsto ([m_1, m_2, m_3], x_3)$ where $(m_i, x_i) = g(n_i, x_{i-1})$ for $i \in [1, 3]$.

2.2 SZX diagram families and list instantiation. We introduce the definition of a family of SZX diagrams $D : \mathbb{N}^k \rightarrow \mathcal{D}$ as a function from k integer parameters to SZX diagrams. We require the structure of the diagrams to be the same for all elements in the family, parameters may only alter the wire tags and spider phases. Partial application is allowed, we write $D(n)$ to fix the first parameter of D .

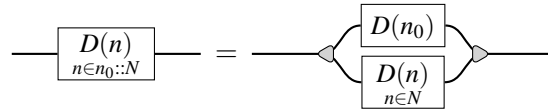
Since instantiations of a family share the same structure, we can compose them in parallel by merging the different values of wire tags and spider phases. We introduce a shorthand for instantiating a family of diagrams on multiple values and combining the resulting diagrams in parallel. This definition is strictly

more general than the *thickening endofunctor* presented by Carette et al. [6], which replicates a concrete diagram in parallel. A *list instantiation* of a family of diagrams $D : \mathbb{N}^{k+1} \rightarrow \mathcal{D}$ over a list N of integers is written as $(D(n), n \in N)$. This results in a family with one fewer parameter, $(D(n), n \in N) : \mathbb{N}^k \rightarrow \mathcal{D}$. We graphically depict a list instantiation as a dashed box in a diagram, as follows.

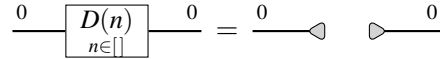


The definition of the list instantiation operator is given recursively on the construction of D in Figure 1. On the diagram wires we use $v(N)$ to denote the wire cardinality $\sum_{n \in N} v(n)$, $\vec{\alpha}(N)$ for the concatenation of phase vectors $\vec{\alpha}(n_1) :: \dots :: \vec{\alpha}(n_m)$, and $\sigma(N)$ for the composition of permutations $\otimes_{n \in N} \sigma(n)$. In general, a permutation arrow $\sigma(N, v, w)$ instantiated in concrete values can be replaced by a reordering of wires between two gather gates using the rewrite rule **(p)**.

Lemma 2.2 For any diagram family $D, n_0 : \mathbb{N}, N : \mathbb{N}^k$,



Lemma 2.3 A diagram family initialized with the empty list corresponds to the empty map. For any diagram family D ,



Lemma 2.4 The list instantiation procedure on an n -node diagram family adds $\mathcal{O}(n)$ nodes to the original diagram.

3 The λ_D calculus

We first define a base language from which to build our translation. In this section we present the calculus λ_D , as a subset of the strongly normalizing Proto-Quipper-D programs. Terms are inductively defined by:

$$\begin{aligned}
M, N, L := & x \mid C \mid R \mid U \mid 0 \mid 1 \mid n \mid \text{meas} \mid \text{new} \mid \lambda x^S.M \mid M N \mid \lambda' x^P.M \mid M @ N \mid \\
& \star \mid M \otimes N \mid \text{let } x^{S_1} \otimes y^{S_2} = M \text{ in } N \mid M; N \mid \\
& \text{VNil}^A \mid M :: N \mid \text{let } x^S :: y^{\text{vec } n S} = M \text{ in } N \\
& M \square N \mid \text{ifz } L \text{ then } M \text{ else } N \mid \text{for } k^P \text{ in } M \text{ do } N
\end{aligned}$$

Where C is a set of implicit bounded recursive primitives used for operating with vectors and iterating functions. $n \in \mathbb{N}$, $\square \in \{+, -, \times, /, \wedge\}$ and $\text{ifz } L \text{ then } M \text{ else } N$ is the conditional that tests for zero.

Here U denotes a set of unitary operations and R is a phase shift gate with a parametrized angle. In this article we fix the former to the CNOT and Hadamard (H) gates, and the latter to the arbitrary rotation gates $R_{z(\alpha)}$ and $R_{x(\alpha)}$.

For the remaining constants, 0 and 1 represent bits, new is used to create a qubit, and meas to measure it. \star is the inhabitant of the `Unit` type, and the sequence $M;N$ is used to discard it. Qubits can be combined via the tensor product $M \otimes N$ with $\text{let } x^{S_1} \otimes y^{S_2} = M \text{ in } N$ as its corresponding destructor.

Given $D : \mathbb{N}^{k+1} \rightarrow \mathcal{D}$, $N = [n_1, \dots, n_m] \in \mathbb{N}^m$,

$$((D_1 \otimes D_2)(n), n \in N) := (D_1(n), n \in N) \otimes (D_2(n), n \in N) \quad \frac{v(N) \boxed{v(n)} v(N)}{n \in N} := \frac{v(N)}{n \in N}$$

$$((D_2 \circ D_1)(n), n \in N) := (D_2(n), n \in N) \circ (D_1(n), n \in N) \quad \frac{v(N) \boxed{v(n)} \parallel}{n \in N} := \frac{v(N)}{\parallel}$$

$$\frac{v(N) \boxed{v(n) \text{ } v(n)} v(N)}{n \in N} := \frac{v(N) \text{ } v(N)}{n \in N} \quad \frac{v(N) \boxed{v(n) \xrightarrow{\sigma(N)} v(n)} v(N)}{n \in N} := \frac{v(N) \xrightarrow{\sigma(N)} v(N)}{n \in N}$$

$$\frac{v(N) \boxed{v(n) \text{ } v(n)} v(N)}{n \in N} := \frac{v(N) \text{ } v(N)}{n \in N} \quad \frac{v(N) \boxed{v(n) \text{ } v(n)} v(N)}{n \in N} := \frac{v(N) \text{ } v(N)}{n \in N}$$

$$\frac{v(N) \boxed{v(n) \text{ } w(N)} v(N)}{n \in N} := \frac{v(N) \text{ } (v+w)(N)}{n \in N} \quad \frac{v(N) \boxed{v(n) \xrightarrow{\sigma(N,v,w)} w(N)} v(N)}{n \in N} := \frac{v(N) \xrightarrow{\sigma(N,v,w)} (v+w)(N)}{n \in N}$$

Where $\sigma(N, v, w) \in \mathbb{F}_2^{v(N)+w(N) \times v(N)+w(N)}$ is the permutation defined as the matrix

$$\sigma(N, v, w) = (\sigma_f^N \mid \sigma_g^N), \quad \sigma_f^N \in \mathbb{F}_2^{v(N)+w(N) \times v(N)}, \quad \sigma_g^N \in \mathbb{F}_2^{v(N)+w(N) \times w(N)}$$

$$\sigma_f^{\boxed{\cdot}} = Id_0 \quad \sigma_f^{n::N'} = \begin{pmatrix} Id_{v(n)} & 0 \\ 0 & 0 \\ 0 & \sigma_f^{N'} \end{pmatrix} \quad \sigma_g^{\boxed{\cdot}} = Id_0 \quad \sigma_g^{n::N'} = \begin{pmatrix} 0 & 0 \\ Id_{w(n)} & 0 \\ 0 & \sigma_g^{N'} \end{pmatrix}$$

Figure 1: Definition of the list instantiation operator.

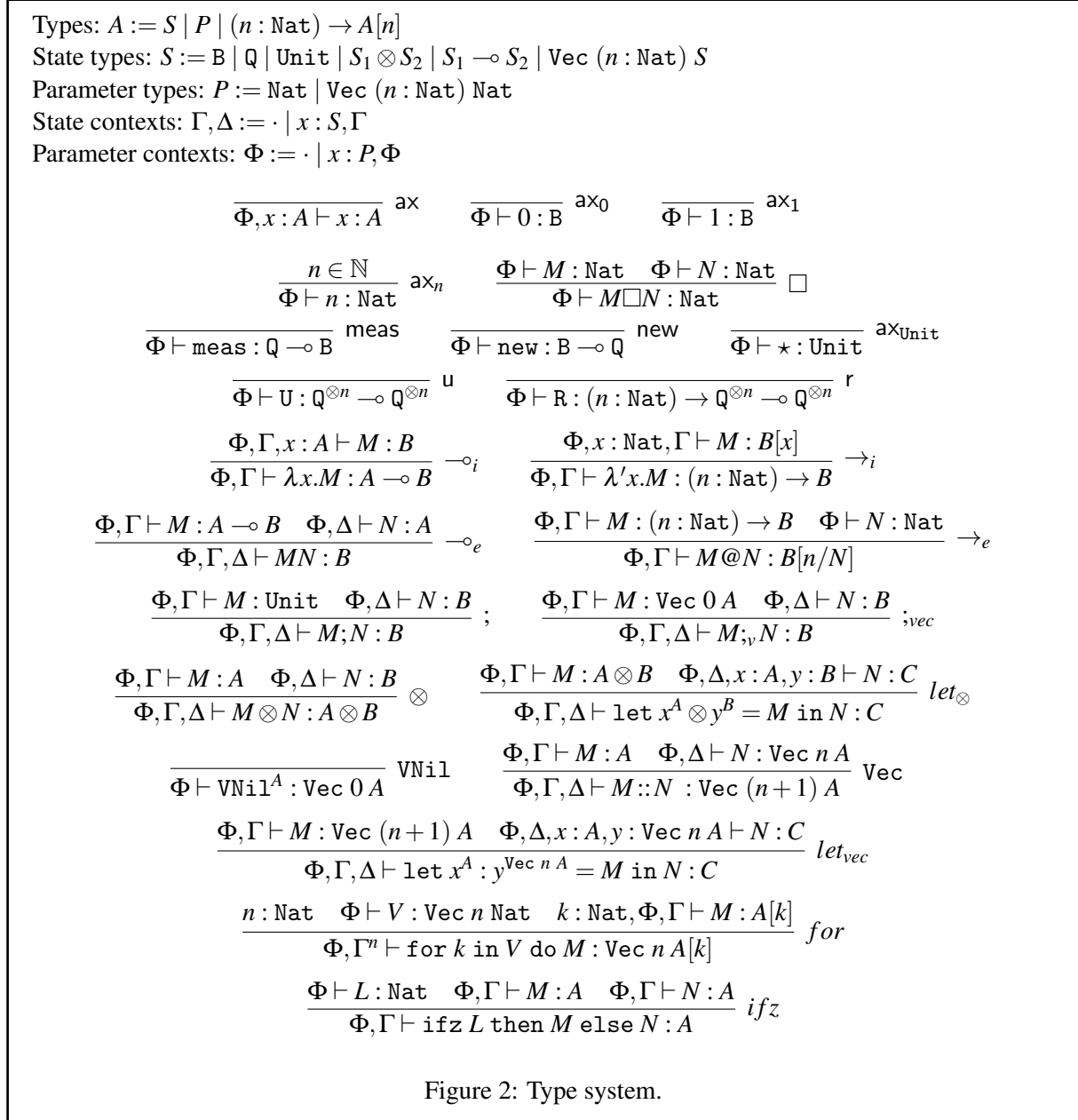
The system supports lists; VNil^A represents the empty list, $M :: N$ the constructor and $\text{let } x^S :: y^{\text{Vec } n S} = M \text{ in } N$ acts as the destructor. Finally, the term $\text{for } k^P \text{ in } M \text{ do } N$ allows iterating over parameter lists.

The typing system is defined in Figure 2. We write $|\Phi|$ for the list of variables in a typing context Φ . The type $\text{Vec } n A$ represents a vector of known length n of elements of type A .

We differentiate between *state contexts* (Noted with Γ and Δ) and *parameter contexts* (Noted with Φ). For our case of study, parameter contexts consist only of pairs $x : \text{Nat}$ or $x : \text{Vec } (n : \text{Nat}) \text{ Nat}$, since they are the only non-linear types of variables that we manage. Every other variable falls under the state context. The terms $\lambda x^S.M$ and MN correspond to the abstraction and application which will be used for state-typed terms. The analogous constructions for parameter-typed terms are $\lambda' x^P.M$ and $M@N$.

In this sense we deviate from the original Proto-Quipper-D type system, which supports a single context decorated with indices. Instead, we use a linear and non-linear approach similar to the work of Cervesato and Pfenning[9].

A key difference between Quipper (and, by extension, Proto-Quipper-D) and λ_D is the approach to defining circuits. In Quipper, circuits are an intrinsic part of the language and can be operated upon. In our case, the translation into SZX diagrams will be mediated with a function defined outside the language.



Types are divided into two kinds; parameter and state types. Both of these can depend on terms of type Nat . For the scope of this work, this dependence may only influence the size of vectors types.

Parameter types represent non-linear variable types which are known at the time of generation of the concrete quantum operations. In the translation into SZX diagrams, these variables may dictate the labels of the wires and spiders. Vectors of Nat terms represent their cartesian product. On the other hand, state

types correspond to the quantum operations and states to be computed. In the translation, these terms inform the shape and composition of the diagrams. Vectors of state type terms represent their tensor product.

In lieu of unbounded and implicit recursion, we define a series of primitive functions for performing explicit vector manipulation. These primitives can be defined in the original language, with the advantage of them being strongly normalizing. The first four primitives are used to manage state vectors, while the last one is used for generating parameters. For ease of translation some terms are decorated with type annotations, however we will omit these for clarity when the type is apparent.

$$\begin{aligned}
\Phi &\vdash \text{accuMap}_{A,B,C} : (n : \text{Nat}) \rightarrow \text{Vec } n A \multimap \text{Vec } n (A \multimap C \multimap B \otimes C) \multimap C \multimap (\text{Vec } n B) \otimes C \\
\Phi &\vdash \text{split}_A : (n : \text{Nat}) \rightarrow (m : \text{Nat}) \rightarrow \text{Vec } (n + m) A \multimap \text{Vec } n A \otimes \text{Vec } m A \\
\Phi &\vdash \text{append}_A : (n : \text{Nat}) \rightarrow (m : \text{Nat}) \rightarrow \text{Vec } n A \multimap \text{Vec } m A \multimap \text{Vec } (n + m) A \\
\Phi &\vdash \text{drop} : (n : \text{Nat}) \rightarrow \text{Vec } n \text{Unit} \multimap \text{Unit} \\
\Phi &\vdash \text{range} : (n : \text{Nat}) \rightarrow (m : \text{Nat}) \rightarrow \text{Vec } (m - n) \text{Nat}
\end{aligned}$$

Since every diagram represents a linear map between qubits there is no representation equivalent to non-terminating terms, even for weakly normalizing programs. This is the main reason behind the design choice of the primitives set. We include the operational semantics of the calculus and primitives in Appendix B. The encoding of the primitives as Proto-Quipper-D functions is shown in Appendix C.

We additionally define the following helpful terms based on the previous primitives to aid in the manipulation of vectors. Cf. Appendix B for their definition as λ_D -terms.

$$\begin{aligned}
\Phi &\vdash \text{map}_{A,B} : (n : \text{Nat}) \rightarrow \text{Vec } n A \multimap \text{Vec } n (A \multimap B) \multimap \text{Vec } n B \\
\Phi &\vdash \text{fold}_{A,C} : (n : \text{Nat}) \rightarrow \text{Vec } n A \multimap \text{Vec } n (A \multimap C \multimap C) \multimap C \multimap C \\
\Phi &\vdash \text{compose}_A : (n : \text{Nat}) \rightarrow \text{Vec } n (A \multimap A) \multimap A \multimap A
\end{aligned}$$

The distinction between primitives that deal with state and parameters highlights the inclusion of the `for` as a construction into the language instead of a primitive. Since it acts over both parameter and state types, its function is effectively to bridge the gap between the two of them. This operation closely corresponds to the list instantiation procedure presented in the Section 2.1.

For example, if we take ns to be a vector of natural numbers, and xs a vector of abstractions $R@k(\text{new}0)$. The term `for k in ns do xs` generates a vector of quantum maps by instantiating the abstractions for each individual parameter in ns .

4 Encoding programs as diagram families

In this section we introduce an encoding of the lambda calculus presented in Section 3 into families of SZX diagrams with context variables as inputs and term values as outputs. We split the lambda-terms into those that represent linear mappings between quantum states and can be encoded as families of SZX diagrams, and parameter terms that can be completely evaluated at compile-time.

4.1 Parameter evaluation. We say a type is *evaluable* if it has the form $A = (n_1 : \text{Nat}) \rightarrow \dots \rightarrow (n_k : \text{Nat}) \rightarrow P[n_1, \dots, n_k]$ with P a parameter type. Since A does not encode a quantum operation, we interpret it directly into functions over vectors of natural numbers. The translation of an evaluable type, $[A]$, is defined recursively as follows:

$$[(n : \text{Nat}) \rightarrow B[n]] = \mathbb{N} \rightarrow \bigcup_{n \in \mathbb{N}} [B[n]] \quad [\text{Nat}] = \mathbb{N} \quad [\text{Vec } (n : \text{Nat}) \text{Nat}] = \mathbb{N}^n$$

Given a type judgement $\Phi \vdash L : P$ where P is an evaluable type, we define $\llbracket L \rrbracket_\Phi$ as the evaluation of the term into a function from parameters into products of natural numbers. Since the typing is syntax directed, the evaluation is defined directly over the terms as follows:

$$\begin{aligned}
\llbracket x \rrbracket_{x:\text{Nat}, \Phi} &= x, |\Phi| \mapsto x & \llbracket n \rrbracket_\Phi &= |\Phi| \mapsto n & \llbracket M \square N \rrbracket_\Phi &= |\Phi| \mapsto \llbracket M \rrbracket_\Phi(|\Phi|) \square \llbracket N \rrbracket_\Phi(|\Phi|) \\
\llbracket M :: N \rrbracket_\Phi &= |\Phi| \mapsto \llbracket M \rrbracket_\Phi(|\Phi|) \times \llbracket N \rrbracket_\Phi(|\Phi|) & \llbracket \text{vNil}^{\text{Nat}} \rrbracket_\Phi &= |\Phi| \mapsto [] \\
\llbracket \lambda' x^P . M \rrbracket_\Phi &= x, |\Phi| \mapsto \llbracket M \rrbracket_\Phi(x, |\Phi|) & \llbracket M @ N \rrbracket_\Phi &= \llbracket M \rrbracket_\Phi(\llbracket N \rrbracket_\Phi(|\Phi|), \Phi) \\
\llbracket \text{ifz } L \text{ then } M \text{ else } N \rrbracket_\Phi &= |\Phi| \mapsto \begin{cases} \llbracket M \rrbracket_\Phi(|\Phi|) & \text{if } \llbracket L \rrbracket_\Phi(|\Phi|) = 0 \\ \llbracket N \rrbracket_\Phi(|\Phi|) & \text{otherwise} \end{cases} & \llbracket \text{range} \rrbracket_\Phi &= n, m, |\Phi| \mapsto \prod_{i=n}^{m-1} i \\
\llbracket \text{for } k \text{ in } V \text{ do } M \rrbracket_\Phi &= |\Phi| \mapsto \prod_{k \in \llbracket V \rrbracket_\Phi(|\Phi|)} \llbracket M \rrbracket_{k:\text{Nat}\Phi}(k, |\Phi|) \\
\llbracket \text{let } x^P :: y^{\text{vec } n P} = M \text{ in } N \rrbracket_\Phi &= |\Phi| \mapsto \llbracket N \rrbracket_{x:P, y:\text{vec } n P, \Phi}(y_1, y_2, \dots, y_n, |\Phi|) \text{ where } [y_1, \dots, y_n] = \llbracket M \rrbracket_\Phi(|\Phi|)
\end{aligned}$$

Lemma 4.1 Given an evaluable type A and a type judgement $\Phi \vdash L : A$, $\llbracket L \rrbracket_\Phi \in \times_{x:P \in \Phi} [P] \rightarrow [A]$.

Lemma 4.2 Given an evaluable type A , a type judgement $\Phi \vdash L : A$, and $M \rightarrow N$, then $\llbracket M \rrbracket_\Phi = \llbracket N \rrbracket_\Phi$.

4.2 Diagram encoding. A non-evaluable type has necessarily the form $A = (n_1 : \text{Nat}) \rightarrow \dots \rightarrow (n_k : \text{Nat}) \rightarrow S$, with S any state type. We call such types *translatable* since they correspond to terms that encode quantum operations that can be described as families of diagrams.

We first define a translation $\llbracket \cdot \rrbracket$ from state types into wire multiplicities as follows. Notice that due to the symmetries of the SZX diagrams the linear functions have the same representation as the products.

$$\llbracket B \rrbracket = 1 \quad \llbracket Q \rrbracket = 1 \quad \llbracket \text{Vec } (n : \text{Nat}) A \rrbracket = [A]^{\otimes n} \quad \llbracket A \otimes B \rrbracket = [A] \otimes [B] \quad \llbracket A \multimap B \rrbracket = [A] \otimes [B]$$

Given a translatable type judgement $\Phi, \Gamma \vdash M : (n_1 : \text{Nat}) \rightarrow \dots \rightarrow (n_k : \text{Nat}) \rightarrow S$ we can encode it as a family of SZX diagrams $n_1, \dots, n_k, |\Phi| \mapsto \frac{\llbracket \Gamma \rrbracket}{\llbracket M(|\Phi|) \rrbracket} \llbracket S[|\Phi|] \rrbracket$. We will omit the brackets in our diagrams for clarity. In a similar manner to the evaluation, we define the translation $\llbracket M \rrbracket_{\Phi, \Gamma}$ recursively on the terms as follows:

$$\begin{aligned}
\llbracket x \rrbracket_{\Phi, x:A} &= |\Phi| \mapsto \text{---}^A & \llbracket 0 \rrbracket_\Phi &= |\Phi| \mapsto \text{---}^Q & \llbracket 1 \rrbracket_\Phi &= |\Phi| \mapsto \text{---}^Q & \llbracket \text{meas} \rrbracket_\Phi &= |\Phi| \mapsto \text{---}^Q \text{---}^Q \\
\llbracket \text{new} \rrbracket_\Phi &= |\Phi| \mapsto \text{---}^{1 \multimap 1} & \llbracket U \rrbracket_\Phi &= |\Phi| \mapsto \text{---}^{nQ \multimap nQ} & \llbracket R \rrbracket_\Phi &= n, |\Phi| \mapsto \text{---}^{Q \multimap Q} \\
\llbracket \lambda' x^A . M \rrbracket_{\Phi, \Gamma} &= x, |\Phi| \mapsto \frac{\Gamma}{M(x, |\Phi|)} \text{---}^A & \llbracket M @ N \rrbracket_{\Phi, \Gamma} &= |\Phi| \mapsto \frac{\Gamma}{M(\llbracket N \rrbracket_\Phi(|\Phi|), |\Phi|)} \text{---}^B \\
\llbracket \lambda x^A . M \rrbracket_{\Phi, \Gamma} &= |\Phi| \mapsto \frac{\Gamma}{M(|\Phi|)} \text{---}^A \text{---}^B & \llbracket M N \rrbracket_{\Phi, \Gamma, \Delta} &= |\Phi| \mapsto \frac{\Delta}{N(|\Phi|)} \text{---}^A \text{---}^B \\
\llbracket M ; N \rrbracket_{\Phi, \Gamma, \Delta} &= |\Phi| \mapsto \frac{\Gamma}{M(|\Phi|)} \text{---}^A \text{---}^B & \llbracket * \rrbracket_\Phi &= |\Phi| \mapsto \text{---} & \llbracket M \otimes N \rrbracket_{\Phi, \Gamma, \Delta} &= |\Phi| \mapsto \frac{\Gamma}{M(|\Phi|)} \text{---}^A \text{---}^B
\end{aligned}$$

$$\begin{aligned}
 \llbracket M;_v N \rrbracket_{\Phi, \Gamma, \Delta} = |\Phi\rangle &\mapsto \frac{\Gamma}{\Delta} \frac{\boxed{M(|\Phi\rangle)}}{\boxed{N(|\Phi\rangle)}} \frac{A}{A} & \llbracket \text{VNil} \rrbracket_{\Phi} = |\Phi\rangle &\mapsto \boxed{} \\
 \llbracket \text{let } x^A \otimes y^B = M \text{ in } N \rrbracket_{\Phi, \Gamma, \Delta} = |\Phi\rangle &\mapsto \frac{\Gamma}{\Delta} \frac{\boxed{M(|\Phi\rangle)}^{A \otimes B}}{\boxed{N(|\Phi\rangle)}^C} \frac{A}{B} \\
 \llbracket \text{let } x^A : y^{\text{vec } n A} = M \text{ in } N \rrbracket_{\Phi, \Gamma, \Delta} = |\Phi\rangle &\mapsto \frac{\Gamma}{\Delta} \frac{\boxed{M(|\Phi\rangle)}^{A^{\otimes n+1}}}{\boxed{N(|\Phi\rangle)}^C} \frac{A}{A^{\otimes n}} \\
 \llbracket M :: N \rrbracket_{\Phi, \Gamma, \Delta} = |\Phi\rangle &\mapsto \frac{\Gamma}{\Delta} \frac{\boxed{M(|\Phi\rangle)}^A}{\boxed{N(|\Phi\rangle)}^{A^{\otimes n}}} \frac{A^{\otimes n+1}}{A^{\otimes n}} & \llbracket \text{for } k \text{ in } V \text{ do } M \rrbracket_{\Phi, \Gamma^n} = |\Phi\rangle &\mapsto \frac{\Gamma^{\otimes n}}{A^{\otimes n}} \frac{\boxed{M(k)}_{k \in |V|}(|\Phi\rangle)}{A^{\otimes n}} \\
 \llbracket \text{ifz } L \text{ then } M \text{ else } N \rrbracket_{\Phi, \Gamma} = |\Phi\rangle &\mapsto \frac{\Gamma}{\Gamma^{\otimes \delta_l > 0}} \frac{\boxed{M}_{k \in [0]^{\otimes \delta_l = 0}}}{\boxed{N}_{k \in [0]^{\otimes \delta_l > 0}}} \frac{A^{\otimes \delta_l = 0}}{A^{\otimes \delta_l > 0}}
 \end{aligned}$$

where δ is the Kronecker delta and $l = \lfloor L \rfloor(|\Phi\rangle)$. Notice that the new and meas operations share the same translation. Although new can be encoded as a simple wire, we keep the additional node to maintain the symmetry with the measurement.

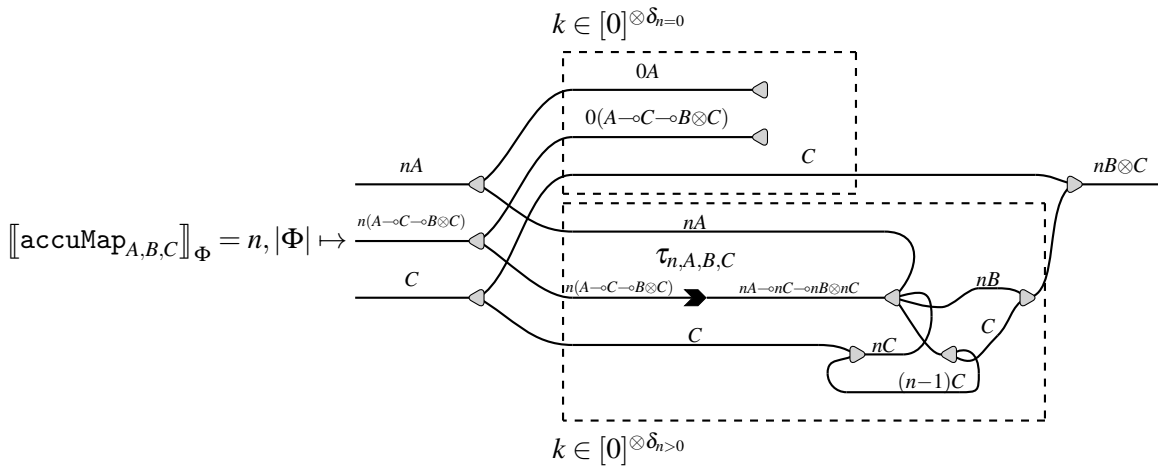
The unitary operators U and rotations R correspond to a predefined set of primitives, and their translation is defined on a by case basis. The following table shows the encoding of the operators used in this paper.

Name	Rz(n)	Rz ⁻¹ (n)	Rx(n)	Rx ⁻¹ (n)	H	CNOT
Encoding						

The primitives `split`, `append`, `drop` and `accuMap` are translated below. Since vectors are isomorphic to products in the wire encoding, the first three primitives do not perform any operation. For the accumulating map we utilize the construction presented in Lemma 2.1, replacing the function box with a function vector input. In the latter we omit the wires and gathers connecting the inputs and outputs of the function to a single wire on the right of the diagram for clarity.

$$\llbracket \text{split}_A \rrbracket_{\Phi} = n, m, |\Phi\rangle \mapsto \frac{(n+m)A}{(n+m)A \rightarrow nA \otimes mA} \quad \llbracket \text{append}_A \rrbracket_{\Phi} = n, m, |\Phi\rangle \mapsto \frac{(n+m)A}{nA \rightarrow mA \rightarrow (n+m)A}$$

$$\llbracket \text{drop} \rrbracket_{\Phi} = n, |\Phi\rangle \mapsto \frac{n0 \rightarrow 0}{k \in [0]^{\otimes \delta_n = 0}}$$



where $\tau_{n,A,B,C}$ is a permutation that rearranges the vectors of functions into tensors of vectors for each parameter and return value. That is, $\tau_{n,A,B,C}$ reorders a sequence of registers $(A,C,B,C) \dots (A,C,B,C)$ into the sequence $(A \dots A)(C \dots C)(B \dots B)(C \dots C)$. It is defined as follows,

$$\tau_{n,A,B,C}(i) = \begin{cases} i \bmod k + a * (i \operatorname{div} k) & \text{if } i \bmod k < a \\ i \bmod k + c * (i \operatorname{div} k) + a * (n - 1) & \text{if } a \leq i \bmod k < (a + c) \\ i \bmod k + b * (i \operatorname{div} k) + (a + c) * (n - 1) & \text{if } (a + c) \leq i \bmod k < (a + c + b) \\ i \bmod k + c * (i \operatorname{div} k) + (a + c + b) * (n - 1) & \text{if } (a + c + b) \leq i \bmod k \end{cases}$$

for $i \in [0, (a + c + b + c) * n)$, where \bmod and div are the integer modulo and division operators, $a = \llbracket A \rrbracket$, $b = \llbracket B \rrbracket$, $c = \llbracket C \rrbracket$, and $k = a + c + b + c$.

As a consequence of Lemma 2.4, the number of nodes in the produced diagrams grows linearly with the size of the input. Notice that the ZX spiders, the ground, and the Hadamard operator are only produced in the translations of the quantum primitives. We may instead have used other variations of the calculus supporting the scalable extension, such as the ZH calculus [2], better suited for other sets of quantum operators.

Lemma 4.3 The translation procedure is correct in respect to the operational semantics of λ_D . If A is a translatable type, $\Phi, \Gamma \vdash M : A$, and $M \rightarrow N$, then $\llbracket M \rrbracket_{\Phi, \Gamma} = \llbracket N \rrbracket_{\Phi, \Gamma}$.

5 Application example: QFT

The Quantum Fourier Transform is an algorithm used extensively in quantum computation, notably as part of Shor's algorithm for integer factorization [16]. The QFT function operates generically over n -qubit states and in general a circuit encoding of it requires $\mathcal{O}(n^2)$ gates. In this section we present an encoding of the algorithm as a λ_D term, followed by the translation into a family of constant-sized diagrams. The corresponding Proto-Quipper-D program is listed in Appendix D.

The following presentation divides the algorithm into three parts. The *crot* term applies a controlled rotation over a qubit with a parametrized angle. *apply_crot* operates over the last $n - k$ qubits of an n -qubit state by applying a Hadamard gate to the first one and then using it as target of successive *crot* applications using the rest of the qubits as controls. Finally, *qft* repeats *apply_crot* for all values of k . In the terms, we use $n \dots m$ as a shorthand for $\text{range } @n @m$.

```
crot : (n : Nat) → (Q ⊗ Q) → (Q ⊗ Q)
crot := λ' nNat. λ qsQ ⊗ Q. let cQ ⊗ qQ = qs in let cQ ⊗ qQ = CNOT c (Rz @2n q) in CNOT c (Rz-1 @2n q)
```

```
apply_crot : (n : Nat) → (k : Nat) → Vec n Q → Vec n Q
```

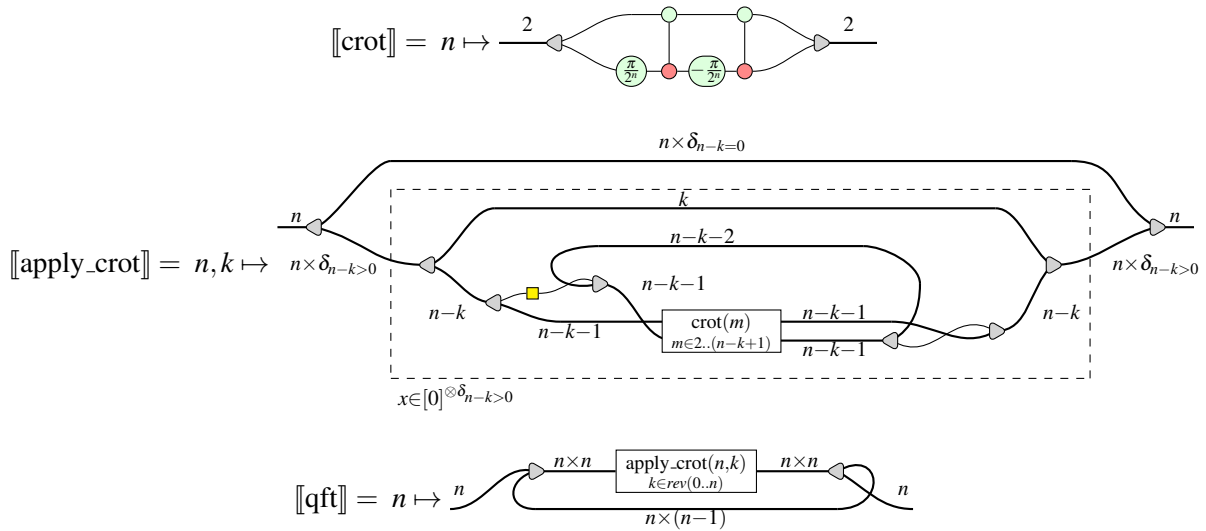
```
apply_crot := λ' nNat. λ' kNat. λ qsVec n Q.
  ifz (n - k) then qs else
  let hVec k Q ⊗ qs'Vec n - k Q = split @k @(n - k) qs in
  let qQ ⊗ csVec n - k - 1 Q = qs' in
  let fsVec (n - k - 1) (Q ⊗ Q → Q ⊗ Q) = for mNat in 2..(n - k + 1) do crot @m in
  let cs'(Vec n - k - 1 Q) ⊗ q'Q = accuMap fs (H q) cs in
  concat h (q' : cs')
```

$$\text{qft} : (n : \text{Nat}) \rightarrow \text{Vec } n \text{ Q} \multimap \text{Vec } n \text{ Q}$$

$$\text{qft} := \lambda' n^{\text{Nat}} . \lambda q_s^{\text{Vec } n \text{ Q}} . \text{compose}$$

$$(\text{for } k^{\text{Nat}} \text{ in reverse_vec } @ (0..n) \text{ do } \lambda q_s' ^{\text{Vec } n \text{ Q}} . \text{apply_crot } @ n @ k q_s') q_s$$

The translation of each term into a family of diagrams is shown below. We omit the wire connecting the function inputs to the right side of the graphs for clarity and eliminate superfluous gathers and splitters using rules **(sg)** and **(gs)**. Notice that, in contrast to a quantum circuit encoding, the resulting diagram's size does not depend on the number of qubits n .



6 Discussion

In this article, we presented an efficient method to compile parametric quantum programs written in a fragment of the Proto-Quipper-D language into families of SZX diagrams. We restricted the fragment to strongly normalizing terms that can be represented as diagrams. Additionally, we introduced a notation to easily compose elements of a diagram family in parallel. We proved that our method produces compact diagrams and shown that it can encode non-trivial algorithms.

A current line of work is defining categorical semantics for the calculus and families of diagrams, including a subsequent proof of adequacy for the translation. More work needs to be done to expand the fragment of the Quipper language that can be translated.

We would like to acknowledge Benoît Valiron for helpful discussion on this topic, and Frank Fu for his help during the implementation of the Proto-Quipper-D primitives. This work was supported in part by the French National Research Agency (ANR) under the research projects SoftQPRO ANR-17-CE25-0009-02 and Plan France 2030 ANR-22-PETQ-0007, by the DGE of the French Ministry of Industry under the research project PIA-GDN/QuantEx P163746-484124, by the project STIC-AmSud project Qapla' 21-SITC-10, the ECOS-Sud A17C03 project, the PICT-2019-1272 project, the French-Argentinian IRP SINFIN and the PIP 11220200100368CO project.

References

- [1] MD SAJID ANIS et al. (2021): *Qiskit: An Open-source Framework for Quantum Computing*, doi:10.5281/zenodo.2562111.
- [2] 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.
- [3] 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.
- [4] Niel de Beaudrap & Dominic Horsman (2020): *The ZX calculus is a language for surface code lattice surgery*. *Quantum* 4, p. 218, doi:10.22331/q-2020-01-09-218. arXiv: 1704.08670.
- [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] Titouan Carette, Yohann D’Anello & Simon Perdrix (2021): *Quantum Algorithms and Oracles with the Scalable ZX-calculus*. *Electronic Proceedings in Theoretical Computer Science* 343, pp. 193–209, doi:10.4204/EPTCS.343.10.
- [7] Titouan Carette, Dominic Horsman & Simon Perdrix (2019): *SZX-calculus: Scalable Graphical Quantum Reasoning*. arXiv:1905.00041 [quant-ph], p. 15 pages, doi:10.4230/LIPIcs.MFCS.2019.55. ArXiv: 1905.00041.
- [8] Titouan Carette, Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2019): *Completeness of Graphical Languages for Mixed States Quantum Mechanics*, doi:10.1145/3464693. ArXiv: 1902.07143.
- [9] Iliano Cervesato & Frank Pfenning (1996): *A linear logical framework*. *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pp. 264–275, doi:10.1109/lics.1996.561339.
- [10] Bob Coecke & Simon Perdrix (2012): *Environment and classical channels in categorical quantum mechanics*. *Logical Methods in Computer Science* Volume 8, Issue 4, doi:10.2168/LMCS-8(4:14)2012.
- [11] Cirq Developers (2021): *Cirq*, doi:10.5281/zenodo.5182845. See full list of authors on Github: <https://github.com/quantumlib/Cirq/graphs/contributors>.
- [12] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2019): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*, doi:10.22331/q-2020-06-04-279. arXiv:1902.03178.
- [13] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2020): *A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper*. arXiv:2005.08396 [quant-ph], doi:10.1007/978-3-030-52482-1_9. ArXiv: 2005.08396.
- [14] Peng Fu, Kohei Kishida & Peter Selinger (2021): *Linear Dependent Type Theory for Quantum Programming Languages*. arXiv:2004.13472 [quant-ph], doi:10.46298/lmcs-18(3:28)2022. ArXiv: 2004.13472.

- [15] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: a scalable quantum programming language*. *ACM SIGPLAN Notices* 48(6), p. 333, doi:10.1145/2499370.2462177.
- [16] Michael A. Nielsen & Isaac L. Chuang (2011): *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, doi:10.1017/cbo9780511976667.
- [17] Damian S. Steiger, Thomas Häner & Matthias Troyer (2018): *ProjectQ: an open source software framework for quantum computing*. *Quantum* 2, p. 49, doi:10.22331/q-2018-01-31-49.
- [18] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*, doi:10.48550/ARXIV.2012.13966.

A Semantics of the SZX calculus

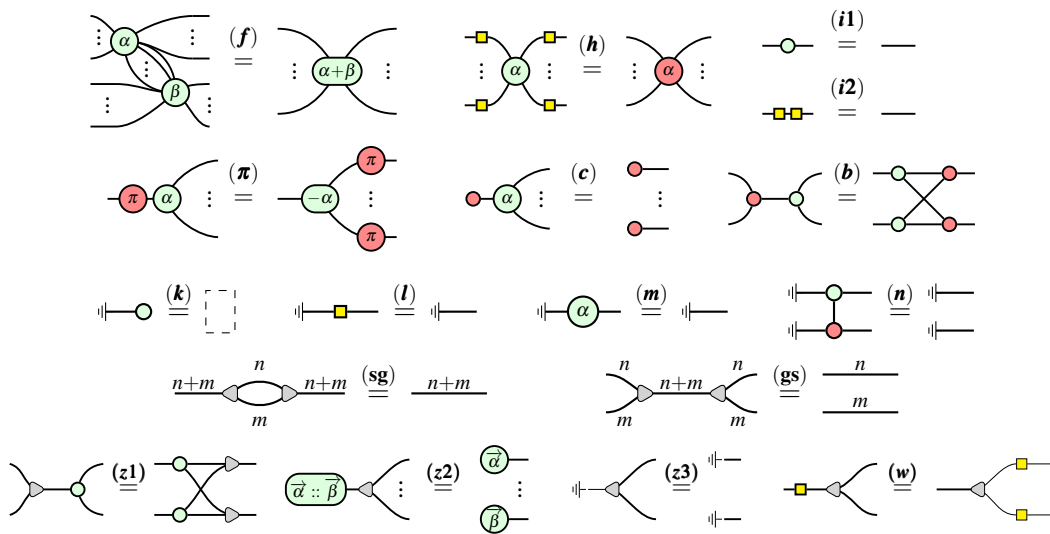
We reproduce below the standard interpretation of SZX_{\pm} diagrams as density matrices and completely positive maps [8, 6], modulo scalars.

Let $D_n \subseteq \mathbb{C}^{2^n \times 2^n}$ be the set of n -qubit density matrices. We define the functor $\{\cdot\} : ZX_{\pm} \rightarrow \text{CPM}(\text{Qubit})$ which associates to any diagram $\mathcal{D} : n \rightarrow m$ a completely positive map $\{\mathcal{D}\} : D_n \rightarrow D_m$, inductively as follows.

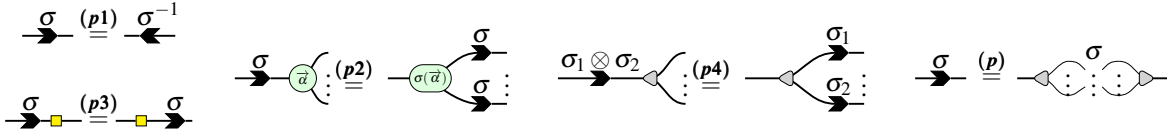
$$\begin{aligned} \{\mathcal{D}_1 \otimes \mathcal{D}_2\} &:= \{\mathcal{D}_1\} \otimes \{\mathcal{D}_2\} & \{\mathcal{D}_2 \circ \mathcal{D}_1\} &:= \{\mathcal{D}_2\} \circ \{\mathcal{D}_1\} \\ \{\text{---} \square \text{---}\} &:= \rho \mapsto V \rho V^\dagger \text{ where } V = \sum_{x,y \in \mathbb{F}_2^k} (-1)^{x \bullet y} |y\rangle\langle x| \\ \left\{ \left\{ \begin{array}{c} k & k \\ \alpha \\ k & k \end{array} \right\} \right\} &:= \rho \mapsto V \rho V^\dagger \text{ where } V = \sum_{x \in \mathbb{F}_2^k} e^{ix \bullet \vec{\alpha}} |x\rangle^{\otimes m} \langle x|^{\otimes n} \\ \left\{ \left\{ \begin{array}{c} k & k \\ \bar{\alpha} \\ k & k \end{array} \right\} \right\} &:= \{\text{---} \square \text{---}\}^{\otimes m} \circ \left\{ \left\{ \begin{array}{c} k & k \\ \alpha \\ k & k \end{array} \right\} \right\} \circ \{\text{---} \square \text{---}\}^{\otimes n} \\ \{\text{---} \dashv \text{---}\} &:= \rho \mapsto \sum_{x \in \mathbb{F}_2^k} \langle x | \rho | x \rangle & \{\text{---} \dashv \text{---}\} &:= \sum_{x \in \mathbb{F}_2^k} |x\rangle\langle x| & \{\text{---} \text{---}\} &:= \rho \mapsto \rho \\ \left\{ \left\{ \begin{array}{c} n \\ m \end{array} \right\} \right\} &:= \rho \mapsto \rho & \{\text{---} \vec{\sigma} \text{---}\} &:= \rho \mapsto V \rho V^\dagger \text{ where } V = \sum_{x \in \mathbb{F}_2^k} |\sigma(x)\rangle\langle x| \\ \left\{ \left\{ \begin{array}{c} k \\ k \end{array} \right\} \right\} &:= \rho \mapsto \sum_{x \in \mathbb{F}_2^k} \langle xx | \rho | xx \rangle & \left\{ \left\{ \begin{array}{c} k \\ k \end{array} \right\} \right\} &:= \sum_{x \in \mathbb{F}_2^k} |xx\rangle\langle xx| & \left\{ \left\{ \begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \right\} \right\} &:= Id_0 \\ \left\{ \left\{ \begin{array}{c} k & l \\ \times \\ k & l \end{array} \right\} \right\} &:= \rho \mapsto V \rho V^\dagger \text{ where } V = \sum_{x \in \mathbb{F}_2^k, y \in \mathbb{F}_2^l} |yx\rangle\langle xy| \end{aligned}$$

where $\forall u, v \in \mathbb{R}^n, u \bullet v = \sum_{i=1}^n u_i v_i$.

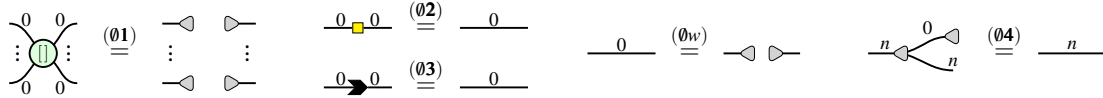
The SZX_{\pm} calculus defines a set of rewrite rules, shown below.



Additionally, for the arrows restricted to permutations of wires we have the following rules [6]:



Finally, since wires with cardinality zero correspond to empty mappings they can be discarded from the diagrams.



B Operational Semantics of the λ_D calculus

We define a weak call-by-value small step operational semantics on Table 1.

A key point to note here is that every rewriting rule preserves the state. There are no measurements or unitary operations applied, the rewriting is merely syntactical. Since our goal is translation into an SZX-diagram, this system is powerful enough. We include the rewrite rules for the primitives on Table 2.

Additionally, we define useful macros based on these functions on Table 3. They provide syntactic sugar to deal with state vectors.

C Implementation of primitives in Proto-Quipper-D

The implicitly recursive primitives defined in Section 3 can be implemented in proto-quipper-D as follows. The implementation has been checked with the dpq tool implemented by Frank Fu (see <https://gitlab.com/frank-peng-fu/dpq-remake>).

```

module Primitives where
import "/dpq/Prelude.dpq"

foreach : ! forall a b (n : Nat)
  -> (Parameter a) => !(a -> b) -> Vec a n -> Vec b n
foreach f l = map f l

split : ! forall a (n : Nat) (m : Nat) -> Vec a (n+m) -> Vec a n * Vec a m
split n m v =
  case v of
  VNil -> (VNil, VNil)
  VCons x v' ->
    case n of
    Z -> (VNil, v)
    S n' ->
      let (v1, v2) = split n' m v'
      in (VCons x v1, v2)

```

$ \begin{aligned} V &:= x \mid C \mid 0 \mid 1 \mid \text{meas} \mid \text{new} \mid U \mid \\ &\lambda x^S.M \mid \lambda' x^P.M \mid \star \mid M \otimes N \mid \\ &\text{VNil} \mid M :: N \end{aligned} $
$ \begin{aligned} &(\lambda x.M)V \rightarrow M[V/x] \\ &(\lambda' x.M)@V \rightarrow M[V/x] \\ \text{let } x \otimes y = M_1 \otimes M_2 \text{ in } N &\rightarrow N[x/M_1][y/M_2] \\ \text{let } x :: y = M_1 :: M_2 \text{ in } N &\rightarrow N[x/M_1][y/M_2] \\ \text{ifz } V \text{ then } M \text{ else } N &\rightarrow \begin{cases} M & \text{If } V = 0 \\ N & \text{Otherwise} \end{cases} \\ \star ; M &\rightarrow M \\ \text{VNil} ;_v M &\rightarrow M \\ V_1 \square V_2 &\rightarrow V \quad \text{Where } V_i = n_i \text{ and } V = n_1 \square n_2 \\ \text{for } k \text{ in } M_1 :: M_2 \text{ do } N &\rightarrow N[k/M_1] :: \text{for } k \text{ in } M_2 \text{ do } N \\ \text{for } k \text{ in VNil do } N &\rightarrow \text{VNil} \end{aligned} $
$ \frac{M \rightarrow N}{MV \rightarrow NV} \quad \frac{M \rightarrow N}{LM \rightarrow LN} \quad \frac{M \rightarrow N}{M@V \rightarrow N@V} \quad \frac{M \rightarrow N}{L@M \rightarrow L@N} $
$ \frac{M \rightarrow N}{\text{let } x \otimes y = M \text{ in } L \rightarrow \text{let } x \otimes y = N \text{ in } L} \quad \frac{M \rightarrow N}{\text{let } x :: y = M \text{ in } L \rightarrow \text{let } x :: y = N \text{ in } L} $
$ \frac{M \rightarrow N}{L \square M \rightarrow L \square N} \quad \frac{M \rightarrow N}{M \square V \rightarrow M \square V} $
$ \frac{M \rightarrow N}{M ; L \rightarrow N ; L} \quad \frac{M \rightarrow N}{\text{let } x : y = M \text{ in } L \rightarrow \text{let } x : y = N \text{ in } L} $
$ \frac{M \rightarrow N}{\text{ifz } M \text{ then } L_1 \text{ else } L_2 \rightarrow \text{ifz } N \text{ then } L_1 \text{ else } L_2} \quad \frac{M \rightarrow N}{\text{for } k \text{ in } M \text{ do } L \rightarrow \text{for } k \text{ in } N \text{ do } L} $
<p>Table 1: Rewrite system for λ_D.</p>

```

accuMap @n xs fs z → ifz n then xs ;v fs ;v VNil ⊗ z else
    let x :: xs' = xs in let f :: fs' = fs in
    let y ⊗ z' = f x z in let ys ⊗ z'' = accuMap @(n-1) xs' fs' z' in
    (y :: ys) ⊗ z''
split @n @m xs → ifz n then VNil ⊗ xs else let y :: xs' = xs in
    let ys1 ⊗ ys2 = split@(n-1) @m xs' in (y :: ys1) ⊗ ys2
append @n @m xs ys → ifz n then xs ;v ys else
    let x :: xs' = xs in x :: (append @(n-1) @m xs' ys)
drop @n xs → ifz n then xs ;v ★ else let x :: xs' = xs in x ; drop @(n-1) xs'
range @n @m → ifz m-n then VNil else n :: range @(n+1) @m

```

Table 2: Reductions pertaining to the primitives.

```

map @n xs fs := let fs' ⊗ u1 = accuMap @n fs
    (for k in (0..n) do λf.λu.(λx.λu.fx ⊗ u) ⊗ u) ★
    in let xs' ⊗ u2 = accuMap @n xs fs' ★
    in u1 ; u2 ; xs'
fold @n xs fs z := let fs' ⊗ u = accuMap @n fs
    (for k in (0..n) do λf.λu.(λx.λy.★ ⊗ f x y) ⊗ u) ★
    in let us ⊗ r = accuMap @n xs fs' z
    in u ; drop @n us ; r
compose @n xs = fold @n xs (for k in 0..n do (λf.λg.λx.f (g x))) (λx.x)

```

Table 3: Function macros.

```

cons : ! forall a (n : Nat) (m: Nat) -> Vec a n -> Vec a m -> Vec a (n+m)
cons n m vn vm =
  case vn of
    VNil -> VNil
    VCons x vn' -> x : cons n m vn' vm

accuMap : ! forall a b c (n : Nat)
  -> Vec a n -> Vec (a -> c -> (b,c)) n -> c -> (Vec b n, c)
accuMap n v fs z =
  case v of
    VNil -> (VNil, z)
    VCons x v' ->
      case n of
        S n' ->
          let (y, z') = f x z
          in (VCons y accuMap n' v' f z', z')

mapp : ! forall a b (n : Nat) -> Vec a n -> Vec (a -> b) n -> Vec b n
mapp n v f =
  let (v', _) = accuMap n v (\x z -> (f x, z)) VNil
  in v'

fold : ! forall a b (n : Nat) -> Vec a n -> Vec (a -> b -> b) n -> b -> b
fold n v f z =
  let (_, z') = accuMap n v (\x z -> (VNil, f x z)) z
  in z'

compose : ! (n : Nat) -> Vec (a -> a) n -> a -> a
compose n fs x = fold fs (replicate n (\f x -> f x)) x

range_aux : ! (n : Nat) -> (m : Nat) -> Nat -> Vec Nat (minus m n)
range_aux n m x =
  case m of
    Z -> VNil
    S m' -> case n of
      Z -> let r' = range_aux Z m' (S x)
          in subst (\x -> Vec Nat x) (minusSZ' m') (VCons x r')
      S n' -> range_aux n' m' (S x)

range : ! (n : Nat) -> (m : Nat) -> Vec Nat (minus m n)
range n m = range_aux n m Z

drop : ! (n : Nat) -> Vec Unit n -> Unit
drop n v = case n of
  Z -> ()

```

```

S n' -> case v of
      VCons _ v' -> drop n' v'

```

D QFT algorithm in Quipper code

The following Proto-Quipper-D code corresponds to the algorithm presented in Section 5. This implementation has been checked with the dpq tool implemented by Frank Fu (see <https://gitlab.com/frank-peng-fu/dpq-remake>). Notice that, in contrast to the presented lambda terms, the type checker implementation requires explicit encodings of the Leibniz equalities between parameter types.

```

module Qft where
import "/dpq/Prelude.dpq"

crot : ! (n : Nat) -> Qubit * Qubit -> Qubit * Qubit
crot n q = let (q',c) = q in flip $ R n q' c

-- Specify types to help the typechecker
applyCrot_aux : ! (n : Nat) -> Qubit -> Qubit -> Qubit * Qubit
applyCrot_aux n ctrl q = crot n (q, ctrl)

-- Apply a CROT sequence to a qubit register, ignoring the first k qubits.
applyCrot : ! (n k : Nat) -> Vec Qubit n -> Vec Qubit n
applyCrot n k qs =
  let WithEq r e = inspect (minus n k)
  in case r of
    Z -> qs
    S n' ->
      let
        -- e : Eq Nat (S n') (minus n k)
        -- e' : Eq Nat (add k (S n')) n
        e' = trans (symAdd k (S n')) $ minusPlus n n' k $ sym (minus n k) e
        -- qs' : Vec Qubit (minus n k)
        qs' = subst (\m -> Vec Qubit m) (sym (add k (S n')) e') qs
        (head, qs') = split k (S n') $ qs'
        (q,ctrls) = chop qs'
        -- fs : Vec (Qubit -> Qubit -> Qubit * Qubit) (minus n' Z)
        fs = foreach (\k -> applyCrot_aux (S(S k))) $ 0..n'
        -- fs : Vec (Qubit -> Qubit -> Qubit * Qubit) Z
        eq = sym n' $ minusZ n'
        fs = subst (\m -> Vec (Qubit -> Qubit -> Qubit * Qubit) m) eq fs
        (ctrls', q') = accumap fs (H q) ctrls
      in subst (\m -> Vec Qubit m) e' $ append head (VCons q' ctrls')

-- Required for the type checker to derive the second !
qft_aux : ! (n : Nat) -> ! (k : Nat) -> Vec Qubit n -> Vec Qubit n
qft_aux n head_size qs = applyCrot n head_size qs

```

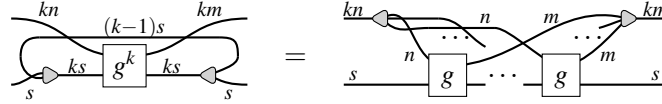
```

qft : ! (n : Nat) -> Vec Qubit n -> Vec Qubit n
qft n qs = let f = qft_aux n in compose' (foreach f $ reverse_vec (0..n)) qs

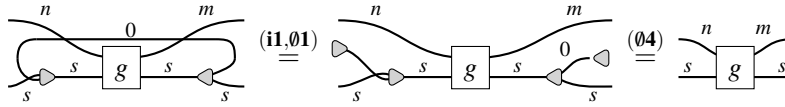
```

E Proofs

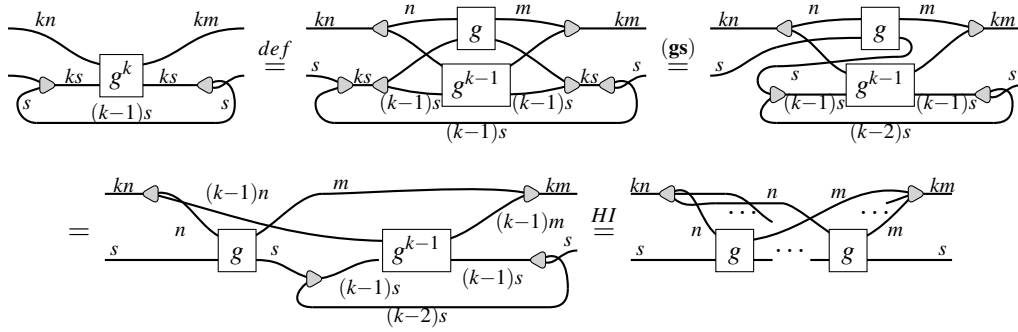
Lemma (2.1) Let $g : 1_n \otimes 1_s \rightarrow 1_m \otimes 1_s$ and $k \geq 1$, then



Proof By induction on k . If $k = 1$,

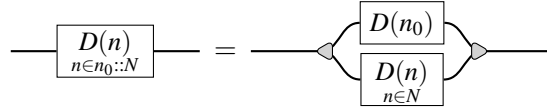


If $k > 1$,



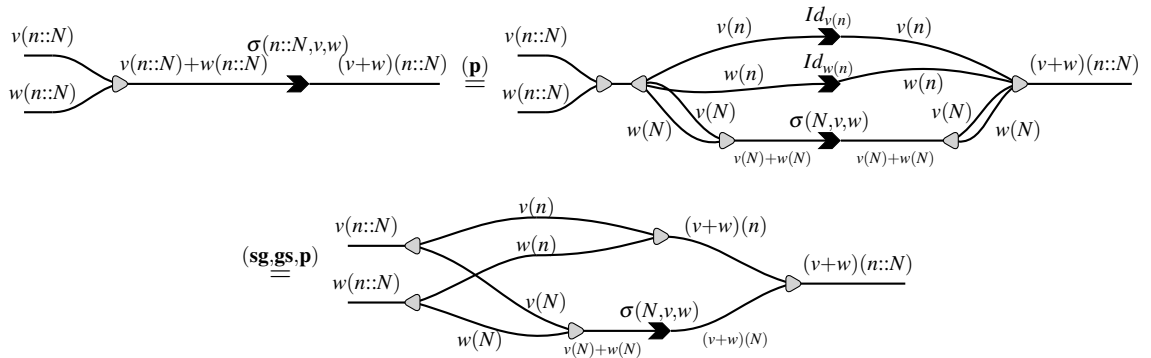
□

Lemma (2.2) For any diagram family D , $n_0 : \mathbb{N}$, $N : \mathbb{N}^k$,



Proof By induction on the term construction

- If \mathcal{D} is a gather,



- The other cases can be directly derived from the commutation properties of the gather generator via rules (z1), (z2), (z3), (w), and (p4). \square

Lemma (2.3) A diagram family initialized with the empty list corresponds to the empty map. For any diagram family D ,

$$\overset{0}{\text{---}} \boxed{\begin{array}{c} D(n) \\ n \in [] \end{array}} \overset{0}{\text{---}} = \overset{0}{\text{---}} \blacktriangleleft \quad \blacktriangleright \overset{0}{\text{---}}$$

Proof Notice that any wire in the initialized diagrams has cardinality zero. By rules (01), (02), (03), (04), and (0w) every internal node can be eliminated from the diagram. \square

Lemma (2.4) The list instantiation procedure on an n -node diagram family adds $\mathcal{O}(n)$ nodes to the original diagram.

Proof By induction on the term construction. Notice that the instantiation of any term except the gather does not introduce any new nodes, and the gather introduction creates exactly one extra node. Therefore, the list instantiation adds a number of nodes equal to the number of gather generators in the diagram. \square

Lemma E.5 Given type judgements $\Phi, x : A \vdash M : B$, and $\Phi \vdash N : A$. $[M]_{x:A,\Phi}([N]_{\Phi}, |\Phi|) = [M[N/x]]_{\Phi}(|\Phi|)$.

Proof Proof by straightforward induction on M . \square

Lemma (4.1) Given an evaluable type A and a type judgement $\Phi \vdash M : A$, $[M]_{\Phi} \in \times_{x:P \in \Phi} [P] \rightarrow [A]$.

Proof By induction on the typing judgement $\Phi \vdash M : A$:

- If $\Phi \vdash n : \text{Nat}$, then $[n]_{\Phi} = |\Phi| \mapsto n \in \times_{x:P \in \Phi} [P] \rightarrow \mathbb{N}$.
- If $\Phi, x : A \vdash x : A$, then $[x]_{x:A,\Phi} = x, |\Phi| \mapsto x \in \times_{y:P \in x:A,\Phi} [P] \rightarrow [A]$.
- If $\Phi \vdash M \square N : \text{Nat} \Phi, \Gamma, \Delta \vdash M :: N : \text{Vec } (n+1) A$, then $[M \square N]_{\Phi} = |\Phi| \mapsto [M]_{\Phi}(|\Phi|) \square [N]_{\Phi}(|\Phi|)$. By inductive hypothesis, $[M]_{\Phi}(|\Phi|), [N]_{\Phi}(|\Phi|) \in \mathbb{N}$. Then, $|\Phi| \mapsto [M]_{\Phi}(|\Phi|) \square [N]_{\Phi}(|\Phi|) \in \times_{x:P \in \Phi} [P] \rightarrow \mathbb{N}$.
- If $\Phi \vdash \lambda' x. M : (x : P) \rightarrow B$ then $[\lambda' x^P. M]_{\Phi} = x, |\Phi| \mapsto [M]_{\Phi}(x, |\Phi|)$. By inductive hypothesis, $[M]_{\Phi}(x, |\Phi|) \in [B]$. Then, $|\Phi| \mapsto [M]_{\Phi}(x, |\Phi|) \in \times_{y:P \in x:P\Phi} [P] \rightarrow [B]$.
- If $\Phi, \Gamma \vdash M @ N : B[x/r]$, then $[M @ N]_{\Phi} = |\Phi| \mapsto [M]_{\Phi}([N]_{\Phi}(|\Phi|), \Phi)$. By inductive hypothesis, $[N]_{\Phi}(|\Phi|) \in \mathbb{N}$ and $x \mapsto [M]_{\Phi}(x, \Phi) \in [x : \text{Nat} \rightarrow B[x]]$. Then, $|\Phi| \mapsto [M]_{\Phi}([N]_{\Phi}(|\Phi|), \Phi) \in \times_{y:P \in \Phi} [P] \rightarrow [B[A/x]]$.
- If $\Phi \vdash \text{VNil}^A : \text{Vec } 0 A$, then $[\text{VNil}^P]_{\Phi} = |\Phi| \mapsto [] \in \times_{x:P \in \Phi} [P] \rightarrow \mathbb{N}^0$.
- If $\Phi, \Gamma, \Delta \vdash M :: N : \text{Vec } (n+1) A$, then $[M :: N]_{\Phi} = |\Phi| \mapsto [M]_{\Phi}(|\Phi|) \times [N]_{\Phi}(|\Phi|)$. By inductive hypothesis $[M]_{\Phi}(|\Phi|) \in [A]$ and $[N]_{\Phi}(|\Phi|) \in [A]^n$. Then, $|\Phi| \mapsto [M]_{\Phi}(|\Phi|) \times [N]_{\Phi}(|\Phi|) \in \times_{x:P \in \Phi} [P] \rightarrow [A]^{n+1}$.
- If $\Phi, \Gamma \vdash \text{ifz } L \text{ then } M \text{ else } N : A$, then

$$[\text{ifz } L \text{ then } M \text{ else } N]_{\Phi} = |\Phi| \mapsto \begin{cases} [M]_{\Phi}(|\Phi|) & \text{if } [L]_{\Phi}(|\Phi|) = 0 \\ [N]_{\Phi}(|\Phi|) & \text{otherwise} \end{cases}.$$

By inductive hypothesis $[m]_{\Phi}(|\Phi|), [n]_{\Phi}(|\Phi|) \in [A]$ and $[L]_{\Phi}(|\Phi|) \in \mathbb{N}$. Then $[\text{ifz } L \text{ then } M \text{ else } N]_{\Phi} \in \times_{x:P \in \Phi} [P] \rightarrow [A]$.

- If $\Phi \vdash \text{for } k \text{ in } N \text{ do } M : \text{Vec } n \ A$, then $\llbracket \text{for } k \text{ in } V \text{ do } M \rrbracket_{\Phi} = |\Phi\rangle \mapsto \times_{k \in [N]_{\Phi}(|\Phi\rangle)} \llbracket M \rrbracket_{k:\text{Nat}\Phi} (k, |\Phi\rangle)$. By induction hypothesis, $\llbracket N \rrbracket_{\Phi} (|\Phi\rangle) \in \text{Nat}^n$ and $x \mapsto \llbracket M \rrbracket_{\Phi} (x, \Phi) \in [k : \text{Nat} \rightarrow A[k]]$. Then $|\Phi\rangle \mapsto \times_{k \in [N]_{\Phi}(|\Phi\rangle)} \llbracket M \rrbracket_{k:\text{Nat}\Phi} (k, |\Phi\rangle) \in \times_{x:P \in \Phi} [P] \rightarrow [A[k/N]]^n$.
- If $\Phi \vdash \text{let } x^B : y^{\text{Vec } n \ B} = M \text{ in } N : A$, then $\llbracket \text{let } x^P :: y^{\text{Vec } n \ P} = M \text{ in } N \rrbracket_{\Phi} = |\Phi\rangle \mapsto [N]_{x:P,y:\text{Vec } n \ P\Phi} (y_1, [y_2, \dots, y_n], |\Phi\rangle)$ where $[y_1, \dots, y_n] = \llbracket M \rrbracket_{\Phi} (|\Phi\rangle)$.
By inductive hypothesis $\llbracket M \rrbracket_{\Phi} (|\Phi\rangle) \in [B]^n$ and $[N]_{x:P,y:\text{Vec } n \ P\Phi} (y_1, [y_2, \dots, y_n], |\Phi\rangle) \in [A]$. Then $|\Phi\rangle \mapsto [N]_{x:P,y:\text{Vec } n \ P\Phi} (y_1, [y_2, \dots, y_n], |\Phi\rangle) \in \times_{x:P \in \Phi} [P] \rightarrow [A]$.
- If $\Phi \vdash \text{range} : (n : \text{Nat}) \rightarrow (m : \text{Nat}) \rightarrow \text{Vec } (m - n) \ \text{Nat}$, then $\llbracket \text{range} \rrbracket_{\Phi} = n, m, |\Phi\rangle \mapsto \times_{i=n}^{m-1} i \in \times_{z:P \in x:\text{Nat}, y:\text{Nat}, \Phi} [P] \rightarrow \mathbb{N}^{m-n}$ \square

Lemma (4.2) Given an evaluable type A , a type judgement $\Phi \vdash M : A$, and $M \rightarrow N$, then $\llbracket M \rrbracket_{\Phi} = \llbracket N \rrbracket_{\Phi}$.

Proof By induction on the evaluation function $\llbracket M \rrbracket_{\Phi}$:

- If $M = x$, $M = n$, $M = \text{VNil}^{\text{Nat}}$, $M = \lambda'x.M'$, $M = M_1 :: M_2$ or $M = \text{range}$ then M is in normal form and it does not reduce.
- If $M = M_1 \square M_2$ we have three cases:
 - If $M \rightarrow M_1 \square N$ with $M_2 \rightarrow N$, then:

$$\begin{aligned} \llbracket M_1 \square M_2 \rrbracket_{\Phi} &= |\Phi\rangle \mapsto \llbracket M_1 \rrbracket_{\Phi} (|\Phi\rangle) \square \llbracket M_2 \rrbracket_{\Phi} (|\Phi\rangle) \\ &= |\Phi\rangle \mapsto \llbracket M_1 \rrbracket_{\Phi} (|\Phi\rangle) \square \llbracket N \rrbracket_{\Phi} (|\Phi\rangle) \\ &= \llbracket M_1 \square N \rrbracket_{\Phi} \end{aligned}$$

- If $M \rightarrow N \square V$ with $M_1 \rightarrow N$, then:

$$\begin{aligned} \llbracket M_1 \square V \rrbracket_{\Phi} &= |\Phi\rangle \mapsto \llbracket M_1 \rrbracket_{\Phi} (|\Phi\rangle) \square \llbracket V \rrbracket_{\Phi} (|\Phi\rangle) \\ &= |\Phi\rangle \mapsto \llbracket N \rrbracket_{\Phi} (|\Phi\rangle) \square \llbracket V \rrbracket_{\Phi} (|\Phi\rangle) \\ &= \llbracket N \square V \rrbracket_{\Phi} \end{aligned}$$

- If $M \rightarrow n$ with $M_i = n_i \in \mathbb{N}$ and $n = n_1 \square n_2$, then:

$$\begin{aligned} \llbracket n_1 \square n_2 \rrbracket_{\Phi} &= |\Phi\rangle \mapsto \llbracket n_1 \rrbracket_{\Phi} (|\Phi\rangle) \square \llbracket n_2 \rrbracket_{\Phi} (|\Phi\rangle) \\ &= |\Phi\rangle \mapsto n_1 \square n_2 \\ &= |\Phi\rangle \mapsto n \\ &= \llbracket n \rrbracket_{\Phi} \end{aligned}$$

- If $M = M_1 @ M_2$ we have three cases:
 - If $M \rightarrow M_1 @ N$ with $M_2 \rightarrow N$, then:

$$\begin{aligned} \llbracket M_1 @ M_2 \rrbracket_{\Phi} &= |\Phi\rangle \mapsto \llbracket M_1 \rrbracket_{\Phi} (\llbracket M_2 \rrbracket_{\Phi} (|\Phi\rangle), \Phi) \\ &= |\Phi\rangle \mapsto \llbracket M_1 \rrbracket_{\Phi} (\llbracket N \rrbracket_{\Phi} (|\Phi\rangle), \Phi) \\ &= \llbracket M_1 @ N \rrbracket_{\Phi} \end{aligned}$$

- If $M \rightarrow N @ V$ with $M_1 \rightarrow N$, then:

$$\begin{aligned} [M_1 @ V]_{\Phi} &= |\Phi| \mapsto [M_1]_{\Phi}([V]_{\Phi}(|\Phi|), \Phi) \\ &= |\Phi| \mapsto [M_1]_{\Phi}([V]_{\Phi}(|\Phi|), \Phi) \\ &= [N @ V]_{\Phi} \end{aligned}$$

- If $M \rightarrow M'[V/x]$ with $M_1 = \lambda'x.M'$ and $M_2 = V$, then:

$$\begin{aligned} [(\lambda'x.M)@V]_{\Phi} &= |\Phi| \mapsto [\lambda'x.M]_{\Phi}([V]_{\Phi}(|\Phi|), \Phi) \\ &= |\Phi| \mapsto (x, |\Phi| \mapsto [M]_{x,\Phi}(x, |\Phi|))([V]_{\Phi}(|\Phi|), \Phi) \\ &= |\Phi| \mapsto [M[V/X]]_{\Phi}(|\Phi|) \\ &= [M[V/X]]_{\Phi} \end{aligned}$$

- If $M = \text{ifz } M' \text{ then } L \text{ else } R$ we have three cases:

- If $M \rightarrow \text{ifz } N \text{ then } L \text{ else } R$ with $M' \rightarrow N$, then:

$$\begin{aligned} [\text{ifz } M' \text{ then } L \text{ else } R]_{\Phi} &= |\Phi| \mapsto \begin{cases} [M]_{\Phi}(|\Phi|) & \text{if } [M']_{\Phi}(|\Phi|) = 0 \\ [N]_{\Phi}(|\Phi|) & \text{otherwise} \end{cases} \\ &= |\Phi| \mapsto \begin{cases} [M]_{\Phi}(|\Phi|) & \text{if } [M']_{\Phi}(|\Phi|) = 0 \\ [N]_{\Phi}(|\Phi|) & \text{otherwise} \end{cases} \\ &= [\text{ifz } N \text{ then } L \text{ else } R]_{\Phi} \end{aligned}$$

- If $M \rightarrow L$ with $M' = 0$, then:

$$\begin{aligned} [\text{ifz } M' \text{ then } L \text{ else } R]_{\Phi} &= |\Phi| \mapsto \begin{cases} [M]_{\Phi}(|\Phi|) & \text{if } [M']_{\Phi}(|\Phi|) = 0 \\ [N]_{\Phi}(|\Phi|) & \text{otherwise} \end{cases} \\ &= |\Phi| \mapsto [L]_{\Phi}(|\Phi|) \\ &= [L]_{\Phi} \end{aligned}$$

- The symmetric case for the else branch is similar to the previous one.

- If $M = \text{for } k \text{ in } M' \text{ do } R$ we have three cases:

- If $M \rightarrow \text{for } k \text{ in } N \text{ do } R$ with $M' \rightarrow N$, then:

$$\begin{aligned} [\text{for } k \text{ in } M' \text{ do } R]_{\Phi} &= |\Phi| \mapsto \bigtimes_{k \in [M']_{\Phi}(|\Phi|)} [R]_{\Phi}(k, |\Phi|) \\ &= |\Phi| \mapsto \bigtimes_{k \in [N]_{\Phi}(|\Phi|)} [R]_{\Phi}(k, |\Phi|) \\ &= [\text{for } k \text{ in } N \text{ do } R]_{\Phi} \end{aligned}$$

- If $M \rightarrow R[k/M_1] : \text{for } k \text{ in } M_2 \text{ do } R$ with $M' = V :: L$, then:

$$\begin{aligned}
[\text{for } k \text{ in } M_1 :: M_2 \text{ do } R]_{\Phi} &= |\Phi\rangle \mapsto \bigtimes_{k \in [M_1 :: M_2]_{\Phi}(|\Phi\rangle)} [R]_{\Phi}(k, |\Phi\rangle) \\
&= |\Phi\rangle \mapsto \bigtimes_{k \in [M_1]_{\Phi}(|\Phi\rangle) \times [M_2]_{\Phi}(|\Phi\rangle)} [R]_{\Phi}(k, |\Phi\rangle) \\
&= |\Phi\rangle \mapsto [R]_{k, \Phi}([M_1]_{\Phi}(|\Phi\rangle), |\Phi\rangle) \times \bigtimes_{k \in [M_2]_{\Phi}(|\Phi\rangle)} [R]_{k, \Phi}(k, |\Phi\rangle) \\
&= [R[M_1/k]]_{\Phi} \times [\text{for } k \text{ in } M_2 \text{ do } R]_{\Phi} \\
&= [R[M_1/k] :: \text{for } k \text{ in } M_2 \text{ do } R]_{\Phi}
\end{aligned}$$

- If $M \rightarrow V\text{Nil}$ with $M' = V\text{Nil}^{\text{Nat}}$, then:

$$\begin{aligned}
[\text{for } k \text{ in } V\text{Nil}^{\text{Nat}} \text{ do } R]_{\Phi} &= |\Phi\rangle \mapsto \bigtimes_{k \in [V\text{Nil}^{\text{Nat}}]_{\Phi}(|\Phi\rangle)} [R]_{\Phi}(k, |\Phi\rangle) \\
&= |\Phi\rangle \mapsto \bigtimes_{k \in []} [R]_{\Phi}(k, |\Phi\rangle) \\
&= |\Phi\rangle \mapsto [] \\
&= [V\text{Nil}^{\text{Nat}}]_{\Phi}
\end{aligned}$$

- If $M = \text{let } x :: y = M_1 \text{ in } M_2$ we have two cases:

- If $M \rightarrow \text{let } x :: y = N \text{ in } M_2$ with $M_1 \rightarrow N$, then:

$$\begin{aligned}
[\text{let } x :: y = M_1 \text{ in } M_2]_{\Phi} &= |\Phi\rangle \mapsto [M_2]_{\Phi}(y_1, [y_2, \dots, y_n], |\Phi\rangle) \text{ where } [y_1, \dots, y_n] = [M_1]_{\Phi}(|\Phi\rangle) \\
&= |\Phi\rangle \mapsto [M_2]_{\Phi}(y_1, [y_2, \dots, y_n], |\Phi\rangle) \text{ where } [y_1, \dots, y_n] = [N]_{\Phi}(|\Phi\rangle) \\
&= [\text{let } x :: y = N \text{ in } M_2]_{\Phi}
\end{aligned}$$

- If $M \rightarrow N[x/M_1][y/M_2]$ with $M' = M_1 :: M_2$, then:

$$\begin{aligned}
[\text{for } k \text{ in } M_1 :: M_2 \text{ do } N]_{\Phi} &= |\Phi\rangle \mapsto [N]_{x, y, \Phi}(y_1, [y_2, \dots, y_n], |\Phi\rangle) \\
&= |\Phi\rangle \mapsto [N]_{x, y, \Phi}(M_1, M_2, |\Phi\rangle) \\
&= |\Phi\rangle \mapsto [N[M_1/x][M_2/y]]_{\Phi}(|\Phi\rangle) \\
&= [N[M_1/x][M_2/y]]_{\Phi}
\end{aligned}$$

where $[y_1, \dots, y_n] = [M_1 :: M_2]_{\Phi}(|\Phi\rangle)$.

- If $M = \text{range } @n @M_2$ then:

$$\begin{aligned}
[\text{range } @n @m]_{\Phi} &= |\Phi\rangle \mapsto \bigtimes_{i=n}^{m-1} i \\
&= |\Phi\rangle \mapsto \begin{cases} [] & \text{if } n - m = 0 \\ n \times \bigtimes_{i=n+1}^{m-1} i & \text{otherwise} \end{cases} \\
&= |\Phi\rangle \mapsto \begin{cases} [] & \text{if } [n - m]_{\Phi} = 0 \\ [n]_{\Phi} \times [\text{range } @(n+1) @m] & \text{otherwise} \end{cases} \\
&= [\text{ifz } m - n \text{ then } V\text{Nil} \text{ else } n :: \text{range } @(n+1) @m]_{\Phi} \quad \square
\end{aligned}$$

Lemma (4.3) The translation procedure is correct in respect to the operational semantics. If A is a translatable type, $\Phi, \Gamma \vdash M : A$, and $M \rightarrow N$, then $\llbracket M \rrbracket_{\Phi, \Gamma} = \llbracket N \rrbracket_{\Phi, \Gamma}$.

Proof By case analysis on the reductions of translatable terms.

- If $M = (\lambda x^A. M')V$ and $N = M'[V/x]$,

$$\begin{aligned} \llbracket (\lambda x^A. M')V \rrbracket_{\Phi, \Delta, \Gamma} = |\Phi| \mapsto & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{V(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} A \quad A \rightarrow B \\ \text{---} \quad \text{---} \end{array} \end{array} \\ \stackrel{\text{(gs)}}{=} |\Phi| \mapsto & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{V(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} = \llbracket M'[V/x] \rrbracket_{\Phi, \Delta, \Gamma} \end{aligned}$$

- If $M = (\lambda' x^A. M')@V$ and $N = M'[V/x]$,

$$\begin{aligned} \llbracket (\lambda' x^A. M')@V \rrbracket_{\Phi, \Gamma} = |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{[(\lambda' x^A. M')]_{\Phi}(\lfloor V \rfloor_{\Phi}(|\Phi|), |\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} \\ = |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{[(M')_{\Phi}(\lfloor V \rfloor_{\Phi}(|\Phi|), |\Phi|)]} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} = \llbracket M'[V/x] \rrbracket_{\Phi, \Gamma} \end{aligned}$$

- If $M = \text{let } x^A \otimes y^B = V_1 \otimes V_2 \text{ in } M'$ and $N = M'[V_1/x][V_2/y]$,

$$\begin{aligned} \llbracket \text{let } x^A \otimes y^B = V_1 \otimes V_2 \text{ in } M' \rrbracket_{\Phi, \Gamma, \Delta, \Lambda} = |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} A \otimes B \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} C \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Lambda \\ \text{---} \\ \text{---} \end{array} \\ \stackrel{\text{(gs)}}{=} |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Lambda \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} C \\ \text{---} \\ \text{---} \end{array} = \llbracket M'[V_1/x][V_2/y] \rrbracket_{\Phi, \Delta, \Gamma, \Lambda} \end{aligned}$$

- If $M = \text{let } x^A :: y^{\text{vec } n A} = V_1 :: V_2 \text{ in } M'$ and $N = M'[V_1/x][V_2/y]$,

$$\begin{aligned} \llbracket \text{let } x^A :: y^{\text{vec } n A} = V_1 :: V_2 \text{ in } M' \rrbracket_{\Phi, \Gamma, \Delta, \Lambda} = |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} nA \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} (n+1)A \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Lambda \\ \text{---} \\ \text{---} \end{array} \\ \stackrel{\text{(gs)}}{=} |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Delta \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} nA \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Lambda \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \text{---} \\ \text{---} \end{array} = \llbracket M'[V_1/x][V_2/y] \rrbracket_{\Phi, \Delta, \Gamma, \Lambda} \end{aligned}$$

- If $M = \text{ifz } L \text{ then } M' \text{ else } N'$,

– if $L = 0$ and $N = M'$, $\lfloor L \rfloor_{\Phi} = 0$ and

$$\begin{aligned} \llbracket \text{ifz } L \text{ then } M' \text{ else } N' \rrbracket_{\Phi, \Gamma} = |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{N'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} 0 \quad 0 \end{array} \\ \stackrel{\text{Lemma 2.3}}{=} |\Phi| \mapsto & \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} \\ & \begin{array}{c} 0 \quad 0 \end{array} \stackrel{\text{(04)}}{=} |\Phi| \mapsto \begin{array}{c} \Gamma \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \boxed{M'(|\Phi|)} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} A \\ \text{---} \\ \text{---} \end{array} = \llbracket M' \rrbracket_{\Phi, \Gamma} \end{aligned}$$

– The case where $L > 0$ and $N = N'$ is symmetric to the case above.

- If $M = \text{VNil}; M'$ and $N = M'$,

$$\llbracket \text{VNil}; M' \rrbracket_{\Phi, \Gamma} = |\Phi\rangle \mapsto \Gamma \begin{array}{c} \boxed{} \\ M'(|\Phi\rangle) \end{array} \xrightarrow{A} \llbracket M' \rrbracket_{\Phi, \Gamma}$$

- If $M = \star; M'$ and $N = M'$,

$$\llbracket \star; M' \rrbracket_{\Phi, \Gamma} = |\Phi\rangle \mapsto \Gamma \begin{array}{c} \boxed{} \\ M'(|\Phi\rangle) \end{array} \xrightarrow{A} \llbracket M' \rrbracket_{\Phi, \Gamma}$$

- If $M = \text{for } k \text{ in } V :: M' \text{ do } N'$ and $N = N'[k/V] :: \text{for } k \text{ in } M' \text{ do } N'$,

$$\llbracket \text{for } k \text{ in } V :: M' \text{ do } N' \rrbracket_{\Phi, \Gamma^{\otimes n}} = |\Phi\rangle \mapsto \Gamma^{\otimes n} \begin{array}{c} \boxed{N'(k, |\Phi\rangle)} \\ k \in [V :: M'](|\Phi\rangle) \end{array} \xrightarrow{A^{\otimes n}}$$

Lemma 2.2 $\llbracket \text{for } k \text{ in } V :: M' \text{ do } N' \rrbracket_{\Phi, \Gamma^{\otimes n}} = |\Phi\rangle \mapsto \Gamma^{\otimes n} \begin{array}{c} \boxed{N'([V](|\Phi\rangle), |\Phi\rangle)} \\ N'(k, |\Phi\rangle) \\ k \in [M'](|\Phi\rangle) \end{array} \xrightarrow{A^{\otimes n}} \llbracket N'[k/V] :: \text{for } k \text{ in } M' \text{ do } N' \rrbracket_{\Phi, \Gamma^{\otimes n}}$

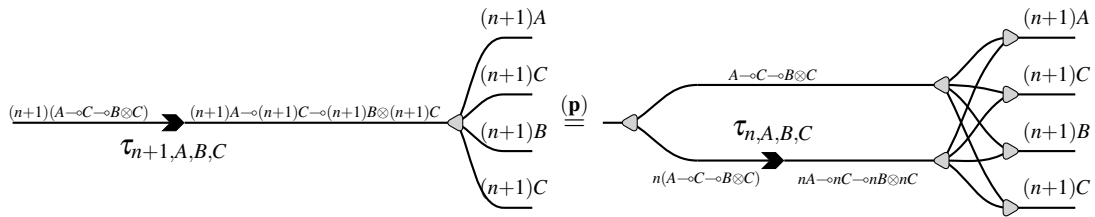
- If $M = \text{for } k \text{ in VNil do } N'$ and $N = \text{VNil}$,

$$\llbracket \text{for } k \text{ in VNil do } N' \rrbracket_{\Phi} = |\Phi\rangle \mapsto \begin{array}{c} 0 \\ \boxed{N'(k, |\Phi\rangle)} \\ k \in [] \end{array} \xrightarrow{0}$$

$$\text{Lemma 2.3 } |\Phi\rangle \mapsto \begin{array}{c} 0 \\ \triangleleft \end{array} \quad \begin{array}{c} 0 \\ \triangleright \end{array} = |\Phi\rangle \mapsto \begin{array}{c} \boxed{} \\ \text{VNil} \end{array} = \llbracket \text{VNil} \rrbracket_{\Phi}$$

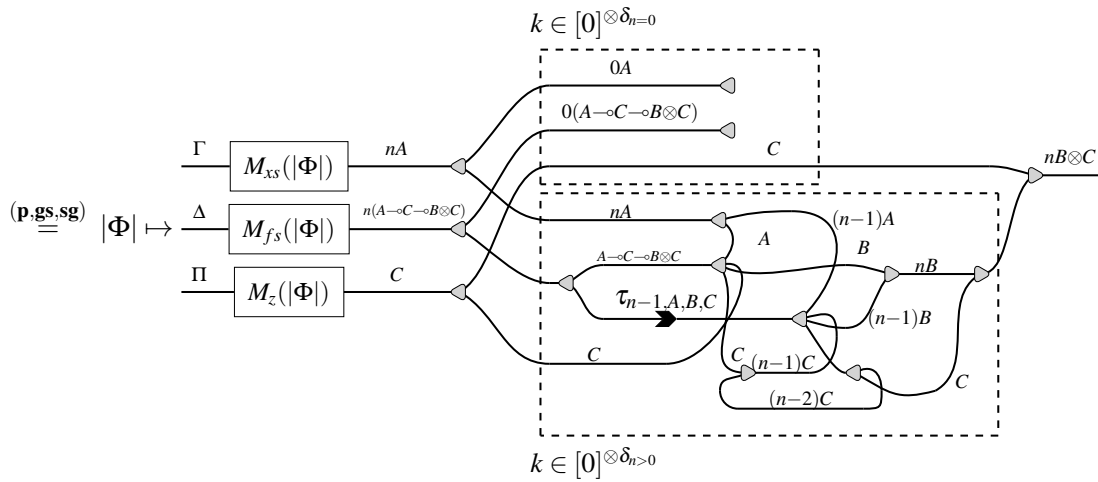
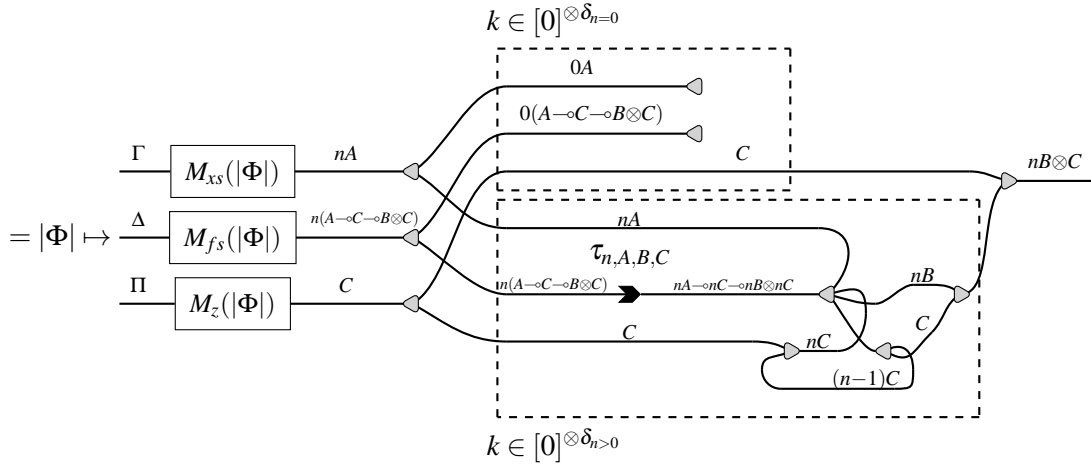
- If $M = \text{accuMap}_{A,B,C} N' M_{xs} M_{fs} M_z$ and $N = \text{ifz } N'$ then $xs ;_v fs ;_v \text{VNil} \otimes M_z$ else let $x :: xs' = M_{xs}$ in let $f :: fs' = M_{fs}$ in let $y \otimes z' = f x z$ in let $ys \otimes z'' = \text{accuMap } @ (N' - 1) xs' fs' z'$ in $(y :: ys) \otimes z''$. Let $n = \lfloor N_1 \rfloor_{\Phi} (|\Phi\rangle)$.

Notice that, by definition of $\tau_{n,A,B,C}$,



Therefore,

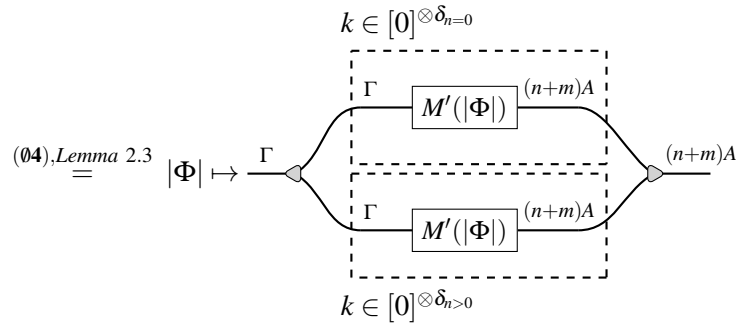
$$\llbracket \text{accuMap}_{A,B,C} N' M_{xs} M_{fs} M_z \rrbracket_{\Phi, \Gamma, \Delta, \Pi}$$

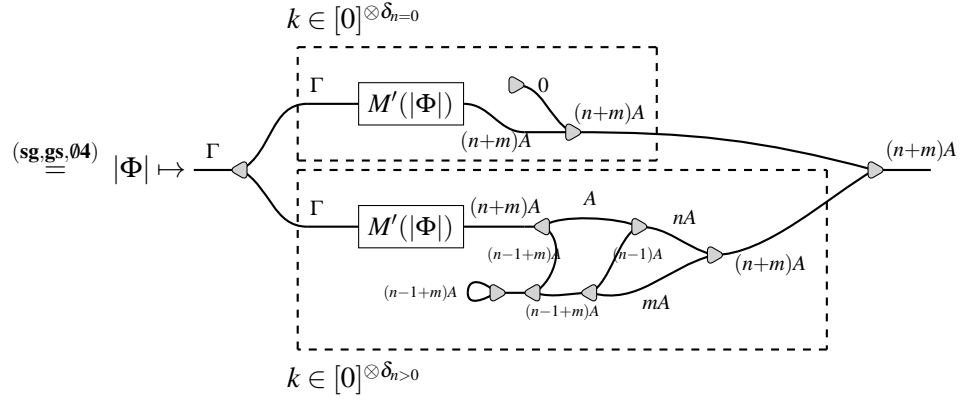


$$= \llbracket \text{ifz } N' \text{ then } xs ;_v fs ;_v \text{VNil} \otimes M_z \text{ else let } x :: xs' = M_{xs} \text{ in let } f :: fs' = M_{fs} \text{ in let } y \otimes z' = f x z \text{ in let } ys \otimes z'' = \text{accuMap } @ (N' - 1) xs' fs' z' \text{ in } (y :: ys) \otimes z'' \rrbracket_{\Phi, \Gamma, \Delta, \Pi}$$

- If $M = \text{split}_A @n @m xs$ and $N = \text{ifz } n \text{ then } \text{VNil} \otimes xs \text{ else let } y :: xs' = xs \text{ in let } ys_1 \otimes ys_2 = \text{split}@ (n-1) @m xs' \text{ in } (y :: ys_1) \otimes ys_2$. Let $n = \lfloor N_1 \rfloor_{\Phi}(|\Phi|)$ and $m = \lfloor N_2 \rfloor_{\Phi}(|\Phi|)$.

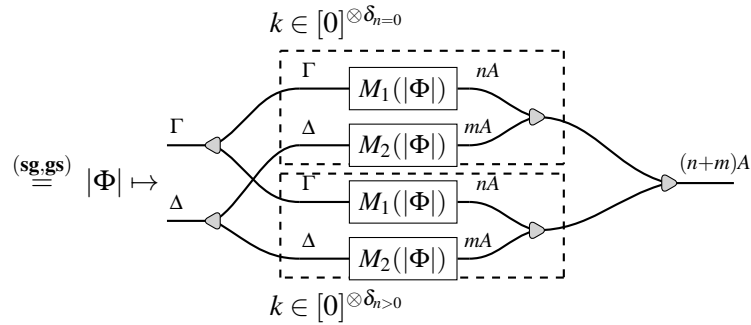
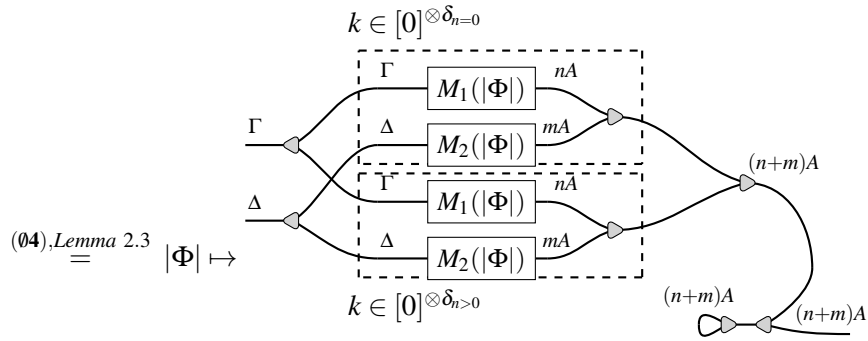
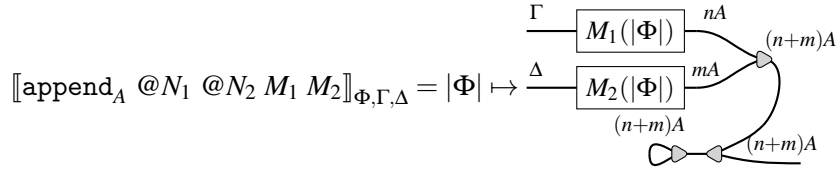
$$\llbracket \text{split}_A @n @m xs \rrbracket_{\Phi, \Gamma} = |\Phi| \mapsto \Gamma \xrightarrow{(n+m)A} \boxed{M'(|\Phi|)} \xrightarrow{(n+m)A} \Gamma \quad \stackrel{(\text{sg,gs})}{=} \quad |\Phi| \mapsto \Gamma \xrightarrow{(n+m)A} \boxed{M'(|\Phi|)} \xrightarrow{(n+m)A} \Gamma$$

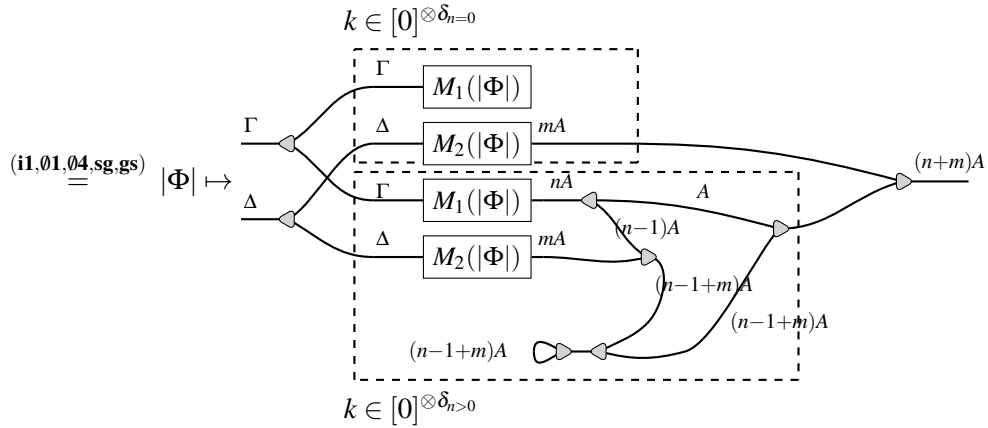




$= \llbracket \text{ifz } n \text{ then } \text{VNil} \otimes xs \text{ else let } y :: xs' = xs \text{ in let } ys_1 \otimes ys_2 = \text{split}@ (n-1) @m xs' \text{ in } (y :: ys_1) \otimes ys_2 \rrbracket_{\Phi, \Gamma}$

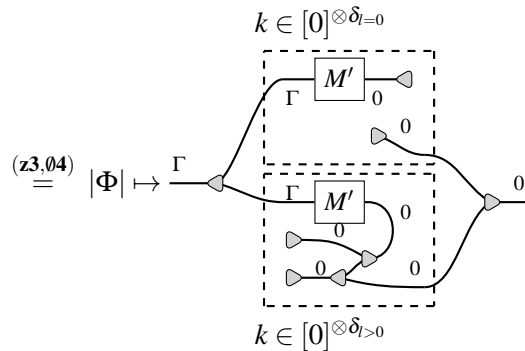
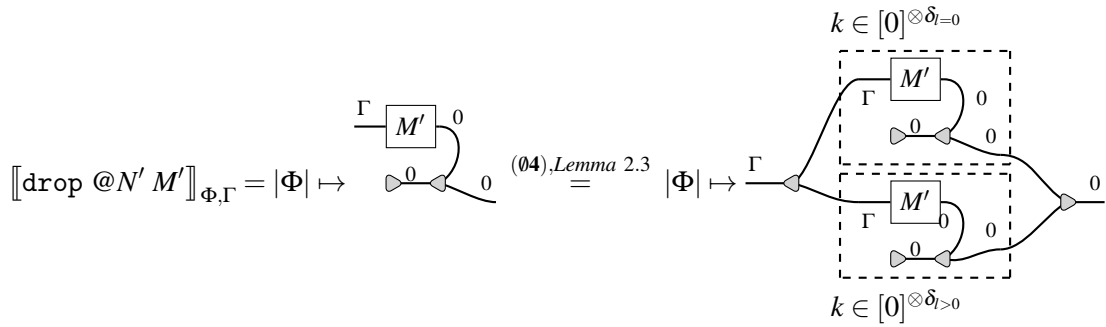
- If $M = \text{append}_A @N_1 @N_2 M_1 M_2$ and $N = \text{ifz } N_1 \text{ then } M_1 ;_v M_2 \text{ else let } x :: xs' = M_1 \text{ in } x :: (\text{append } @(N_1 - 1) @N_2 M_1 M_2)$. Let $n = \lfloor N_1 \rfloor_{\Phi}(|\Phi|)$ and $m = \lfloor N_2 \rfloor_{\Phi}(|\Phi|)$.





$$= \llbracket \text{ifz } N_1 \text{ then } M_1 ;_v M_2 \text{ else let } x :: xs' = M_1 \text{ in } x :: (\text{append } @(N_1 - 1) @N_2 M_1 M_2) \rrbracket_{\Phi, \Gamma, \Delta}$$

- If $M = \text{drop } @N' M'$ and $N = \text{ifz } N' \text{ then } M' ; \star \text{ else let } x :: xs' = M' \text{ in } x ; \text{drop } @(N' - 1) xs'$. Let $l = \llbracket N' \rrbracket_{\Phi}(|\Phi|)$,



$$= \llbracket \text{ifz } N' \text{ then } M' ; \star \text{ else let } x :: xs' = M' \text{ in } x ; \text{drop } @(N' - 1) xs' \rrbracket_{\Phi, \Gamma}$$

- If $M \rightarrow N$ is an internal reduction of a translatable term, then the diagrams result equivalent via the inductive hypothesis.
- If $M \rightarrow N$ is an internal reduction of an evaluable term, then the diagrams result equivalent via the inductive hypothesis and Lemma 4.2. \square

Q# as a Quantum Algorithmic Language

Kartik Singhal

University of Chicago
ks@cs.uchicago.edu

Kesha Hietala

University of Maryland
kesha@cs.umd.edu

Sarah Marshall

Microsoft Quantum
sarah@sarahmarshall.name

Robert Rand

University of Chicago
rand@uchicago.edu

Q# is a standalone domain-specific programming language from Microsoft for writing and running quantum programs. Like most industrial languages, it was designed without a formal specification, which can naturally lead to ambiguity in its interpretation. We aim to provide a formal language definition for Q#, placing the language on a solid mathematical foundation and enabling further evolution of its design and type system. This paper presents $\lambda_{Q\#}$, an idealized version of Q# that illustrates how we may view Q# as a quantum ALGOL (algorithmic language). We show the safety properties enforced by $\lambda_{Q\#}$'s type system and present its equational semantics based on a fully complete algebraic theory by Staton.

1 Introduction

Microsoft's Q# programming language [56] is one of the most full-featured quantum programming languages that have emerged from the recent boom in quantum computing research. However, with a growing code base and increasing popularity comes the demand for more features and the resulting added complexity. Hence, Q# faces challenges familiar to many growing programming languages—maintaining correctness, ease of use, and intuitive understanding while evolving to meet users' needs.

Quantum programming languages face unique challenges that are not present in classical languages. Quantum algorithms are more challenging to design and reason about than classical ones as they use quantum phenomena like superposition and entanglement. Quantum programs are challenging to test and debug. Their simulation on classical computers is slow and limited to a handful of qubits while languages like Q# are designed for large-scale fault-tolerant quantum computers with thousands of logical qubits. When running a quantum program directly on quantum hardware, we cannot observe the whole quantum state directly, and measuring a classical result during execution (partial observation) may itself destroy the state. Additionally, existing quantum hardware provides limited qubit count and poor gate fidelity.

These challenges underscore why it is essential for Q# to have a well-specified definition that can serve as a foundation for extensions, multiple implementations, and formal verification of programs written in the language. A formal specification and mechanization of its metatheory will help ensure that Q# is robust enough to meet the unique needs of the developing field of quantum software engineering.

A tried-and-tested approach to achieving this ambitious goal is to define an idealized core version of the language, provide an elaboration from the surface language to this core language, and provide static and dynamic semantics for the core. In this paper, we argue that even though Q# is a relatively large language, we can condense it to a small core capturing most of its interesting features. We call this core $\lambda_{Q\#}$. In it, we make several implicit features of Q# explicit: its treatment of qubits as references, its stack-like memory management that enables reasoning about the quantum state in a local manner, and its safe synthesis of effectful and pure computation. In the classical setting, this stack-like memory management and combination of effectful and pure computation are inherent to many ALGOL-like languages [48].

Contributions In 2018, when introducing Q#, its designers stated that as opposed to several existing *circuit definition* languages, “Q# is an *algorithm definition* language” [56]; the goal of our paper is to show that, in its essence, Q# is a quantum *algorithmic language* (ALGOL).

- To support this characterization, we introduce $\lambda_{Q\#}$, an idealized version of Q# inspired by Harper’s language MA (Modernized Algol) [17]. In $\lambda_{Q\#}$, we expose values of the `Qubit` type in Q# as references to logical qubits and formalize the ALGOL-like stack discipline implicit in Q#’s quantum memory management.
- We develop a type system for $\lambda_{Q\#}$ that extends Q#’s type system to enforce the no-cloning theorem and stack-like management of qubits.
- We provide an equational dynamics for $\lambda_{Q\#}$ building upon the fully complete equational theory of quantum computation by Staton [54].
- Finally, we provide an elaboration relation from Q# to $\lambda_{Q\#}$, thereby endowing a significant portion of Q# with a formal specification and additional safety guarantees.

Outline In the rest of the paper, we review background on the Q# programming language and Staton’s theory for quantum computation (§2); introduce $\lambda_{Q\#}$ along with its syntax and semantics (§3); describe how $\lambda_{Q\#}$ is faithful to the surface Q# language (§4); and discuss related and future work (§5 and §6).

2 Background

Before introducing $\lambda_{Q\#}$, we discuss the two projects that inspired our work. The first is Microsoft’s Q# [56], a modern, self-contained quantum programming language that boasts a large community of developers. The second is Sam Staton’s equational theory for quantum programs [54], which provides a compelling alternative to the standard matrix-based semantics for quantum programs.

2.1 The Q# Programming Language

Q# [56] is a hybrid quantum-classical programming language that supports interlacing stateful quantum *operations* with pure classical *functions*, collectively referred to as *callable*s. Q# encourages thinking about quantum programs as algorithms rather than circuits, where quantum operations can be combined with classical control flow such as branches and loops. When a programmer measures a qubit, they can perform an arbitrary classical computation on the result, and the program execution can continue without requiring the qubit to be released. This computational model allows quantum and classical algorithms to be fully mixed. At the same time, Q# enforces a degree of separation between the quantum and classical components. Operations can call functions, but functions cannot call operations. An example Q# program, implementing the quantum teleportation protocol, is shown in Listing 1 in Appendix D.

Q# contains a blend of functional and imperative features (it evolved from an F#-like language [4]). For classical data, Q# follows the so-called value semantics [22], perhaps better known as referential transparency. Q# functions are always pure, and variable bindings are immutable by default. Bindings may be declared `mutable`, but they correspond to a local state change, enclosed in the scope of the parent callable. Hence, equational reasoning is possible across function boundaries. By contrast, qubits are opaque types that act as references to logical qubits [11, 12]—their values are never exposed. Gate operations are inherently *effectful*: a (single-qubit) quantum gate application is a procedure that takes a qubit reference as input and returns a trivial output of type `Unit` after altering the quantum state.

```

operation NewQubit () : Qubit {
    use q = Qubit();
    return q;
}

operation Clone () : Unit {
    use q1 = Qubit();
    let q2 = q1;
    CNOT(q1, q2);
}

```

(a) Returning a qubit after its lifetime has ended.

(b) Using the same qubit as both control and target.

Figure 1: Sample unsafe Q# programs.

Callables in Q# can be *higher-order*: functions and operations are values and can be given as arguments to, or returned by, other functions and operations. Both functions and operations can be partially applied. Quantum algorithms parameterized by quantum subroutines are easily expressed in Q# using higher-order operations. For example, an operation implementing Grover’s search [15] can accept an oracle as a parameter and apply it in each iteration.

Q# supports a restricted form of metaprogramming, where the compiler can automatically generate the adjoint and controlled versions of unitary operations. Operations can declare their support for `Adjoint` and `Controlled` *functors* (in Q#’s terminology¹) using the *characteristics* `Adj` and `Ct1`, respectively.

Q# follows the QRAM model of computation [26], which assumes an unbounded supply of logical qubits from which the programmer can obtain a reference to a new qubit by calling the `use` command. Qubits are hence allocated and deallocated in a stack-like manner, where the lifetime of a qubit is equivalent to the lexical scope of the `use` command. Even though this stack discipline can ensure safe (quantum) memory management, it is currently not enforced by the Q# compiler and type system. Figure 1a shows a minimal example that passes the type checker but fails at runtime (in a simulator).

Programmers are allowed to create new bindings using `let` that refer to the same qubit as another binding, leading to *aliasing* of qubit references. While aliasing is ubiquitous in Q#, it can lead to unsafe behavior in violation of the *no-cloning theorem* [60], which forbids duplication of qubits. In Figure 1b, both `q1` and `q2` refer to the same qubit. Applying `CNOT` with `q1` as the control and `q2` as the target is equivalent to cloning the underlying qubit. Currently, Q# cannot prevent this issue statically.

An informal specification of the Q# language was recently published [44]. However, it does not capture the subtle aspects of the language, such as the aliasing of qubit references or its goal of maintaining a stack discipline. Our work makes these subtleties explicit and formal.

2.2 An Equational Theory for QRAM

Staton [54] presents a substructural (linear) version of his framework for “parameterized algebraic theories” [53]. He develops an axiomatization for quantum computation using this framework, which he shows to be fully complete. Staton then extracts an equational theory for a quantum programming language from his algebraic theory that uses *generic effects* rather than *algebraic operations* [43]. Finally, Staton remarks upon a variant of his theory [54, §6.2] that applies to the QRAM model, where instead of working with qubits, we work with references to qubits. This is the approach taken in projects like the Quantum IO Monad [2], Quantum Hoare Type Theory [50, 51], and, to our advantage, Q#.²

Here we reproduce Staton’s theory of a “quantum local store” [54, §6.2, p. 11] for reference; we will

¹Perhaps a better name for functors would be ‘combinators’ from the functional programming community to avoid confusion with other accepted meanings of the term ‘functor.’

²However, the stack-like management of qubits is unique to Q#.

see in §3.3 how this algebraic theory helps us describe the equational dynamics of $\lambda_{Q\#}$. We assume that the qubit references are unique, which we guarantee for $\lambda_{Q\#}$ in §3.2.1.

Generic Effects Staton adds the following generic effects to a standard linear type theory and obtains a quantum programming language [54, §5] similar to Selinger’s QPL [49].

$$\frac{}{\vdash \underline{\text{new}}() : \text{qubit}} \quad \frac{\Gamma \vdash t : \text{qubit}^{\otimes n}}{\Gamma \vdash \underline{\text{apply}}_U(t) : \text{qubit}^{\otimes n}} \quad \frac{\Gamma \vdash t : \text{qubit}}{\Gamma \vdash \underline{\text{measure}}(t) : \text{bool}}$$

Program Equations There are two interesting classes of axioms (ignoring the axioms that describe commutativity of let). For completeness, we also show axiom (C) pertaining to the *discard* operation (equivalent to measuring a qubit and ignoring its result). However, it does not apply in the QRAM model as noted by Staton [54, §6.2].

Axioms relating unitary gates and measurement:

$$\begin{aligned} (A) \quad & \underline{\text{measure}}(\underline{\text{apply}}_X(a)) \equiv \neg \underline{\text{measure}}(a) \\ (B) \quad & \text{let } (a', x') \text{ be } \underline{\text{apply}}_{D(U,V)}(a, x) \text{ in } (\underline{\text{measure}}(a'), x') \equiv \\ & \text{if } \underline{\text{measure}}(a) = 0 \text{ then } (0, \underline{\text{apply}}_U(x)) \text{ else } (1, \underline{\text{apply}}_V(x)) \\ (C) \quad & \underline{\text{discard}}(\underline{\text{apply}}_U(x)) \equiv \underline{\text{discard}}(x) \end{aligned}$$

Axioms relating allocation with unitaries and measurement:

$$\begin{aligned} (D) \quad & \underline{\text{measure}}(\underline{\text{new}}()) \equiv 0 \\ (E) \quad & \underline{\text{apply}}_{D(U,V)}(\underline{\text{new}}(), x) \equiv (\underline{\text{new}}(), \underline{\text{apply}}_U(x)) \end{aligned}$$

where $D(U, V) = U \oplus V = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix}$ applies U or V depending on the value of its first argument.

Axiom (A) says that applying the quantum X gate to a qubit and then measuring it is the same as negating the measurement result. Axiom (B) explains the action of a block diagonal matrix $D(U, V)$ as quantum control by stating that applying the diagonal matrix and then measuring the control qubit is equivalent to measuring the control qubit and branching on the result to decide whether to apply U or V . Axiom (C) says that if the qubits are to be discarded, applying a unitary is the same as doing nothing. Axiom (D) states that measuring a new qubit always results in 0, i.e., qubits are always initialized to 0. Axiom (E) says that using a new qubit as control is the same as controlling by 0.

We will show in §3.3 that our $\lambda_{Q\#}$ calculus follows similar program equations.

3 $\lambda_{Q\#}$: A Core Calculus for Q#

Our approach closely follows the type-theoretic interpretation of Standard ML, where Harper and Stone [20] developed a well-typed internal language for Standard ML, defined an elaboration relation between the external language and this internal language, and proved the properties of the metatheory of the language using the internal language. Harper and collaborators [7, 28] followed this work with the mechanization of the metatheory using the Twelf logical framework [42]. As a first step, we identify and isolate the core language, $\lambda_{Q\#}$, that captures the essential aspects of Q#. This core language is explicitly typed, and the safety properties of its type structure can be easily stated and proved.

Once we have identified the core, we define an elaboration relation from the surface Q# language to $\lambda_{Q\#}$ (§4). A Q# program is well-formed when it has a well-typed elaboration, and its semantics is defined to be that of its elaboration. The advantage of this approach is that proving properties about the metatheory of a large language becomes tractable because we only need to do it for the small, well-formed core.

To mirror the separation between operations and functions in Q#, we base the design of $\lambda_{Q\#}$ on Harper’s MA (Modernized Algol) [17], which maintains a separation between commands that modify state and expressions that do not. Q# is an ALGOL-like language in more ways than one—syntax, block-structure, local (classical) state, and safe integration of functional and imperative paradigms. However, unlike Reynolds’ Idealized ALGOL [48], the variables in Q# are immutable by default, and the language follows a call-by-value semantics, both of which make it closer to Harper’s MA.

Before presenting $\lambda_{Q\#}$, let us motivate our design choices and establish some terminology. Q# has two kinds of variable bindings. Those defined using the `let` keyword are the same as the variables in MA and follow the usual substitution-based semantics of functional programming languages. Those defined using the `mutable` keyword correspond to *assignables* that can be reassigned similar to “variables” in imperative languages. Significantly, they are restricted to the lexical scope in which they are bound. Since they do not affect equational reasoning across function boundaries, and our focus is on the quantum state, we ignore `mutable` variables in the rest of this paper. Qubits have type `Qubit` and syntactically look just like other variables but are references to underlying logical qubits that are never exposed. Unlike classical bindings, which follow value semantics, aliasing is permitted on qubits, leading to problems such as the violation of the no-cloning theorem discussed in §2.1. Qubits come into scope with either the `use` or `borrow` keywords. The former provides access to freshly allocated qubits in state $|0\rangle$, while the latter allows access to previously allocated (and potentially entangled) qubits. We do not consider borrowing in this work as it is an optimization concern that lets a programmer reuse ancillae in their code. The only allowed operations on qubits are gate application and measurement.

3.1 Syntax

Figure 2 presents the abstract syntax of $\lambda_{Q\#}$. We divide the grammar into a monadic effectful command language and a pure expression language (the simply-typed λ -calculus extended with encapsulated commands). We precisely specify the binding structure of the syntax following the notion of abstract binding trees from Harper’s PFPL [18]. Following the PFPL-syntactic conventions, `qref` $\langle q \rangle$, `gateap` $\langle U_{2^n} \rangle(e)$, and `diagap` $\langle U_{2^n}, V_{2^n} \rangle(e_1; e_2)$ are indexed by symbols [18, Ch. 31] (marked in color) and variadic product operators are indexed by finite sets, n , where we slightly abuse the notation, $n \triangleq \{1, 2, \dots, n\}$. We also use the notation, $\tau_n \triangleq i \mapsto \tau_i \mid i \in n$. Some operators take optional arguments marked by square brackets. We will often use the concrete syntax in blue color, and some standard derived forms from Harper’s language MA, shown in Appendix A, wherever there is no possibility of confusion.

The qubit reference type `qref` $\langle q \rangle$ is a singleton type [3, 21], which is equivalent to `ptr` (l) in alias types [52]. Qubit symbols are shown in orange to distinguish them from the usual variables denoted by the metavariable x ; we use qubit symbols to model the underlying logical qubit that the surface Q# language does not expose. Unitary operations, U (shown in pink), are parametric to the grammar; similar to Q#, which does not prefer a specific gate set. An n -qubit unitary is typed as $U : \times_{i \in n} (i \mapsto \text{qref}\langle q_i \rangle) \rightarrow \text{cmd}(\text{unit})$, where $\dim(U) = 2^n$. This type ensures that multi-qubit gates can be applied only to distinct qubits. The `apply` $_U(e)$ command applies the given unitary operation to a tuple of unique qubit references, where we follow singleton-tuple equivalence like Q# in case of a single-qubit unitary. Controlled unitaries can be represented using block diagonals, e.g., $\text{CNOT} \triangleq \mathbf{D}_{(I_2, X)}$ and are typed as $\mathbf{D}_{(U, V)} : \times_{i \in n+1} (i \mapsto \text{qref}\langle q_i \rangle) \rightarrow \text{cmd}(\text{unit})$, where $\dim(U) = \dim(V) = 2^n$. It is understood that

<i>Sort</i>	<i>Abstract</i>	<i>Concrete</i>	
Typ $\tau ::=$	$\text{qref}(q)$	$\text{qref}(q)$	qubit reference
	$\text{fun}(\tau_1; \tau_2)$	$\tau_1 \rightarrow \tau_2$	function
	$\text{cmd}(\tau)$	$\text{cmd}(\tau)$	command
	$\text{prod}(i \hookrightarrow \tau_i \mid i \in n)$	$\times_{i \in n}(i \hookrightarrow \tau_i)$	variadic product
	bool	bool	boolean
	unit	unit	unit
Exp $e ::=$	x	x	variable
	$\text{let}[\tau_1; \tau_2](e_1; x.e_2)$	$\text{let } x \text{ be } e_1 \text{ in } e_2$	let binding
	$\lambda[\tau_1; \tau_2](x.e)$	$\lambda(x.e)$	function
	$\text{ap}[\tau_1; \tau_2](e_1; e_2)$	$\text{ap}(e_1; e_2)$	application
	$\text{cmd}[\tau](m)$	$\text{cmd}(m)$	encapsulated command
	$\text{tuple}[\tau_n](i \hookrightarrow e_i \mid i \in n)$	$\langle i \hookrightarrow e_i \mid i \in n \rangle$	tuple
	$\text{proj}(i)[\tau_n](e)$	$e \cdot i$	projection
	true	true	true
	false	false	false
	$\text{if}[\tau](e; e_1; e_2)$	$\text{if } e \text{ then } e_1 \text{ else } e_2$	if expression
	unit	$\langle \rangle$	unit
Cmd $m ::=$	$\text{ret}[\tau](e)$	$\text{ret}(e)$	return
	$\text{bnd}[\tau_1; \tau_2](e; x.m)$	$\text{bnd } x \leftarrow e; m$	bind
	$\text{newqref}[\tau](x.m)$	$\text{new } x \text{ in } m$	new qubit reference
	$\text{gateap}(U_{2^n})(e)$	$\text{apply}_U(e)$	gate application
	$\text{diagap}(U_{2^n}, V_{2^n})(e_1; e_2)$	$\text{apply}_{D(U,V)}(e_1; e_2)$	diagonal gate application
	$\text{meas}(e)$	$\text{meas}(e)$	measure

Figure 2: Abstract and concrete syntax of $\lambda_{Q\#}$.

the number of arguments required for both forms of gate application depends on the dimension of the unitary parameters involved and is enforced by the typing rules.

3.2 Static Semantics

The pure fragment of $\lambda_{Q\#}$ is the usual simply-typed λ -calculus, so we will not say much about it here. We show typing rules for the effectful portion of $\lambda_{Q\#}$ in Figure 3.

All of our command typing judgments are parameterized by a signature, Σ , that keeps track of available qubit symbols in scope and corresponds to the shape of the quantum memory, much like store shapes³ in the semantics of ALGOL [35, 36, 48]. The intuition behind incorporating a signature is that the block structure induced by the allocation command changes the shape of the quantum memory under consideration by making a new qubit available to the program on entry and removing it on exit. This is the essence of the stack-like treatment of local state.⁴

Rules **CMD-RET** and **CMD-BND** are the two standard rules for monadic return and bind operations. The other four rules are specific to quantum computation.

The $\text{newqref}(x.m)$ command allocates a fresh logical qubit q and immediately makes a reference to it available in the scope of m . Its typing rule **CMD-NEWQREF** says that if command m returns a value of

³Store shapes follow laws similar to what are known as lenses in the current literature [10].

⁴Another view is to think of the commands as being parametrically polymorphic [9, 33, 34] to the store, an idea considered by Reynolds as early as 1975 [6, 47], even before store shapes. Still, we prefer the signature-based approach taken in Harper's **MA**.

$\Gamma \vdash_{\Sigma} m \dot{\sim} \tau$	$(m \text{ is a well-formed command relative to } \Sigma, \text{ returning a value of type } \tau)$	
$\frac{\text{CMD-RET}}{\Gamma \vdash_{\Sigma} \text{ret}(e) \dot{\sim} \tau}$	$\frac{\text{CMD-BND}}{\Gamma \vdash_{\Sigma} \text{bnd}(e; x.m) \dot{\sim} \tau'}$	$\frac{\text{CMD-NEWQREF}}{\Gamma \vdash_{\Sigma} \text{newqref}(x.m) \dot{\sim} \tau}$
$\frac{\text{CMD-GATEAPREF}}{\Gamma \vdash_{\Sigma} \text{gateap}(U_{2^n})(e) \dot{\sim} \text{unit}}$	$\frac{\text{CMD-DIAGAPREF}}{\Gamma \vdash_{\Sigma} \text{diagap}(U_{2^n}, V_{2^n})(e_1; e_2) \dot{\sim} \text{unit}}$	$\frac{\text{CMD-MEASREF}}{\Gamma \vdash_{\Sigma} \text{meas}(e) \dot{\sim} \text{bool}}$

Figure 3: Typing of commands. Γ is the standard typing context, and the signature, Σ , keeps track of qubit symbols in scope. Each qubit symbol is required to be distinct. See other rules in [Appendix B.1](#).

type τ in a context containing $x : \text{qref}(q)$ and a signature extended with q , then $\text{newqref}(x.m)$ returns a value of type τ . The binding structure ensures that the lifetime of the newly allocated qubit is equal to its lexical scope, ensuring a strict stack discipline and providing safe and automatic management of qubits.

Rules [CMD-GATEAPREF](#) and [CMD-DIAGAPREF](#) for unitary operations enforce the constraint that the input qubit references are distinct. Rule [CMD-MEASREF](#) shows how to obtain a boolean value from an expression that resolves to a qubit reference.

In summary, the allocation command changes the shape of the store (quantum state under consideration), while commands like unitary application and measurement change the store (quantum state).

3.2.1 Safety Properties

We claim that our type system supports two safety properties currently not offered by Q#:

Proposition 1. $\lambda_{Q\#}$ supports controlled aliasing and hence, statically enforces the no-cloning theorem for all unitary operations.

This follows from rules [CMD-GATEAPREF](#) and [CMD-DIAGAPREF](#): The premises of both typing rules require the input qubit references to be unique as all the entries in a tuple are required to be references to different logical qubits. In the case of the block diagonal, the control qubit reference, e_1 , is also required to be distinct from the qubit references in e_2 .

Example 3.1. The unsafe code fragment from [Figure 1b](#) can be written in $\lambda_{Q\#}$ syntax as:

$$\text{newqref}(q_1 . \text{ret}(\text{let}(q_1; q_2 . \text{cmd}(\text{diagap}(I_2, X_2)(q_1; q_2))))))$$

or in the concrete syntax as `new q1 in ret(let q2 be q1 in cmd(applyD(I2, X)(q1; q2)))`. Since the type of the qubit reference in $\lambda_{Q\#}$, $\text{qref}(q)$, is indexed by the symbolic name of the qubit, we can tell statically that q_1 and q_2 reference the same underlying logical qubit. This allows our type system to reject the above program even though the Q# compiler allows it to pass.

Proposition 2. $\lambda_{Q\#}$ statically ensures safe memory management and disallows dangling qubit references.

The allocation command, $\text{newqref}(x.m)$, as previously explained, comes with its own binding form, which ensures that the reference created during allocation can never escape its lexical scope. In rule [CMD-NEWQREF](#), the fresh logical qubit q allocated during this command is only available in the extended signature in the premise and not in the conclusion at the end of the command.

Example 3.2. Using $\lambda_{Q\#}$ concrete syntax and the derived forms from [Appendix A](#), the unsafe code fragment shown in [Figure 1a](#) can be written as `let NewQubit be proc() { new x in ret(x) } in $\langle \rangle$` . Here, while `ret(x)` has type $\text{qref}\langle q \rangle$ when q is in the signature, i.e., $\Gamma, x : \text{qref}\langle q \rangle \vdash_{\Sigma, q} \text{ret}(x) \dot{\sim} \text{qref}\langle q \rangle$; the conclusion of rule **CMD-NEWQREF** removes q from scope and renders `new x in ret(x)` ill-typed, i.e., $\Gamma \not\vdash_{\Sigma} \text{new } x \text{ in ret}(x) \dot{\sim} \text{qref}\langle q \rangle$.

3.3 Dynamic Semantics

As Staton [54, p. 3] suggests, “by giving a fully complete equational theory we can understand quantum computation from the axioms of the theory without having to turn to denotational models built from operator algebra”; we rely on his equational theory for quantum local store [54, §6.2] to provide an equational dynamics for the effectful quantum fragment of $\lambda_{Q\#}$.⁵ Unlike Staton’s language, our unitary operations do not return qubits but modify them in place. In this presentation, we use several derived forms from [Appendix A](#). Specifically, `do` returns the result of sequential execution of commands. The program equations assume the availability of a universal gate set.

Interesting Axioms

$$a : \text{qref}\langle q \rangle \vdash \text{do} \{ \text{apply}_x(a); \text{meas}(a) \} \equiv \neg \text{do} \{ \text{meas}(a) \} \quad (\text{A})$$

$$a : \text{qref}\langle q \rangle, b : \prod_{i \in n} (i \hookrightarrow \text{qref}\langle r_i \rangle) \vdash \{ \text{apply}_{D(U,V)}(a; b); \text{meas}(a); \text{ret}(\langle \rangle) \} \equiv \\ \{ x \leftarrow \text{meas}(a); \text{ret}(\text{if } x \text{ then cmd}(\text{apply}_V(b)) \text{ else cmd}(\text{apply}_U(b))) \} \quad (\text{B})$$

$$\cdot \vdash \text{do} \{ \text{new } a \text{ in meas}(a) \} \equiv \text{false} \quad (\text{D})$$

$$b : \text{qref}\langle q \rangle \vdash \text{do} \{ \text{new } a \text{ in apply}_{D(U,V)}(a; b) \} \equiv \text{do} \{ \text{apply}_U(b); \text{new } a \text{ in ret}(\langle \rangle) \} \quad (\text{E})$$

As mentioned in [§2.2](#), we omit Staton’s axiom (C) because, in the QRAM model, the discard operation just forgets the name of the reference to a qubit. In $Q\#$ and $\lambda_{Q\#}$, we may consider an equivalent behavior: qubit references are automatically forgotten when they reach the end of their lexical scope. A degenerate case of axiom (C) (for a 1×1 unitary) holds for both Staton’s theory for a quantum local store and $\lambda_{Q\#}$; it says that one can ignore the global phase.

Administrative Axioms The following equations correspond to respecting the composition and product monoidal structure of unitaries:

$$m_1 : \text{cmd}(\tau_1), m_2 : \text{cmd}(\tau_2) \vdash \text{do} \{ \text{new } a \text{ in new } b \text{ in } m_1; \text{apply}_{\text{SWAP}}(a, b); m_2 \} \equiv \\ \text{do} \{ \text{new } a \text{ in new } b \text{ in } m_1; \text{let } \langle b, a \rangle \text{ be } \langle a, b \rangle \text{ in cmd}(m_2) \} \quad (\text{F})$$

$$e : \prod_{i \in n} (i \hookrightarrow \text{qref}\langle q_i \rangle) \vdash \text{do} \{ \text{apply}_{I_{2^n}}(e) \} \equiv \langle \rangle \quad (\text{G})$$

$$e : \prod_{i \in n} (i \hookrightarrow \text{qref}\langle q_i \rangle) \vdash \text{do} \{ \text{apply}_{VU}(e) \} \equiv \text{do} \{ \text{apply}_U(e); \text{apply}_V(e) \} \quad (\text{H})$$

$$e_1 : \prod_{i \in m} (i \hookrightarrow \text{qref}\langle q_i \rangle), \\ e_2 : \prod_{i \in n} (i \hookrightarrow \text{qref}\langle r_i \rangle) \vdash \text{do} \{ \text{apply}_{U \otimes V}(e_1, e_2) \} \equiv \text{do} \{ \text{apply}_U(e_1); \text{apply}_V(e_2) \} \quad (\text{I})$$

Selinger [49] notes that the *SWAP* gate is equivalent to classically renaming qubit references, which captures the intuition behind equation (F). However, in our case, we have to ensure that the scope of the

⁵We show the traditional operational semantics in [Appendix B.2](#).

qubits is limited to our expression (since *SWAP* is stateful). Axiom (G) says that applying an identity gate is equivalent to doing nothing. Axioms (H) and (I) show the two ways of composing unitaries—sequential and tensor products (horizontal and vertical composition, respectively, in circuit notation).

Like Staton, we also state the commutativity equations that hold for $\lambda_{Q\#}$:

$$a : \text{qref}\langle q \rangle, b : \text{qref}\langle r \rangle, m : \text{cmd}(\tau) \vdash \text{do} \{x \leftarrow \text{meas}(a) ; y \leftarrow \text{meas}(b) ; m\} \equiv \text{do} \{y \leftarrow \text{meas}(b) ; x \leftarrow \text{meas}(a) ; m\} \quad (\text{J})$$

$$m : \text{cmd}(\tau) \vdash \text{do} \{\text{new } a \text{ in new } b \text{ in } m\} \equiv \text{do} \{\text{new } b \text{ in new } a \text{ in } m\} \quad (\text{K})$$

$$b : \text{qref}\langle q \rangle, m : \text{cmd}(\tau) \vdash \text{do} \{\text{new } a \text{ in } y \leftarrow \text{meas}(b) ; m\} \equiv \text{do} \{y \leftarrow \text{meas}(b) ; \text{new } a \text{ in } m\} \quad (\text{L})$$

Now that we have shown that the quantum portion of $\lambda_{Q\#}$ is equivalent to Staton’s quantum programming language [54, §5]⁶ and corresponding program equations with his theory of quantum local store, we can restate Staton’s result [54, p. 8, Theorem 11] for our language:

Theorem 1 (Universality of $\lambda_{Q\#}$). *For any linear map $f : M_{2^{n_1}} \oplus \cdots \oplus M_{2^{n_k}} \rightarrow M_{2^p}$ that is completely positive and unital (i.e. corresponds to a trace-preserving superoperator), there is a $\lambda_{Q\#}$ term, t , such that t implements f .*

This corresponds to Staton’s Theorem 11.1 [54], which is a variation on Selinger’s Theorem 6.14 [49]. The proof relies on the correspondence between Staton’s simple quantum language and a fragment of $\lambda_{Q\#}$, where the translation is straightforward (Appendix C).

Theorem 2 (Completeness). *Assuming an axiomatization of unitaries, if two terms t and u have equivalent interpretation in a common context, Γ , then $\Gamma \vdash t \equiv u$ is derivable.*

Note that since Q# is parameterized over gate sets, we need an equational theory over unitaries for $\lambda_{Q\#}$. In the simplest case, we can declare two unitaries equal if their corresponding matrices are equal. Again, the result follows from Staton’s Theorem 11.2. There are two differences: (1) instead of algebraic operations, our theorem is stated in terms of generic effects (which correspond directly to programming); (2) in addition to the equations (A)–(L) stated above, we also need the standard $\beta\eta$ -equalities of simply-typed λ -calculus, which are required because our axioms do not live in isolation but are written as typed expressions in a context.

4 Translation from Q# to $\lambda_{Q\#}$

We summarize the rules for converting from the supported features of Q# to $\lambda_{Q\#}$ in Table 1. For ease of presentation, we use the derived forms from Appendix A. Figure 4 in Appendix D shows the elaboration of the Q# teleport example from Listing 1.

Elaboration maintains a *context* (not shown in Table 1) that stores the logical qubit associated with each qubit reference. To translate the Q# type `Qubit` to the $\lambda_{Q\#}$ type `qref` $\langle q \rangle$, we look up the reference associated with the `Qubit` type in the context or add a new logical qubit to the context. The `use` statements and `operation` parameters update the context to include a mapping from the new qubit or qubit parameter(s) to a fresh logical qubit. In Figure 4, a , b , and m are distinct logical qubits introduced by elaboration.

Elaboration performs some type checking to produce well-formed $\lambda_{Q\#}$ terms. For example, elaboration checks that the first argument to a `Controlled` or `Adjoint` functor is equipped with the `Ctl` and/or `Adj`

⁶See Appendix C for the trivial term translation.

Q# Syntax	$\lambda_{Q\#}$ Translation
$\llbracket (\tau_1, \dots, \tau_n) \rrbracket$	$\times_{i \in n} (i \leftrightarrow \llbracket \tau_i \rrbracket)$
$\llbracket \tau_1 \rightarrow \tau_2 \rrbracket$	$\llbracket \tau_1 \rrbracket \rightarrow \llbracket \tau_2 \rrbracket$
$\llbracket \tau_1 \Rightarrow \tau_2 \rrbracket$	$\llbracket \tau_1 \rrbracket \Rightarrow \llbracket \tau_2 \rrbracket$
$\llbracket \text{Bool} \rrbracket$ and $\llbracket \text{Result} \rrbracket$	<code>bool</code>
$\llbracket \text{Qubit} \rrbracket$	<code>qref</code> $\langle q \rangle$
$\llbracket \text{Unit} \rrbracket$	<code>unit</code>
$\llbracket \text{function } f(x_1 : \tau_1, \dots) : \tau \{e\} \rrbracket$	$\lambda [\times_{i \in n} (i \leftrightarrow \llbracket \tau_i \rrbracket); \tau] (\langle i \leftrightarrow x_i \mid i \in n \rangle . \llbracket e \rrbracket)$
$\llbracket \text{operation } f(x_1 : \tau_1, \dots) : \tau \{s\} \rrbracket$	$\lambda [\times_{i \in n} (i \leftrightarrow \llbracket \tau_i \rrbracket); \text{cmd}(\tau)] (\langle i \leftrightarrow x_i \mid i \in n \rangle . \llbracket s \rrbracket)$
$\llbracket \text{return } e \rrbracket$	<code>ret</code> $(\llbracket e \rrbracket)$
$\llbracket \text{let } x = e; \dots \rrbracket$	<code>let</code> x <code>be</code> $\llbracket e \rrbracket$ <code>in</code> $\llbracket \dots \rrbracket$
$\llbracket \text{if } e \{s_1\} \text{ else } \{s_2\} \rrbracket$	<code>if</code> $\llbracket e \rrbracket$ <code>then</code> <code>cmd</code> $(\llbracket s_1 \rrbracket)$ <code>else</code> <code>cmd</code> $(\llbracket s_2 \rrbracket)$
$\llbracket \text{use } q = \text{Qubit} \{s\} \rrbracket$	<code>new</code> q <code>in</code> $\llbracket s \rrbracket$
$\llbracket \text{Adjoint } e_1(e_2) \rrbracket$	<code>apply</code> $_{U^\dagger}(\llbracket e_2 \rrbracket)$, where $U = \text{mat}(\llbracket e_1 \rrbracket)$
$\llbracket \text{Controlled } e_1(q, e_2) \rrbracket$	<code>apply</code> $_{D(i_2^n, U)}(q; \llbracket e_2 \rrbracket)$, where $U = \text{mat}(\llbracket e_1 \rrbracket)$
$\llbracket e_1(e_2) \rrbracket$	$\llbracket e_1 \rrbracket(\llbracket e_2 \rrbracket)$
$\llbracket (e_1, \dots, e_n) \rrbracket$	$\langle \llbracket e_i \rrbracket, \dots, \llbracket e_n \rrbracket \rangle$
$\llbracket \text{true} \rrbracket$ and $\llbracket \text{One} \rrbracket$	<code>true</code>
$\llbracket \text{false} \rrbracket$ and $\llbracket \text{Zero} \rrbracket$	<code>false</code>

Table 1: Select Q# to $\lambda_{Q\#}$ elaboration rules. f , x , and q are variable names, e is a Q# expression, s is a Q# statement, and τ is a Q# type. $\llbracket \cdot \rrbracket$ is the elaboration function and $\text{mat}(\cdot)$ converts a $\lambda_{Q\#}$ expression to its corresponding unitary operator. In the rule for `Qubit`, q is determined from the elaboration context.

characteristics and inlines the corresponding operation specialization, converting it to a unitary operator. We need to do this during elaboration because we do not yet encode characteristic information or specializations in $\lambda_{Q\#}$. An adjointable Q# operation with type $(\text{Qubit}, \dots, \text{Qubit}) \Rightarrow \text{Unit}$ can be converted into a unitary matrix by composing the unitary representations of its primitive gates. Note that the type signature and the fact that the operation is adjointable (i.e., no measurement) mean it can be unfolded to a sequence of primitive gates. We also expand multi-controlled operations (`Controlled` statements with a list of controls) into a nested group of single-qubit controlled operations, and expand `if-elif-else` expressions into nested `if-then-else` expressions using $\langle \rangle$ in place of an empty `else` block. Finally, we restrict Q# `function` bodies to be pure expressions since we do not handle classical `mutable` values.

As this is our initial attempt to get the foundations right, we do not yet support several Q# features: namespaces; operation characteristics; custom operation specializations (i.e., implementations of controlled or adjoint variants); general application of the `Adjoint` and `Controlled` functors; arrays and slices; type parameters; base types outside of `Bool`, `Result`, and `Unit`; iteration using `for`, `while`, or `repeat`; and `within-apply` blocks (which apply an operation and its adjoint). We say more in §6 about the challenges involved in supporting some of these features.

5 Related Work

Large Language Definition Efforts In starting this project, we were encouraged by previous efforts in the formal specification of large programming languages such as Standard ML [20, 28], Java [23],

JavaScript [16], Rust [24, 25], and, most recently, Go [14]. As we mentioned in §3, we more or less followed the pioneering methodology of the formalization and mechanization of the definition of Standard ML [29] by identifying a well-founded core language and performing all metatheoretical reasoning on that core. These projects demonstrate the extent to which it is possible to distill large and complex languages into their formal and faithful essence. Java and Go serve as examples of industry-scale languages in mass use benefiting from formalization and academic study: Extensions such as generics (polymorphism) were first investigated on smaller cores of the respective languages before being adopted in production over the years. In the case of JavaScript, perhaps the impact of a careful formal study was even more significant as JavaScript is the de-facto programming language of the web. We hope our work serves as a similar playground for extensions and future impact.

Equational Theories Like Staton, we do not focus on the axiomatization of unitaries but of quantum computation in general. We discuss two similar works here.

Paykin and Zdancewic [40] build upon Staton’s work and present an equational theory for quantum computation embedded inside homotopy type theory (HoTT) [57]. The essential idea to treat unitaries as higher inductive paths simplifies the presentation of the equational theory as several axioms can be derived using the rich structure of HoTT. While their work focuses on embedding a quantum language inside a highly expressive dependent type theory, we are motivated by practical concerns in defining semantics for a real-world quantum language.

Peng et al. [41] introduce Non-Idempotent Kleene Algebra (NKAT) to reason about programs algebraically. Their language is based on Kozen’s Kleene Algebra with Test (or KAT), which models both programs and assertions, allowing for a lightweight implementation of a Hoare-style logic [27]. While the underlying language of regular expressions is not designed for convenience in programming, their use of NKAT to verify quantum program transformations is a key use case of equational theories and one we plan to explore in the future.

Linearity and Monadic Quantum Languages Research-oriented languages like QWIRE [39] and Silq [5] employ a linear type system to enforce the no-cloning theorem. So far, industry languages, including Q#, have not adopted linear typing. The lack of linear typing in Q# is justified by its monadic treatment of state. That is, the monad interface imposes a sequential order to manipulate the quantum state as every monad can be treated as a linear-use state monad [30]. The design decision in Q# to permit uncontrolled aliasing of qubits for user comfort is the only reason a monadic interface is not enough, which is addressed by our type system.

Other monadic languages include Quantum IO Monad [2] (QIO) and Quantum Hoare Type Theory [50, 51] (QHTT). QIO is a pioneering monadic interface that isolates quantum effects inside a monad as we do in $\lambda_{Q\#}$. QHTT is a typed framework that extends the QIO monad with pre- and postconditions so that precise specifications about the quantum state can be stated and proved in a dependent type theory.

ALGOL-like Quantum Languages The IQu language [38] extends Idealized ALGOL with quantum circuits and quantum variables, much as we extend Harper’s Modernized Algol (MA). Like $\lambda_{Q\#}$, IQu uses references to access qubits and therefore does not need a linear type system to prevent cloning. Though every newly allocated qubit is unique, IQu does not have a way to guarantee that multiple references to the same qubit are not passed to a single operation. Instead, IQu’s use of Idealized ALGOL is focused on programmability, following a design philosophy similar to that of Q#. IQu allows programmers to write the classical parts of their programs in a familiar way while providing access to the quantum state.

6 Conclusion and Perspectives

We present a core calculus for the Q# programming language, dubbed $\lambda_{Q\#}$. We maintain a separation between the quantum effectful and the pure expression sub-languages, expose the monadic nature of computation inherent in Q#, make qubit aliasing and block structure explicit, and present an equational semantics for $\lambda_{Q\#}$, building upon Staton’s fully complete equational theory for quantum computation.

A formal specification of the whole Q# language still requires more work. Some extensions are straightforward; e.g., classical mutable bindings in Q# can be modeled after assignables in Harper’s MA; conveniently, they follow the model of classical local store analogous to how we modeled the quantum local store in this paper. Here it helps that Q# does not allow references to any types other than qubits. Other features are more challenging, including arrays, slices, iteration, polymorphism, and patterns like `within-apply` and `repeat-until-success` [37]. Then there is the question of how to treat operations that have specializations supplied by the programmer versus those auto-generated by the Q# compiler (which is not known statically); we may need to consider a phase distinction [19] here to distinguish between what can be derived statically using types and what requires inspecting the code.

We plan to gain confidence in our formalization by mechanizing its metatheory. We see potential in recent developments such as the Agda-based formalization of Second-Order Abstract Syntax [8], which lets users concisely specify algebraic theories such as Staton’s and significantly reduces the boilerplate code required to state interesting theorems about the theory. However, this tool does not support substructural assumptions on qubit symbols, making our proposed extension a nontrivial prospect.

A major goal of this project is to form a playground for prototyping extensions to the Q# type system. For instance, a peculiar decision in Q# is to allow uncontrolled aliasing of qubits to support user-friendly features such as qubit arrays. While convenient, reasoning about interference freedom for arrays is notoriously hard; specifically, our approach to enforce no-cloning inspired by alias types [52, 59] does not easily scale to arrays [58, §3.5.1]. We are extending our $\lambda_{Q\#}$ type checker with a constraint solver to evaluate potential solutions for scenarios that occur in practice in Q# library code. Depending on the complexity of the array indexing used in practice, we may use a natural number inequality checker, a simple symbolic numerical solver, or a full-fledged SMT solver like Z3 [31] to guarantee qubit distinctness.

We could also statically check Q#’s `Adj` and `Ct1` characteristics for validity. In the simplest case, we would flag operations as unitary or non-unitary in order to inform the compiler when adjoints and controls can be trivially synthesized, in the manner of Silq [5]. However, more complex programs are adjointable and controllable in practice, which may require a more sophisticated approach.

One of our insights from this project is that even though quantum computation is a fundamentally new abstraction, many classical techniques from programming languages and compilers communities can be adapted to the quantum setting [55]. Q# and its Quantum Development Kit (QDK) are significant examples of realizing that vision [1]. As a high-level programming language, Q# must also compile to efficient, low-level machine instructions. Recently, Microsoft announced QIR, a Quantum Intermediate Representation based on the popular LLVM framework [13], which has gained significant industry backing in the form of the QIR Alliance [45]. This provides an exciting avenue for future development. We plan to explore semantics-preserving compilation from Q# to QIR using our formalization. This project will require formally specifying the semantics of QIR, for which we will draw upon the Verified LLVM (Vellvm) project [61, 62]. We also aim to formalize QIR’s *profiles*, which specify what kinds of quantum operations are allowed on a given quantum architecture. This, along with our current work, will constitute a significant step toward our broader vision of a fully verified quantum stack [46].

Acknowledgments

We thank Bettina Heim and Alan Geller for helping us get this project off the ground, Bob Harper for several fruitful conversations, Mike Hicks for pointing us to Alias Types during a preliminary presentation at PPlanQC 2021, and Ohad Kammar for making several valuable connections. Jennifer Paykin, Sam Staton, Bob Harper, and anonymous reviewers from the PC of FSCD 2022 gave critical feedback on a previous draft. For presenting $\lambda_{Q\#}$ syntax, Bob Harper’s PFPL syntax macros⁷ were very helpful. We also thank Matt Amy, Adrian Lehmann, and the anonymous reviewers of QPL 2022 for their feedback on the manuscript.

Funding This material is supported by EPiQC, an NSF Expedition in Computing, under Grant No. CCF-1730449, the Air Force Office of Scientific Research under Grant No. FA95502110051 and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Testbed Pathfinder Program under Award Number DE-SC0019040.

References

- [1] Alfred Aho and Jeffrey Ullman. “Abstractions, Their Algorithms, and Their Compilers”. In: *Commun. ACM* 65.2 (Jan. 2022), pp. 76–91. DOI: [10.1145/3490685](https://doi.org/10.1145/3490685).
- [2] Thorsten Altenkirch and Alexander S. Green. “The Quantum IO Monad”. In: *Semantic Techniques in Quantum Computation*. Ed. by Simon J. Gay and Ian Mackie. Cambridge, UK: Cambridge University Press, 2009, pp. 173–205. DOI: [10.1017/CB09781139193313.006](https://doi.org/10.1017/CB09781139193313.006). URL: <https://www.cs.nott.ac.uk/~psztxa/g5xnsc/chapter.pdf>.
- [3] David Aspinall. “Subtyping with Singleton Types”. In: *Computer Science Logic*. Vol. 933. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1995, pp. 1–15. DOI: [10.1007/BFb0022243](https://doi.org/10.1007/BFb0022243). URL: <https://homepages.inf.ed.ac.uk/da/papers/lss/>.
- [4] John Azariah. *F# & Q# - A tale of two languages*. F# and Q# Advent Calendars 2018. Dec. 2018. URL: <https://johnazariah.github.io/2018/12/04/tale-of-two-languages.html>.
- [5] Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. “Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics”. In: *Proc. PLDI ’20*. New York, NY: ACM, 2020, pp. 286–300. DOI: [10.1145/3385412.3386007](https://doi.org/10.1145/3385412.3386007). URL: <https://files.sri.inf.ethz.ch/website/papers/pldi20-silq.pdf>.
- [6] Stephen Brookes, Peter W. O’Hearn, and Uday Reddy. “The Essence of Reynolds”. In: *Proc. POPL ’14*. New York, NY: ACM, 2014, pp. 251–255. DOI: [10.1145/2535838.2537851](https://doi.org/10.1145/2535838.2537851).
- [7] Karl Crary and Robert Harper. *Mechanized Definition of Standard ML*. GitHub. The Standard ML Language Family, 2009. URL: <https://github.com/SMLFamily/The-Mechanization-of-Standard-ML>.
- [8] Marcelo Fiore and Dmitriy Szamozvancev. “Formal Metatheory of Second-Order Abstract Syntax”. In: *Proc. ACM Program. Lang.* 6.POPL, 53 (2022). Source: <https://github.com/DimaSamoz/agda-soas>. DOI: [10.1145/3498715](https://doi.org/10.1145/3498715).
- [9] Matthew Fluet and Greg Morrisett. “Monadic regions”. In: *J. Funct. Program.* 16.4-5 (2006), pp. 485–545. DOI: [10.1017/S095679680600596X](https://doi.org/10.1017/S095679680600596X).

⁷Available on GitHub at <https://github.com/RobertHarper/pfpl-syntax>.

- [10] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. “Combinators for Bidirectional Tree Transformations: A Linguistic Approach to the View-Update Problem”. In: *ACM Trans. Program. Lang. Syst.* 29.3 (2007), 17–es. DOI: [10.1145/1232420.1232424](https://doi.org/10.1145/1232420.1232424).
- [11] Alan Geller. *Qubits in Q#*. Q# Advent Calendar 2018. Dec. 2018. URL: <https://devblogs.microsoft.com/qsharp/qubits-in-qsharp/>.
- [12] Alan Geller. *What are Qubits?* Q# Advent Calendar 2019. Dec. 2019. URL: <https://devblogs.microsoft.com/qsharp/what-are-qubits/>.
- [13] Alan Geller. *Introducing Quantum Intermediate Representation (QIR)*. Q# Blog. Sept. 2020. URL: <https://devblogs.microsoft.com/qsharp/introducing-quantum-intermediate-representation-qir/>.
- [14] Robert Griesemer, Raymond Hu, Wen Kokke, Julien Lange, Ian Lance Taylor, Bernardo Toninho, Philip Wadler, and Nobuko Yoshida. “Featherweight Go”. In: *Proc. ACM Program. Lang.* 4.OOPSLA, 149 (2020). DOI: [10.1145/3428217](https://doi.org/10.1145/3428217). arXiv: [2005.11710](https://arxiv.org/abs/2005.11710).
- [15] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. New York, NY: ACM, 1996, pp. 212–219. DOI: [10.1145/237814.237866](https://doi.org/10.1145/237814.237866). arXiv: [quant-ph/9605043](https://arxiv.org/abs/quant-ph/9605043).
- [16] Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi. “The Essence of JavaScript”. In: *ECOOP 2010 – Object-Oriented Programming*. Ed. by Theo D’Hondt. Berlin, Heidelberg: Springer, 2010, pp. 126–150. DOI: [10.1007/978-3-642-14107-2_7](https://doi.org/10.1007/978-3-642-14107-2_7). arXiv: [1510.00925](https://arxiv.org/abs/1510.00925).
- [17] Robert Harper. “Modernized Algol”. In: *Practical Foundations for Programming Languages*. 2nd ed. Cambridge, UK: Cambridge University Press, 2016. Chap. 34, pp. 301–312. DOI: [10.1017/CB09781316576892.036](https://doi.org/10.1017/CB09781316576892.036).
- [18] Robert Harper. *Practical Foundations for Programming Languages*. 2nd ed. Abbreviated online edition, with corrections: <https://www.cs.cmu.edu/~rwh/pfpl/2nded.pdf>. Cambridge, UK: Cambridge University Press, 2016. DOI: [10.1017/CB09781316576892](https://doi.org/10.1017/CB09781316576892).
- [19] Robert Harper. *Phase Distinctions in Type Theory*. Talk at The Topos Institute Colloquium. Dec. 2021. URL: <https://www.cs.cmu.edu/~rwh/talks/ti-talk.pdf>.
- [20] Robert Harper and Christopher A. Stone. “A Type-Theoretic Interpretation of Standard ML”. In: *Proof, Language, and Interaction: Essays in Honor of Robin Milner*. Cambridge, MA: MIT Press, 2000, pp. 341–387. DOI: [10.7551/mitpress/5641.003.0019](https://doi.org/10.7551/mitpress/5641.003.0019). URL: <https://www.cs.cmu.edu/~rwh/papers/ttismml/ttismml.pdf>.
- [21] Susumu Hayashi. “Singleton, Union, and Intersection Types for Program Extraction”. In: *Information and Computation* 109.1–2 (1994), pp. 174–210. DOI: [10.1006/inco.1994.1016](https://doi.org/10.1006/inco.1994.1016).
- [22] Bettina Heim, Mathias Soeken, Sarah Marshall, Christopher E. Granade, Martin Roetteler, Alan Geller, Matthias Troyer, and Krysta M. Svore. “Quantum Programming Languages”. In: *Nature Reviews Physics* 2.12 (2020), pp. 709–722. DOI: [10.1038/s42254-020-00245-7](https://doi.org/10.1038/s42254-020-00245-7).
- [23] Atsushi Igarashi, Benjamin C. Pierce, and Philip Wadler. “Featherweight Java: A Minimal Core Calculus for Java and GJ”. In: *ACM Trans. Program. Lang. Syst.* 23.3 (2001), pp. 396–450. DOI: [10.1145/503502.503505](https://doi.org/10.1145/503502.503505).
- [24] Ralf Jung. “Understanding and Evolving the Rust Programming Language”. PhD thesis. Saarland University, 2020. DOI: [10.22028/D291-31946](https://doi.org/10.22028/D291-31946). URL: <https://www.ralfj.de/research/thesis.html>.

- [25] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. “RustBelt: Securing the Foundations of the Rust Programming Language”. In: *Proc. ACM Program. Lang.* 2.POPL, 66 (2017). DOI: [10.1145/3158154](https://doi.org/10.1145/3158154).
- [26] Emanuel Knill. *Conventions for Quantum Pseudocode*. Tech. rep. LA-UR-96-2724. Los Alamos National Laboratory, June 1996. DOI: [10.2172/366453](https://doi.org/10.2172/366453).
- [27] Dexter Kozen. “Kleene Algebra with Tests”. In: *ACM Trans. Program. Lang. Syst.* 19.3 (1997), pp. 427–443. DOI: [10.1145/256167.256195](https://doi.org/10.1145/256167.256195).
- [28] Daniel K. Lee, Karl Cray, and Robert Harper. “Towards a Mechanized Metatheory of Standard ML”. In: *Proc. POPL '07*. New York, NY: ACM, 2007, pp. 173–184. DOI: [10.1145/1190216.1190245](https://doi.org/10.1145/1190216.1190245). URL: <https://www.cs.cmu.edu/~dklee/papers/tslf-popl.pdf>.
- [29] Robin Milner, Robert Harper, David MacQueen, and Mads Tofte. *The Definition of Standard ML*. Revised Edition. Cambridge, MA: MIT Press, 1997. DOI: [10.7551/mitpress/2319.001.0001](https://doi.org/10.7551/mitpress/2319.001.0001). URL: <https://smlfamily.github.io/sml97-defn.pdf>.
- [30] Rasmus Ejlers Møgelberg and Sam Staton. “Linear Usage of State”. In: *Log. Methods Comput. Sci.* 10.1, 17 (2014). DOI: [10.2168/LMCS-10\(1:17\)2014](https://doi.org/10.2168/LMCS-10(1:17)2014).
- [31] Leonardo de Moura and Nikolaj Bjørner. “Z3: An Efficient SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '08)*. Berlin, Heidelberg: Springer, 2008, pp. 337–340. DOI: [10.1007/978-3-540-78800-3_24](https://doi.org/10.1007/978-3-540-78800-3_24).
- [32] Mariia Mykhailova. “The Quantum Katas: Learning Quantum Computing Using Programming Exercises”. In: *Proc. SIGCSE 2020*. Source: <https://github.com/microsoft/QuantumKatas>. New York, NY: ACM, 2020, p. 1417. DOI: [10.1145/3328778.3372543](https://doi.org/10.1145/3328778.3372543).
- [33] Peter W. O’Hearn and John C. Reynolds. “From Algol to Polymorphic Linear Lambda-Calculus”. In: *J. ACM* 47.1 (2000), pp. 167–223. DOI: [10.1145/331605.331611](https://doi.org/10.1145/331605.331611).
- [34] Peter W. O’Hearn and Robert D. Tennent. “Parametricity and Local Variables”. In: *ALGOL-like Languages*. Ed. by Peter W. O’Hearn and Robert D. Tennent. Vol. 2. Boston, MA: Birkhäuser, 1997, pp. 109–163. DOI: [10.1007/978-1-4757-3851-3_6](https://doi.org/10.1007/978-1-4757-3851-3_6). Repr. of “Parametricity and Local Variables”. In: *J. ACM* 42.3 (1995), pp. 658–709. DOI: [10.1145/210346.210425](https://doi.org/10.1145/210346.210425).
- [35] Frank J. Oles. “A Category-Theoretic Approach to the Semantics of Programming Languages”. Order no. AAI8301650. PhD thesis. Syracuse University, 1982. URL: <https://www.cs.cmu.edu/afs/cs.cmu.edu/project/fox-19/member/jcr/www/FrankOlesThesis.pdf>.
- [36] Frank J. Oles. “Type Algebras, Functor Categories and Block Structure”. In: *Algebraic Methods in Semantics*. Ed. by Maurice Nivat and John C Reynolds. Cambridge University Press, 1985, pp. 543–573. DOI: [10.7146/dpb.v12i156.7430](https://doi.org/10.7146/dpb.v12i156.7430).
- [37] Adam Paetznick and Krysta M. Svore. “Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries”. In: *Quantum Inf. Comput.* 14.15 & 16 (2014). Corresponding Q# code sample: <https://docs.microsoft.com/en-us/samples/microsoft/quantum/repeat-until-success/>, pp. 1277–1301. DOI: [10.26421/qic14.15-16-2](https://doi.org/10.26421/qic14.15-16-2). arXiv: [1311.1074](https://arxiv.org/abs/1311.1074).
- [38] Luca Paolini, Luca Roversi, and Margherita Zorzi. “Quantum Programming Made Easy”. In: *Proc. Linearity-TLLA '18*. Vol. 292. EPTCS, 2019, pp. 133–147. DOI: [10.4204/EPTCS.292.8](https://doi.org/10.4204/EPTCS.292.8).
- [39] Jennifer Paykin, Robert Rand, and Steve Zdancewic. “QWIRE: A Core Language for Quantum Circuits”. In: *Proc. POPL '17*. New York, NY: ACM, 2017, pp. 846–858. DOI: [10.1145/3009837.3009894](https://doi.org/10.1145/3009837.3009894). URL: https://jpaykin.github.io/papers/prz_qwire_2017.pdf.
- [40] Jennifer Paykin and Steve Zdancewic. “A HoTT Quantum Equational Theory (Extended Version)”. Quantum Physics and Logic (QPL) (Chapman University). June 2019. arXiv: [1904.04371](https://arxiv.org/abs/1904.04371).

- [41] Yuxiang Peng, Mingsheng Ying, and Xiaodi Wu. “Algebraic Reasoning of Quantum Programs via Non-Idempotent Kleene Algebra”. In: *Proc. PLDI '22*. New York, NY: ACM, 2022, p. 14. DOI: [10.1145/3519939.3523713](https://doi.org/10.1145/3519939.3523713). arXiv: 2110.07018.
- [42] Frank Pfenning and Carsten Schürmann. “System Description: Twelf—A Meta-Logical Framework for Deductive Systems”. In: *Automated Deduction—CADE-16*. Berlin, Heidelberg: Springer, 1999, pp. 202–206. DOI: [10.1007/3-540-48660-7_14](https://doi.org/10.1007/3-540-48660-7_14). URL: <https://www.cs.cmu.edu/~fp/papers/cade99.pdf>.
- [43] Gordon Plotkin and John Power. “Algebraic Operations and Generic Effects”. In: *Applied Categorical Structures* 11.1 (2003), pp. 69–94. DOI: [10.1023/A:1023064908962](https://doi.org/10.1023/A:1023064908962). URL: https://homepages.inf.ed.ac.uk/gdp/publications/alg_ops_gen_effects.pdf.
- [44] *Q# Language Specification*. Microsoft. Sept. 2020. URL: <https://github.com/microsoft/qsharp-language/tree/main/Specifications/Language#q-language>. See also Bettina Heim. “Development of Quantum Applications”. PhD thesis. ETH Zurich, Dec. 2020. Chap. 8: Domain-Specific Language Q#. DOI: [10.3929/ethz-b-000468201](https://doi.org/10.3929/ethz-b-000468201).
- [45] *QIR Specification*. Version 0.1. QIR Alliance: <https://qir-alliance.org>. Nov. 2021. URL: <https://github.com/qir-alliance/qir-spec>.
- [46] Robert Rand, Kesha Hietala, and Michael Hicks. “Formal Verification vs. Quantum Uncertainty”. In: *3rd Summit on Advances in Programming Languages (SNAPL 2019)*. Vol. 136. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 2019, 12:1–12:11. DOI: [10.4230/LIPIcs.SNAPL.2019.12](https://doi.org/10.4230/LIPIcs.SNAPL.2019.12).
- [47] John C. Reynolds. “A Polymorphic Model of ALGOL”. Unpublished manuscript. 1975.
- [48] John C. Reynolds. “The Essence of ALGOL”. In: *ALGOL-like Languages*. Ed. by Peter W. O’Hearn and Robert D. Tennent. Vol. 1. Boston, MA: Birkhäuser, 1997, pp. 67–88. DOI: [10.1007/978-1-4612-4118-8_4](https://doi.org/10.1007/978-1-4612-4118-8_4). Repr. of “The Essence of ALGOL”. In: *Algorithmic Languages: Proceedings of the International Symposium on Algorithmic Languages*. Amsterdam: North-Holland, 1981, pp. 345–372. URL: <http://www.cs.cmu.edu/afs/cs/user/crary/www/819-f09/Reynolds81.ps>.
- [49] Peter Selinger. “Towards a Quantum Programming Language”. In: *Mathematical Structures in Computer Science* 14.4 (2004), pp. 527–586. DOI: [10.1017/S0960129504004256](https://doi.org/10.1017/S0960129504004256). URL: <https://www.mathstat.dal.ca/~selinger/papers/papers/qpl.pdf>.
- [50] Kartik Singhal. “Quantum Hoare Type Theory”. Master’s paper. University of Chicago, Dec. 2020. arXiv: 2012.02154. URL: <https://ks.cs.uchicago.edu/publication/qhtt-masters/>.
- [51] Kartik Singhal and John Reppy. “Quantum Hoare Type Theory: Extended Abstract”. In: *Proc. QPL '20*. Vol. 340. EPTCS. Homepage: <https://ks.cs.uchicago.edu/publication/qhtt/>. 2021, pp. 291–302. DOI: [10.4204/EPTCS.340.15](https://doi.org/10.4204/EPTCS.340.15).
- [52] Frederick Smith, David Walker, and Greg Morrisett. “Alias Types”. In: *Proc. ESOP '00*. Ed. by Gert Smolka. Berlin, Heidelberg: Springer, 2000, pp. 366–381. DOI: [10.1007/3-540-46425-5_24](https://doi.org/10.1007/3-540-46425-5_24). URL: <https://www.cs.cornell.edu/talc/papers/alias.pdf>.
- [53] Sam Staton. “An Algebraic Presentation of Predicate Logic (Extended Abstract)”. In: *Foundations of Software Science and Computation Structures (FoSSaCS)*. Ed. by Frank Pfenning. Berlin, Heidelberg: Springer, 2013, pp. 401–417. DOI: [10.1007/978-3-642-37075-5_26](https://doi.org/10.1007/978-3-642-37075-5_26). URL: <http://www.cs.ox.ac.uk/people/samuel.staton/papers/fossacs13.pdf>.

- [54] Sam Staton. “Algebraic Effects, Linearity, and Quantum Programming Languages”. In: *Proc. POPL ’15*. New York, NY: ACM, 2015, pp. 395–406. DOI: 10.1145/2676726.2676999. URL: <http://www.cs.ox.ac.uk/people/samuel.staton/papers/popl2015.pdf>.
- [55] Krysta M. Svore, Alfred V. Aho, Andrew W. Cross, Isaac L. Chuang, and Igor L. Markov. “A Layered Software Architecture for Quantum Computing Design Tools”. In: *Computer* 39.1 (2006), pp. 74–83. DOI: 10.1109/MC.2006.4. URL: <https://web.eecs.umich.edu/~imarkov/pubs/jour/computer06-q.pdf>.
- [56] Krysta M. Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher E. Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. “Q#: Enabling Scalable Quantum Computing and Development with a High-Level DSL”. In: *Proceedings of the Real World Domain Specific Languages Workshop 2018*. RWDSL ’18. New York, NY: ACM, Feb. 2018, 7:1–7:10. DOI: 10.1145/3183895.3183901. arXiv: 1803.00652.
- [57] The Univalent Foundations Program. *Homotopy Type Theory. Univalent Foundations of Mathematics*. Princeton, NJ: Institute for Advanced Study, 2013. URL: <https://homotopytypetheory.org/book>.
- [58] David Walker. “Typed Memory Management”. Order no. 9988176. PhD thesis. Cornell University, 2001. URL: https://www.cs.cmu.edu/~dpw/papers/thesis_ps.gz.
- [59] David Walker and Greg Morrisett. “Alias Types for Recursive Data Structures”. In: *Types in Compilation*. Berlin, Heidelberg: Springer, 2001, pp. 177–206. DOI: 10.1007/3-540-45332-6_7. URL: <https://www.cs.cornell.edu/talc/papers/alias-recursion.pdf>.
- [60] W. K. Wootters and W. H. Zurek. “A single quantum cannot be cloned”. In: *Nature* 299.5886 (1982), pp. 802–803. DOI: 10.1038/299802a0. URL: <https://copilot.caltech.edu/documents/17036/299802a0.pdf>.
- [61] Yannick Zakowski, Calvin Beck, Irene Yoon, Ilia Zaichuk, Vadim Zaliva, and Steve Zdancewic. “Modular, Compositional, and Executable Formal Semantics for LLVM IR”. In: *Proc. ACM Program. Lang.* 5.ICFP, 67 (2021). DOI: 10.1145/3473572. URL: <https://github.com/vellvm/vellvm>.
- [62] Jianzhou Zhao, Santosh Nagarakatte, Milo M. K. Martin, and Steve Zdancewic. “Formalizing the LLVM Intermediate Representation for Verified Program Transformations”. In: *Proc. POPL ’12*. New York, NY: ACM, 2012, pp. 427–440. DOI: 10.1145/2103656.2103709. URL: <https://www.cis.upenn.edu/~stevez/papers/ZNMZ12.pdf>.

A Derived Forms

In addition to the syntax shown in [Figure 2](#), we use these straightforward derived forms from Harper’s language **MA** [17]:

$$\begin{aligned}
\{x \leftarrow m_1; m_2\} &\triangleq \text{bnd } x \leftarrow \text{cmd}(m_1) ; m_2 \\
\{x_1 \leftarrow m_1; \dots x_{n-1} \leftarrow m_{n-1}; m_n\} &\triangleq \{x_1 \leftarrow m_1; \dots \{x_{n-1} \leftarrow m_{n-1}; m_n\}\} \\
\{m_1; m_2\} &\triangleq \{- \leftarrow m_1; m_2\} \\
\{m_1; \dots m_{n-1}; m_n\} &\triangleq \{m_1; \dots \{m_{n-1}; m_n\}\} \\
\text{do } m &\triangleq \{x \leftarrow m; \text{ret}(x)\} \\
\tau_1 \Rightarrow \tau_2 &\triangleq \tau_1 \rightarrow \text{cmd}(\tau_2) \\
\text{proc } (x : \tau) m &\triangleq \lambda(x. \text{cmd}(m)) \\
\text{call } e_1(e_2) &\triangleq \text{do}(\text{ap}(e_1; e_2)) \\
\text{call } e &\triangleq \text{call } e(\langle \rangle)
\end{aligned}$$

B Remaining Static and Dynamic Rules

These are standard rules from Harper’s **PFPL** [18] adapted to quantum computation. Note that in the **PFPL** terminology, we are following the *scoped dynamics* of symbols [18, Ch. 31]. Instead of typed assignables [18, §34.3], we only have a single type of qubit symbols, which we hence do not annotate in the signature, Σ , i.e., the signature only contains active qubit symbols in scope and nothing else. Further, since there are no reference types [18, Ch. 35] except for a single qubit reference type, we do not explicitly state any mobility conditions [18, §31.1]. Under scoped dynamics, qubit references are immobile [18, §35.2]. This mobility restriction is crucial in ensuring the stack discipline for qubit management.

B.1 Type System

We provide the most interesting rules of our type system in [Figure 3](#). We include the following rules here for completeness.

$\Gamma \vdash e : \tau$	<i>(Expression e has type τ in context Γ)</i>		
TY-VAR	TY-LET	TY-LAM	TY-AP
$\frac{}{\Gamma, x : \tau \vdash x : \tau}$	$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \text{let}(e_1; x.e_2) : \tau_2}$	$\frac{}{\Gamma \vdash \lambda \{ \tau_1 \}(x.e) : \text{fun}(\tau_1; \tau_2)}$	$\frac{\Gamma \vdash e_1 : \text{fun}(\tau_2; \tau) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \text{ap}(e_1; e_2) : \tau}$
TY-PR	TY-TPL		
$\frac{\Gamma \vdash e : \text{prod}(\overline{i \mapsto \tau_i}^{i \in 1..n}) \quad 1 \leq i \leq n}{\Gamma \vdash \text{proj } \langle i \rangle(e) : \tau_i}$	$\frac{\overline{\Gamma \vdash e_i : \tau_i}^{i \in 1..n}}{\Gamma \vdash \text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n}) : \text{prod}(\overline{i \mapsto \tau_i}^{i \in 1..n})}$		

In rule **TYS-QLOC** and rule **VS-QLOC** in the next subsection, $\mathbf{qloc}[q]$ is the value of the reference to an active qubit symbol q . It can be thought of as a classical pointer value indexed by a qubit symbol. The signature plays a role wherever commands or qubits are involved.

$\boxed{\Gamma \vdash_{\Sigma} e : \tau}$ (Expression e has type τ relative to the signature)

$$\frac{\text{TYS-CMD} \quad \Gamma \vdash_{\Sigma} m \dot{\sim} \tau}{\Gamma \vdash_{\Sigma} \text{cmd}(m) : \text{cmd}(\tau)} \qquad \text{TYS-QLOC} \quad \frac{}{\Gamma \vdash_{\Sigma, q} \mathbf{qloc}\langle q \rangle : \mathbf{qref}\langle q \rangle}$$

B.2 Operations Semantics

Pure Classical Sub-language

$\boxed{e \text{ val}}$ (e is a value)

$$\frac{\text{V-LAM}}{\lambda \{ \tau \}(x.e) \text{ val}} \qquad \text{V-TPL} \quad \frac{\overline{e_i \text{ val}}^{i \in 1..n}}{\text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n}) \text{ val}}$$

$\boxed{e \mapsto e'}$ (e steps to e')

$$\frac{\text{TR-LET} \quad e_1 \mapsto e'_1}{\text{let}(e_1; x.e_2) \mapsto \text{let}(e'_1; x.e_2)} \qquad \text{TR-LETINSTR} \quad \frac{e_1 \text{ val}}{\text{let}(e_1; x.e_2) \mapsto [e_1/x]e_2} \qquad \text{TR-APL} \quad \frac{e_1 \mapsto e'_1}{\text{ap}(e_1; e_2) \mapsto \text{ap}(e'_1; e_2)}$$

$$\frac{\text{TR-APR} \quad \frac{e_1 \text{ val} \quad e_2 \mapsto e'_2}{\text{ap}(e_1; e_2) \mapsto \text{ap}(e_1; e'_2)}}{\text{ap}(e_1; e_2) \mapsto \text{ap}(e_1; e'_2)} \qquad \text{TR-APISTR} \quad \frac{e_2 \text{ val}}{\text{ap}(\lambda \{ \tau_2 \}(x.e_1); e_2) \mapsto [e_2/x]e_1}$$

$$\text{TR-TPL} \quad \frac{\overline{e_i \text{ val}}^{i \in 1..n_a} \quad e \mapsto e'}{\text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n_a}, k \mapsto e, \overline{j \mapsto e'_j}^{j \in 1..n_b}) \mapsto \text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n_a}, k \mapsto e', \overline{j \mapsto e'_j}^{j \in 1..n_b})}$$

$$\text{TR-PR} \quad \frac{e \mapsto e'}{\text{proj}\langle i \rangle(e) \mapsto \text{proj}\langle i \rangle(e')} \qquad \text{TR-PRINSTR} \quad \frac{\text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n}) \text{ val} \quad 1 \leq j \leq n}{\text{proj}\langle j \rangle(\text{tuple}(\overline{i \mapsto e_i}^{i \in 1..n})) \mapsto e_j}$$

$\boxed{e \text{ val}_{\Sigma}}$ (e is a value relative to Σ)

$$\frac{\text{VS-CMD}}{\text{cmd}(m) \text{ val}_{\Sigma}} \qquad \text{VS-QLOC} \quad \frac{}{\mathbf{qloc}\langle q \rangle \text{ val}_{\Sigma, q}}$$

Effectful Quantum Sublanguage In the following rules, we do not show the quantum store, preferring the equational dynamics shown in §3.3. In reading these rules, consider the quantum store shape being expanded and restored during the allocation command (as reflected in the signature) and the quantum state being modified during the measurement and the gate application commands.

$$\boxed{m \text{ final}_\Sigma} \quad (\text{Command } m \text{ is complete})$$

$$\text{FN-RET} \quad \frac{e \text{ val}_\Sigma}{\text{ret}(e) \text{ final}_\Sigma}$$

$$\boxed{m \mapsto_\Sigma m'} \quad (\text{Command } m \text{ steps to } m')$$

$$\begin{array}{c}
\text{ST-RET} \quad \frac{e \mapsto_\Sigma e'}{\text{ret}(e) \mapsto_\Sigma \text{ret}(e')} \quad \text{ST-BND} \quad \frac{e \mapsto_\Sigma e'}{\text{bnd}(e; x.m) \mapsto_\Sigma \text{bnd}(e'; x.m)} \quad \text{ST-BNDINSTR} \quad \frac{e \text{ val}_\Sigma}{\text{bnd}(\text{cmd}(\text{ret}(e)); x.m) \mapsto_\Sigma [e/x]m} \\
\text{ST-BNDCMD} \quad \frac{m_1 \mapsto_\Sigma m'_1}{\text{bnd}(\text{cmd}(m_1); x.m_2) \mapsto_\Sigma \text{bnd}(\text{cmd}(m'_1); x.m_2)} \quad \text{ST-NEWQREF} \quad \frac{m \mapsto_{\Sigma, q} m'}{\text{newqref}(x.m) \mapsto_\Sigma \text{newqref}(x.m')} \\
\text{ST-NEWQREFINSTR} \quad \frac{e \text{ val}_\Sigma}{\text{newqref}(x.\text{ret}(e)) \mapsto_\Sigma \text{ret}(e)} \quad \text{ST-GATEAPREF} \quad \frac{e \mapsto_\Sigma e'}{\text{gateap}\langle U_{2^n} \rangle(e) \mapsto_\Sigma \text{gateap}\langle U_{2^n} \rangle(e')} \\
\text{ST-DIAGAPREFL} \quad \frac{e_1 \mapsto_\Sigma e'_1}{\text{diagap}\langle U_{2^n}, V_{2^n} \rangle(e_1; e_2) \mapsto_\Sigma \text{diagap}\langle U_{2^n}, V_{2^n} \rangle(e'_1; e_2)} \\
\text{ST-DIAGAPREFR} \quad \frac{e_1 \text{ val}_\Sigma \quad e_2 \mapsto_\Sigma e'_2}{\text{diagap}\langle U_{2^n}, V_{2^n} \rangle(e_1; e_2) \mapsto_\Sigma \text{diagap}\langle U_{2^n}, V_{2^n} \rangle(e_1; e'_2)} \quad \text{ST-MEASREF} \quad \frac{e \mapsto_\Sigma e'}{\text{meas}(e) \mapsto_\Sigma \text{meas}(e')}
\end{array}$$

C Correspondence between $\lambda_{Q\#}$ and Staton's quantum language

We can easily translate the quantum-specific fragment of $\lambda_{Q\#}$ to Staton's quantum programming language [54, §5]. Note that the generic effects of his quantum language are equivalent to the algebraic operations (of the algebraic theory) [43] that he uses in his proof of Theorem 11 [54, pp. 12–15, Appendix A]. The translation follows:

$$\begin{aligned}
\text{let } q = \underline{\text{new}}() \text{ in } m &\equiv \text{newqref}(q.m) \\
\underline{\text{measure}}(e) &\equiv \text{meas}(e) \\
\underline{\text{apply}}_U(e) &\equiv \text{gateap}\langle U_{2^n} \rangle(e)
\end{aligned}$$

Note that $\text{gateap}\langle U_{2^n} \rangle(e)$ subsumes $\text{diagap}\langle U_{2^n}, V_{2^n} \rangle(e_1; e_2)$, just like in Staton's work. In other words, we can define each of Staton's terms using terms of our language.

D An Elaboration Example

Listing 1 shows a sample Q# program. Figure 4 shows the corresponding elaboration to $\lambda_{Q\#}$.

```
namespace Quantum.Kata.Teleportation {
    open Microsoft.Quantum.Intrinsic; // for H, X, Z, CNOT, and M

    operation Entangle (qAlice : Qubit, qBob : Qubit) : Unit is Adj {
        H(qAlice);
        CNOT(qAlice, qBob);
    }

    operation SendMsg (qAlice : Qubit, qMsg : Qubit) : (Bool, Bool) {
        CNOT(qMsg, qAlice);
        H(qMsg);
        return (M(qMsg) == One, M(qAlice) == One);
    }

    operation DecodeMsg (qBob : Qubit, (b1 : Bool, b2 : Bool)) : Unit {
        if b1 { Z(qBob); }
        if b2 { X(qBob); }
    }

    operation Teleport (qAlice : Qubit, qBob : Qubit, qMsg : Qubit) : Unit {
        Entangle(qAlice, qBob);
        let classicalBits = SendMsg(qAlice, qMsg);
        DecodeMsg(qBob, classicalBits);
    }
}
```

Listing 1: Teleportation in Q# (adapted from Quantum Katas [32]).

```

let Entangle be proc ((qAlice, qBob) : qref⟨a⟩ × qref⟨b⟩) {
  applyH(qAlice);
  applyD(I2, X)(qAlice; qBob)} in
let SendMsg be proc ((qAlice, qMsg) : qref⟨a⟩ × qref⟨m⟩) {
  applyD(I2, X)(qMsg; qAlice);
  applyH(qMsg);
  ret((cmd(meas(qMsg)), cmd(meas(qAlice))))} in
let DecodeMsg be proc ((qBob, ⟨b1, b2⟩) : qref⟨b⟩ × (bool × bool)) {
  if b1 then applyZ(qBob) else ⟨⟩;
  if b2 then applyX(qBob) else ⟨⟩} in
let Teleport be proc ((qAlice, qBob, qMsg) : qref⟨a⟩ × qref⟨b⟩ × qref⟨m⟩) {
  call Entangle((qAlice, qBob));
  classicalBits ← call SendMsg((qAlice, qMsg));
  call DecodeMsg((qBob, classicalBits))} in ⟨⟩

```

Figure 4: $\lambda_{Q\#}$ elaboration of the Q# program in Listing 1.

Universal Properties of Partial Quantum Maps

Pablo Andrés-Martínez*

University of Edinburgh
Quantinuum

Chris Heunen†

University of Edinburgh

Robin Kaarsgaard‡

University of Southern Denmark

We provide a universal construction of the category of finite-dimensional C^* -algebras and completely positive trace-nonincreasing maps from the rig category of finite-dimensional Hilbert spaces and unitaries. This construction, which can be applied to any dagger rig category, is described in three steps, each associated with their own universal property, and draws on results from dilation theory in finite dimension. In this way, we explicitly construct the category that captures hybrid quantum/classical computation with possible nontermination from the category of its reversible foundations. We discuss how this construction can be used in the design and semantics of quantum programming languages.

1 Introduction

The account of quantum measurement offered by *decoherence* establishes that the irreversible nature of mixed-state evolution occurs when a system is considered in isolation from its environment. When the environment is brought back into view, mathematically through techniques such as *quantum state purification* and *Stinespring dilation*, the reversible underpinnings of mixed-state evolution are exposed.

This perspective has in recent years led to the study of quantum theory through categorical completions of its reversible foundations, the category of finite-dimensional Hilbert spaces and unitaries, demonstrating connections between *universal* constructions and effectful quantum programming [11]. This article constructs in a universal way the category of finite-dimensional C^* -algebras and *partial quantum channels* (completely positive trace-nonincreasing maps) from the rig category of finite-dimensional Hilbert spaces and unitaries. The construction has three stages, each with a universal property of its own.

- Freely allowing *partiality respecting the dagger structure* (by making the additive unit a *zero object*) allows contractive maps to be described by unitaries through *Halmos dilation* [8, 25, 20].
- Freely allowing the *hiding of states* in a way that *respects partiality* (by making the multiplicative unit *terminal* for *total* maps) allows completely positive trace-nonincreasing maps to be described through contractions, using a variant of *Stinespring dilation* [29]. This construction has an interesting universal property as a pushout of monoidal categories.
- Freely splitting certain idempotents on finite-dimensional Hilbert spaces yields finite-dimensional C^* -algebras, which describe *hybrid quantum/classical* computation.

All three universal constructions are abstract and apply to any suitably structured category. They show that the traditional model of C^* -algebras inevitably arises from the mere concepts of quantum circuits, partiality, hiding, and classical communication, without any concept of *e.g.* norm. Thus they inform the design of quantum programming languages [11], as part of a highly effective broader approach to program semantics from universal properties [30, 16, 27].

*Supported by EPSRC grant EP/L01503X/1 via the CDT in Pervasive Parallelism

†Supported by EPSRC Fellowship EP/R044759/1

‡Supported by *DFF | Natural Sciences* International Postdoctoral Fellowship 0131-00025B

Related work The role of universal properties and categorical completions in quantum theory, in particular the *monoidal indeterminates* construction [9], has been studied in recent years [24, 14, 15, 4, 10, 11] (see also [28] for a related approach in the probabilistic case, and [31, 22] for other accounts of dilations as universal properties). The role of partiality in effectus theory was studied in [2]. We deepen the connections between dilations and universal properties of functors between categories of quantum systems first observed in [14, 15]. A direct connection between universal constructions and the design and semantics of quantum programming languages with effects was demonstrated in [11]. The idempotent splitting of the category of finite-dimensional Hilbert spaces and completely positive maps, in particular the fact that it contains all finite-dimensional C*-algebras, has been the subject of study and discussion in [12, 24, 3].

Overview Section 2 recalls some facts about rig categories and their additive affine completion and relation to partiality, which we extend in Section 3 to the biaffine completion and dagger partiality, and show that the biaffine completion of the category **Unitary** of (finite-dimensional Hilbert spaces and) unitaries is precisely **Contraction** of contractive maps. Section 4 describes a construction that completes **Contraction** to the category **FHil**_{CPTN} of *partial quantum channels* (completely positive trace-nonincreasing maps) using a variant of Stinespring dilation, and we show that this construction satisfies a universal property as a pushout in the category of monoidal categories. Finally, Section 5 shows that splitting along a particular class of idempotents, corresponding in **FHil**_{CPTN} to *measurements*, completes **FHil**_{CPTN} to the category **FCstar**_{CPTN} of finite-dimensional C*-algebras and partial quantum channels. We end in Section 6 with a discussion of the applications of these constructions to the design and semantics of quantum programming languages.

2 The additive affine completion of rig categories

We enter the story assuming that the reader is familiar with monoidal categories and dagger categories [13], and proceed with preliminaries about rig categories and their additive affine completion.

A *rig category* is a category which is symmetric monoidal in two different ways, such that one monoidal product distributes over the other, subject to a large amount of coherence equations [19]. In analogy with the situation in Hilbert spaces, we usually write these monoidal products as (\otimes, I) (the “tensor product”) and (\oplus, O) (the “direct sum”) with \otimes distributing (up to natural isomorphism) over \oplus via distributors $\delta^L: A \otimes (B \oplus C) \rightarrow (A \otimes B) \oplus (A \otimes C)$ and $\delta^R: (A \oplus B) \otimes C \rightarrow (A \otimes C) \oplus (B \otimes C)$ and annihilators (“nullary distributors”) $\delta_0^L: O \otimes A \rightarrow O$ and $\delta_0^R: A \otimes O \rightarrow O$. A *dagger rig category* is a dagger category with a rig structure such that all coherence isomorphisms are unitary (*i.e.*, satisfy $f^{-1} = f^\dagger$). Natural examples of rig categories are *distributive categories* where \otimes is a categorical product with a terminal object as its monoidal unit and \oplus is a categorical coproduct with an initial unit. However, not all rig categories are of this form: the category **Unitary** of finite-dimensional Hilbert spaces and unitaries (with tensor product and direct sum) and the category **FinBij** of finite sets and bijections (with cartesian product and disjoint union) are both (dagger) rig categories, but neither has products or coproducts.

2.1 Partiality and the additive affine completion

What is the appropriate notion of *partiality* for a given rig category? If coproducts and a terminal object are available the *lift monad* $(-)+1$ can answer this question, but not every rig category has these. Instead, we can think of a partial map $A \rightarrow B$ as a map $A \rightarrow B \oplus E$, *i.e.*, extend the output state space with an extra

part E to receive all the inputs we wish to be undefined. Any map $f: A \rightarrow B$ can be lifted to a total map $\rho_{\oplus}^{-1} \circ f: A \rightarrow B \oplus O$ using the inverse right unitor, and partial maps $f: A \rightarrow B \oplus E$ and $g: B \rightarrow C \oplus E'$ can be composed by composing their defined parts, i.e., $\alpha_{\oplus} \circ g \otimes \text{id}_E \circ f: A \rightarrow C \oplus (E' \oplus E)$. This is an information-preserving variant of Kleisli-composition for the lift monad.

This describes the *additive affine completion* of a rig category, barring one detail: given that E describes where the map is undefined, it shouldn't actually matter how we represent this particular part. For example, given some partial map $f: A \rightarrow B \oplus E$ and some manipulation $m: E \rightarrow E'$ of the undefined part, f and $(\text{id} \oplus m) \circ f$ morally describe the same partial map. Therefore, given morphisms $f: A \rightarrow B \oplus E$ and $f': A \rightarrow B \oplus E'$, we write $f \leq_L f'$ if and only if there exists some *mediator* $m: E \rightarrow E'$ such that

$$\begin{array}{ccc} & A & \\ f \swarrow & & \searrow f' \\ B \oplus E & \xrightarrow{\text{id} \oplus m} & B \oplus E' \end{array}$$

commutes. This straightforwardly gives a preorder, though not (necessarily) an equivalence relation since mediators need not be invertible. However, since we would like momentarily to treat it as an equivalence, we consider instead its equivalence closure \sim_L , i.e., the least equivalence relation containing \leq_L .

Definition 1. Given a rig category \mathbf{C} , its *additive affine completion* $L_{\oplus}(\mathbf{C})$ is the category whose

- objects are those of \mathbf{C} ,
- morphisms $[f, E]: A \rightarrow B$ are pairs of an object E and an equivalence class of morphisms $f: A \rightarrow B \oplus E$ of \mathbf{C} under \sim_L ,
- identities $A \rightarrow A$ are $[\rho_{\oplus}^{-1}, O]$ (with $\rho_{\oplus}: A \oplus O \rightarrow A$ the right unitor), and
- composition of $[f, E]: A \rightarrow B$ and $[g, E']: B \rightarrow C$ is $[\alpha_{\oplus} \circ g \otimes \text{id}_E \circ f, E' \oplus E]$.

There is a dual to this construction, the *additive coaffine completion* $R_{\oplus}(\mathbf{C})$, defined as $L_{\oplus}(\mathbf{C}^{\text{op}})^{\text{op}}$. Explicitly, morphisms $A \rightarrow B$ in $R_{\oplus}(\mathbf{C})$ are equivalence classes of morphisms $A \oplus E \rightarrow B$, and so hide part of their *source* space rather than their *target* space. We summarise some features of these categories.

Proposition 2. When \mathbf{C} is a rig category, so are $L_{\oplus}(\mathbf{C})$ and $R_{\oplus}(\mathbf{C})$.

Proof. That $R_{\oplus}(\mathbf{C})$ is a rig category was shown in [11, Lemma 12]; that $L_{\oplus}(\mathbf{C})$ is also a rig category follows by $L_{\oplus}(\mathbf{C}) \cong R_{\oplus}(\mathbf{C}^{\text{op}})^{\text{op}}$ and the fact that \mathbf{C} is a rig category iff \mathbf{C}^{op} is. \square

Proposition 3. The additive unit O is terminal in $L_{\oplus}(\mathbf{C})$ and initial in $R_{\oplus}(\mathbf{C})$.

Proof. The inverse left unitor $\lambda_{\oplus}^{-1}: A \rightarrow O \oplus A$ of \mathbf{C} represents a morphism $A \rightarrow O$ in $L_{\oplus}(\mathbf{C})$; this satisfies the universal property of the terminal object by definition of \sim_L . Dually, O is initial in $R_{\oplus}(\mathbf{C})$. \square

We call a rig category *additively coaffine* when the additive unit is initial, and *additively affine* when it is terminal.

Proposition 4. There are strict rig functors $\mathcal{D}: \mathbf{C} \rightarrow L_{\oplus}(\mathbf{C})$ and $\mathcal{E}: \mathbf{C} \rightarrow R_{\oplus}(\mathbf{C})$.

Proof. Define $\mathcal{D}(A) = A$ on objects, and $\mathcal{D}(f) = [\rho_{\oplus}^{-1} \circ f, O]$ on morphisms, and \mathcal{E} dually. Straightforward calculations show that this defines strict rig functors. \square

As the name suggests, these are, indeed, completions on rig categories.

Proposition 5. $L_{\oplus}(\mathbf{C})$ is the additive affine completion of \mathbf{C} in the following sense: given any additively affine rig category \mathbf{D} and strong rig functor $F: \mathbf{C} \rightarrow \mathbf{D}$, there is a unique strong rig functor $\hat{F}: L_{\oplus}(\mathbf{C}) \rightarrow \mathbf{D}$ making the diagram below commute.

$$\begin{array}{ccc} \mathbf{C} & \xrightarrow{\mathcal{G}} & L_{\oplus}(\mathbf{C}) \\ & \searrow F & \downarrow \hat{F} \\ & & \mathbf{D} \end{array}$$

Proof. By [11, Theorem 19] and duality. □

Dualising this result exhibits $R_{\oplus}(\mathbf{C})$ as the additive coaffine completion of \mathbf{C} .

3 Dagger partiality and the additive biaffine completion

As the (co)affine completion explicitly involves hiding a part of the source or target space of a morphism, we cannot expect to lift either construction to a completion of *dagger* rig categories. A great example of this fact is demonstrated by considering $R_{\oplus}(\mathbf{Unitary})$. Write **Isometry** for the category of finite-dimensional Hilbert spaces and morphisms satisfying $f^{\dagger} \circ f = \text{id}$, and **coIsometry** for its dual.

Proposition 6. *There are rig equivalences $R_{\oplus}(\mathbf{Unitary}) \simeq \mathbf{Isometry}$ and $L_{\oplus}(\mathbf{Unitary}) \simeq \mathbf{coIsometry}$.*

Proof. That $R_{\oplus}(\mathbf{Unitary}) \simeq \mathbf{Isometry}$ was shown in [14]. The other statement follows from duality: $L_{\oplus}(\mathbf{Unitary}) \simeq R_{\oplus}(\mathbf{Unitary}^{\text{op}})^{\text{op}} \simeq R_{\oplus}(\mathbf{Unitary})^{\text{op}} \simeq \mathbf{Isometry}^{\text{op}} \simeq \mathbf{coIsometry}$. □

However, though **Unitary** is a dagger rig category, **Isometry** and **coIsometry** are mere rig categories. Intuitively, this must be the case because dagger categories are self-dual (*i.e.*, satisfy $\mathbf{C} \cong \mathbf{C}^{\text{op}}$) so limits and colimits coincide, but the affine completion only adds (certain) limits without the corresponding colimits. However, this also suggests that if we seek a notion of partiality that respects daggers, we would need the additive unit to be both initial and terminal, *i.e.*, a *zero object*. Fortunately, we can ensure this by applying L_{\oplus} after R_{\oplus} or vice versa.

Proposition 7. *The additive unit O is a zero object in both $L_{\oplus}(R_{\oplus}(\mathbf{C}))$ and $R_{\oplus}(L_{\oplus}(\mathbf{C}))$.*

Proof. By [11, Lemma 11], $R_{\oplus}(-)$ preserves terminal objects, and by duality, $L_{\oplus}(-)$ preserves initial objects. Thus O is both initial and terminal (*i.e.*, a zero object) in both $L_{\oplus}(R_{\oplus}(\mathbf{C}))$ and $R_{\oplus}(L_{\oplus}(\mathbf{C}))$. □

The situation is interesting: neither $L_{\oplus}(-)$ nor $R_{\oplus}(-)$ on their own preserve dagger rig categories, but as we will see, their combination does. Hence it is advantageous to consider them together for dagger rig categories, which also leads to a slightly simpler presentation. In a rig category, define $\sim_{LR_{\oplus}}$ as the least equivalence relation containing the three relations $\sim_{\text{id}_{\oplus}}$, $\sim_{L_{\oplus}}$, and $\sim_{R_{\oplus}}$ defined as follows, for all $f: A \oplus H \rightarrow B \oplus G$:

- $f \sim_{L_{\oplus}} (\text{id} \oplus m) \circ f$ for all $m: G \rightarrow G'$;
- $f \sim_{R_{\oplus}} f \circ (\text{id} \oplus n)$ for all $n: H' \rightarrow H$;
- $f \sim_{\text{id}_{\oplus}} \alpha_{\oplus} \circ (f \oplus \text{id}_X) \circ \alpha_{\oplus}^{-1}$ for all identities id_X .

Definition 8. Given a rig category \mathbf{C} , its *biaffine completion* $LR_{\oplus}(\mathbf{C})$ is the category whose

- objects are those of \mathbf{C} ,

- morphisms $[H, f, G]: A \rightarrow B$ are triples consisting of two objects H and G and a morphism $A \oplus H \rightarrow B \oplus G$ of \mathbf{C} quotiented by $\sim_{LR_{\oplus}}$,
- identities $A \rightarrow A$ are $[O, \text{id}_{A \oplus O}, O]$, and
- the composition of $f: A \oplus H \rightarrow B \oplus G$ and $g: B \oplus H' \rightarrow C \oplus G'$ is given by

$$A \oplus (H \oplus H') \xrightarrow{\alpha_{\oplus}^{-1}} (A \oplus H) \oplus H' \xrightarrow{f \oplus \text{id}} (B \oplus G) \oplus H' \xrightarrow{\cong} (B \oplus H') \oplus G \xrightarrow{g \oplus \text{id}} (C \oplus G') \oplus G \xrightarrow{\alpha_{\oplus}} C \oplus (G' \oplus G).$$

We state some properties of the LR_{\oplus} -construction, the proofs of which can be found in the appendix.

Proposition 9. *The additive unit O is a zero object in $LR_{\oplus}(\mathbf{C})$.*

Proposition 10. *When \mathbf{C} is a dagger rig category, so is $LR_{\oplus}(\mathbf{C})$.*

As before, there is a functor $\mathcal{F}: \mathbf{C} \rightarrow LR_{\oplus}(\mathbf{C})$ given on objects by $\mathcal{F}(A) = A$ and on morphisms by $F(f) = [0, f \oplus \text{id}_0, 0]$, making $LR_{\oplus}(\mathbf{C})$ a completion in the formal sense. We say that a rig category is *additively biaffine* if the unit of the sum is a zero object.

Theorem 11. *$LR_{\oplus}(\mathbf{C})$ is the additive biaffine completion of \mathbf{C} in the following sense: given any additively biaffine rig category \mathbf{D} and strong rig functor $F: \mathbf{C} \rightarrow \mathbf{D}$, there is a unique strong rig functor $\widehat{F}: LR_{\oplus}(\mathbf{C}) \rightarrow \mathbf{D}$ making the diagram below commute.*

$$\begin{array}{ccc} \mathbf{C} & \xrightarrow{\mathcal{F}} & LR_{\oplus}(\mathbf{C}) \\ & \searrow F & \downarrow \widehat{F} \\ & & \mathbf{D} \end{array}$$

Proof. Assuming such a functor exists, we first prove its uniqueness. Imposing $F = \widehat{F} \circ \mathcal{F}$ implies that $\widehat{F}(A) = F(A)$ on objects and, on morphisms in the image of \mathcal{F} , we have that $\widehat{F}(\mathcal{F}(f)) = F(f)$. It is easy to check (for instance, diagrammatically) that any $[H, f, G]: A \rightarrow B$ in $LR_{\oplus}[\mathbf{C}]$ decomposes as follows:

$$[H, f, G] = A \xrightarrow{\rho^{-1}} A \oplus O \xrightarrow{\text{id} \oplus !} A \oplus H \xrightarrow{\mathcal{F}(f)} B \oplus G \xrightarrow{\text{id} \oplus !} B \oplus O \xrightarrow{\rho} B$$

where the objects and morphisms shown are in $LR_{\oplus}[\mathbf{C}]$, where O is a zero object, and $!$ refer to the corresponding unique morphisms. The image under \widehat{F} of each of these morphisms is uniquely determined because \widehat{F} is assumed to be monoidal, the fact that \mathbf{D} has $0 \cong F(O)$ as its zero object, and the equality $\widehat{F}(\mathcal{F}(f)) = F(f)$ discussed above. Thus it only remains to prove that \widehat{F} as defined above is indeed a strong rig functor. By definition, $\widehat{F}(\text{id}) = F(\text{id})$, and functoriality is easy to check using naturality of ρ and the fact that 0 is a zero object. The fact that \widehat{F} is a strong rig functor follows directly from the same property for F , since \mathcal{F} is a strict rig functor, $\widehat{F}(O) = F(O) \cong 0$, and $\widehat{F}(A \oplus B) = F(A \oplus B) \cong F(A) \oplus F(B) = \widehat{F}(A) \oplus \widehat{F}(B)$. \square

Using this characterisation, we can show commutativity of the L_{\oplus} and R_{\oplus} constructions, the proof of which is found in the appendix.

Proposition 12. *When \mathbf{C} is a rig category, $L_{\oplus}(R_{\oplus}(\mathbf{C})) \cong LR_{\oplus}(\mathbf{C}) \cong R_{\oplus}(L_{\oplus}(\mathbf{C}))$.*

3.1 Unitaries with dagger partiality are contractions

A linear map $f: A \rightarrow B$ between normed spaces A and B is (weakly) *contractive* iff $\|f(x)\| \leq \|x\|$ for all $x \in A$. So contractive maps include all isometries, coisometries, and unitaries. We now show that applying the LR_{\oplus} -construction to the category of finite-dimensional Hilbert spaces and unitaries is equivalent to the category **Contraction** of finite-dimensional Hilbert spaces and contractions. Another way of saying this is that extending the unitaries with dagger partiality gives precisely the contractions.

Theorem 13. $LR_{\oplus}(\mathbf{Unitary}) \cong \mathbf{Contraction}$.

Proof. Proposition 12 gives $LR_{\oplus}(\mathbf{Unitary}) \cong L_{\oplus}(R_{\oplus}(\mathbf{Unitary}))$, and Proposition 6 gives $R_{\oplus}(\mathbf{Unitary}) \cong \mathbf{Isometry}$, so it suffices to show $L_{\oplus}(\mathbf{Isometry}) \cong \mathbf{Contraction}$. The strategy is to define a functor $F: \mathbf{Contraction} \rightarrow L_{\oplus}(\mathbf{Isometry})$ and prove it is full, faithful and essentially surjective. On objects, $F(A) = A$ so F is essentially surjective. Let $T: A \rightarrow B$ be a contraction and let $f: A \rightarrow B \oplus G$ be an isometry such that $T = \pi_B \circ f$: for example, by [8, 25], an isometric dilation $A \rightarrow B \oplus A$ can be constructed as

$$f = \begin{pmatrix} T \\ (1 - T^{\dagger}T)^{\frac{1}{2}} \end{pmatrix}.$$

Let $F(T) = [f, G]$. We need to check this mapping is well-defined, *i.e.*, if $f': A \rightarrow B \oplus G'$ also satisfies $T = \pi_B \circ f'$, verify that $[f, G] = [f', G']$. To do this, first consider the Hilbert space $\text{im}(f) \subseteq B \oplus G$ and similarly $\text{im}(f')$ and define a function $g: \text{im}(f') \rightarrow \text{im}(f)$ by $f'(a) \mapsto f(a)$. This function is well-defined because f and f' are isometries and hence injective. It is easy to see that g is linear; also notice that

$$\langle g(f'a) | g(f'a) \rangle = \langle fa | fa \rangle = \langle a | a \rangle = \langle f'a | f'a \rangle$$

because f and f' are isometries. This applies to all vectors in $\text{im}(f')$, so g is an isometry; in fact, because $\dim(\text{im}(f)) = \dim(\text{im}(f'))$ is finite, g has an inverse that is also an isometry, so $g: \text{im}(f') \rightarrow \text{im}(f)$ is unitary. Whenever $f(a) \in B$ it is necessary that $f(a) = f'(a)$ for them to be dilations of T , which means that g acts as the identity on $\text{im}(T)$. Since $\text{im}(f) = (\text{im}(f) \cap B) \oplus (\text{im}(f) \cap G)$ and similarly for $\text{im}(f')$, while $\text{im}(f) \cap B = \text{im}(T) = \text{im}(f') \cap B$, we can decompose g into a block matrix of the form $g: \text{im}(T) \oplus (\text{im}(f') \cap G') \rightarrow \text{im}(T) \oplus (\text{im}(f) \cap G)$. Given that the component $\text{im}(T) \rightarrow \text{im}(T)$ has been established to be the identity, and g is unitary, it follows that g must be of the form

$$g = \begin{pmatrix} \text{id} & 0 \\ 0 & h \end{pmatrix}$$

for some unitary $h: \text{im}(f') \cap G' \rightarrow \text{im}(f) \cap G$. Next, we lift h to a map $k: G' \rightarrow G$. Assume without loss of generality that $\dim(G') \leq \dim(G)$ and pick an isometry $k: G' \rightarrow G$ satisfying $(\text{id}_B \oplus k) \circ f' = f$. Such a function exists: let $k(x) = h(x)$ whenever $x \in \text{im}(f') \cap G'$ and, for each element $x \in G'$ not in $\text{im}(f')$, choose an element $y \in G$ not in $\text{im}(f)$ and let $k(x) = y$. Since $\dim(G') \leq \dim(G)$ by assumption, the latter choices can be made so that k is an isometry. Then, it is immediate that $f \sim_{L_{\oplus}} f'$ since $k: G' \rightarrow G$ is an isometry acting as their mediator, and we conclude that $[f, G] = [f', G']$. Moreover, if $F(T) = [f, G]$ and $F(S) = [g, G']$ then it's easy to check that $(g \oplus \text{id}_G) \circ f$ is an isometric dilation of $S \circ T$, so F is indeed a functor. For any two contractions T and S , if $F(T) = F(S)$ then there is an isometric dilation f such that $T = \pi \circ f = S$ and, hence, F is faithful. Finally, $\pi_G \circ f$ is a contraction for all $[f, G]$, so F is full. \square

4 Hiding in a partial setting

The previous section considered (co)affine completions with respect to the direct sum. But rig categories have another monoidal structure, the tensor product. Just as we can form the *additive* affine completion $L_{\oplus}(\mathbf{C})$ of a rig category \mathbf{C} , we can also form the *multiplicative* one $L_{\otimes}(\mathbf{C})$: objects are those of \mathbf{C} , morphisms $A \rightarrow B$ are equivalence classes of morphisms $A \rightarrow B \otimes G$ in \mathbf{C} , with identities and composition just as in Definition 1. Where the L_{\oplus} -construction captures *partiality* by allowing part of the *state space* to be hidden, the L_{\otimes} -construction models *discarding* by allowing any *state* to be hidden (partly or fully).

This construction was shown to capture Stinespring dilation in [14], where the authors argued that $L_{\otimes}(\mathbf{Isometry})$ is monoidally equivalent to the category $\mathbf{FHilb}_{\text{CPTP}}$ of finite-dimensional Hilbert spaces and completely positive trace-preserving (CPTP) maps (or *quantum channels*). A drawback is that this does not generally preserve the direct sum, so $L_{\otimes}(\mathbf{C})$ is generally only monoidal (under tensor product) when \mathbf{C} is a rig category (though remnants of the direct sum persist, see [11, Section 4.4]).

Unfortunately, we cannot hope to use the L_{\otimes} -construction to construct the category $\mathbf{FHilb}_{\text{CPTN}}$ of finite-dimensional Hilbert spaces and completely positive trace-nonincreasing (CPTN) maps (or *partial quantum channels*) for a simple reason: the unit I of the tensor product is terminal in $L_{\otimes}(\mathbf{C})$, but unlike $\mathbf{FHilb}_{\text{CPTP}}$, it is not terminal in $\mathbf{FHilb}_{\text{CPTN}}$. The unique map $A \rightarrow I$ in $\mathbf{FHilb}_{\text{CPTP}}$ is given by the trace of a density matrix on A , and it is unique because all density matrices have unit trace. On the other hand, the *subnormalised* density matrices found in $\mathbf{FHilb}_{\text{CPTN}}$ take their traces in the unit interval, so even though there is only one *trace-preserving* map $A \rightarrow I$, there are as many trace-nonincreasing ones as there are real numbers in $[0, 1]$. Another way to say this is that the L_{\otimes} -construction fails to respect *partiality* in $\mathbf{FHilb}_{\text{CPTN}}$ (as also discussed in [10]).

This section generalises the result from [14] to the partial case. To do this, we show a variant of Stinespring dilation for completely positive trace-nonincreasing maps, describe a construction $L'_{\otimes}(\mathbf{C})$ that extends \mathbf{C} with unique *total* deletion maps, and relate the two by showing that $L'_{\otimes}(\mathbf{Contraction}) \simeq \mathbf{FHilb}_{\text{CPTN}}$. We relate this to the total case by showing that $L'_{\otimes}(\mathbf{C})$ has a universal property as a certain pushout in the category of (locally small) monoidal categories.

4.1 Stinespring dilation for partial quantum channels

We begin with a small lemma, which turns out to be incredibly useful when working with contractions.

Proposition 14. *T is contractive if and only if $T^{\dagger}T \leq 1$.*

Proof. By definition, $T^{\dagger}T \leq 1$ iff $1 - T^{\dagger}T$ is positive semidefinite, which in turn is the case iff $\langle \phi | (1 - T^{\dagger}T) | \phi \rangle \geq 0$ for all $|\phi\rangle$. But since

$$\langle \phi | (1 - T^{\dagger}T) | \phi \rangle = \langle \phi | \phi \rangle - \langle \phi | T^{\dagger}T | \phi \rangle = \|\phi\|^2 - \|T|\phi\rangle\|^2,$$

$1 - T^{\dagger}T$ is positive semidefinite iff $\|\phi\|^2 - \|T|\phi\rangle\|^2 \geq 0$ for all $|\phi\rangle$, *i.e.*, when T is contractive. \square

The previous proposition links handily to the following theorem about the Kraus representation of completely positive trace-nonincreasing maps.

Proposition 15 ([1]). *Any CPTN map Φ admits a representation $\Phi(\rho) = \sum_{i=1}^k M_i \rho M_i^{\dagger}$ with $\sum_i M_i^{\dagger} M_i \leq 1$.*

As in the trace-preserving case, we can construct a Stinespring dilation from the Kraus representation.

Proposition 16. *Every CPTN map Φ admits a Stinespring dilation $\Phi(\rho) = \text{tr}_E(T\rho T^{\dagger})$ for some contraction T and Hilbert space E .*

Proof. Let $\mathcal{B}(A) \xrightarrow{\Phi} \mathcal{B}(B)$ be a partial quantum channel with Kraus representation $\Phi(\rho) = \sum_{i=1}^k M_i \rho M_i^\dagger$. Define $E = \mathbb{C}^k$, let $A \otimes \mathbb{C} \xrightarrow{U_R} A$ denote the right unitor, and let $T = (\sum_{i=1}^k M_i \otimes |i\rangle) U_R^\dagger$. Now for any ρ :

$$\begin{aligned} \text{tr}_E(T\rho T^\dagger) &= \text{tr}_E \left(\left(\sum_{i=1}^k M_i \otimes |i\rangle \right) U_R^\dagger \rho U_R \left(\sum_{j=1}^k M_j^\dagger \otimes \langle j| \right) \right) = \text{tr}_E \left(\sum_{i=1}^k \sum_{j=1}^k M_i \rho M_j^\dagger \otimes |i\rangle \langle j| \right) \\ &= \sum_{i=1}^k \sum_{j=1}^k \text{tr}(|i\rangle \langle j|) M_i \rho M_j^\dagger = \sum_{i=1}^k M_i \rho M_i^\dagger = \Phi(\rho) . \end{aligned}$$

It remains to show that T is contractive:

$$\begin{aligned} T^\dagger T &= U_R \left(\sum_{j=1}^k M_j^\dagger \otimes \langle j| \right) \left(\sum_{i=1}^k M_i \otimes |i\rangle \right) U_R^\dagger = U_R \left(\sum_{j=1}^k \sum_{i=1}^k M_j^\dagger M_i \otimes \langle j|i\rangle \right) U_R^\dagger \\ &= U_R \left(\sum_{i=1}^k M_i^\dagger M_i \otimes \langle i|i\rangle \right) U_R^\dagger = \sum_{i=1}^k M_i^\dagger M_i \leq 1 \end{aligned}$$

which finishes the proof. \square

This representation is essentially unique, *i.e.*, unique up to an isometry applied to the ancilla.

Lemma 17 ([21, Theorem 8.2]). *Let Φ be a CP map with two Kraus representations $\Phi(\rho) = \sum_{i=1}^k E_i \rho E_i^\dagger$ and $\Phi(\rho) = \sum_{j=1}^{k'} F_j \rho F_j^\dagger$. Assume $k = k'$; if $k < k'$, add some $E_i = 0$ for all $k < i \leq k'$. There is a unitary k -by- k matrix $U = (u_{ij})$ such that $E_i = \sum_{j=1}^k u_{ij} F_j$ for all i .*

Proposition 18. *Let Φ be a CPTN map, and let $T_E: A \rightarrow B \otimes G$, $T_F: A \rightarrow B \otimes G'$ be two contractions such that $\Phi(\rho) = \text{tr}_G(T_E \rho T_E^\dagger) = \text{tr}_{G'}(T_F \rho T_F^\dagger)$. If $\dim(G) \leq \dim(G')$, then there is an isometry $W: G \rightarrow G'$ such that $T_F = (1 \otimes W)T_E$.*

Proof. Write U_R for the right unitor. Fix an orthonormal basis for G and define $E_i = U_R(1 \otimes |i\rangle)T_E$; do the same for $F_j = U_R(1 \otimes |j\rangle)T_F$. Notice that $T_E = (\sum_i E_i \otimes |i\rangle)U_R^\dagger$ thanks to $\sum_i |i\rangle \langle i| = 1_G$ for an orthonormal basis. Take V to be an injection of G into a Hilbert space with the same dimension as G' . Let $U = (u_{ij})$ be the unitary from the lemma above. Then:

$$T_E = \left(\sum_i E_i \otimes |i\rangle \right) U_R^\dagger = \left(\sum_i \left(\sum_j u_{ij} F_j \right) \otimes |i\rangle \right) U_R^\dagger = \left(\sum_i \left(\sum_j u_{ij} F_j \otimes |i\rangle \right) \right) U_R^\dagger = \left(\sum_j F_j \otimes \left(\sum_i u_{ij} |i\rangle \right) \right) U_R^\dagger .$$

Because U is unitary, $|\hat{j}\rangle = \sum_i u_{ij} |i\rangle$ form an orthonormal basis for G' . Finally, we obtain the isometry $W: G \rightarrow G'$ by composing $W = RUV$ where R is the unitary mapping $|\hat{j}\rangle \mapsto |j\rangle$. \square

Moreover, as in the trace-preserving case, contractions give rise to CPTN maps through conjugation.

Proposition 19. *Every contraction T gives rise to a CPTN map $\Phi(\rho) = T\rho T^\dagger$.*

Proof. Conjugation by any linear map is completely positive. That Φ is trace-nonincreasing follows by $\text{tr}(T\rho T^\dagger) = \text{tr}(\rho T^\dagger T) \leq \text{tr}(\rho 1) = \text{tr}(\rho)$. \square

Taken together, these results show that the situation between **Contraction** and **FHilb**_{CPTN} mirrors that between **Isometry** and **FHilb**_{CPTP}: there is a (strict monoidal) functor **Contraction** \rightarrow **FHilb**_{CPTN} that sends a contraction T to conjugation by T , and every CPTN map can be expressed this way (in an essentially unique way) if we allow ourselves an ancilla system.

4.2 Total eclipse of the state

We now formulate the new L_{\otimes}^t -construction, adding a notion of hiding that cooperates with the *total* maps in a monoidal dagger category. Consider a monoidal dagger category \mathbf{C} . We can think of its dagger monomorphisms (*i.e.*, morphisms f satisfying $f^\dagger \circ f = \text{id}$) as its *total maps*. These form a subcategory of \mathbf{C} , which we denote $\text{DagMon}(\mathbf{C})$. The significance of the total maps is that, instead of being able to mediate with *arbitrary* maps from \mathbf{C} (as in the L_{\otimes} -construction), we are only permitted to mediate with *total* maps (*i.e.*, morphisms in $\text{DagMon}(\mathbf{C})$) in the L_{\otimes}^t -construction. Explicitly, for $f: A \rightarrow B \otimes G$ and $f': A \rightarrow B \otimes G'$, write $f \leq_{L_{\otimes}^t} f'$ iff there exists a dagger monomorphism $m: G \rightarrow G'$ making

$$\begin{array}{ccc} & A & \\ f \swarrow & & \searrow f' \\ B \otimes G & \xrightarrow{\text{id} \otimes m} & B \otimes G' \end{array}$$

commute. Let $\sim_{L_{\otimes}^t}$ denote the equivalence closure of $\leq_{L_{\otimes}^t}$.

Definition 20. The *partial multiplicative affine completion* $L_{\otimes}^t(\mathbf{C})$ of a symmetric monoidal dagger category \mathbf{C} is the category whose

- objects are those of \mathbf{C} ,
- morphisms $[f, G]: A \rightarrow B$ are pairs of an object G and an equivalence class of morphisms $f: A \rightarrow B \otimes G$ of \mathbf{C} under $\sim_{L_{\otimes}^t}$,
- identities are $[\rho_{\otimes}^{-1}, I]$, and
- composition of $[f, G]: A \rightarrow B$ and $[g, G']: B \rightarrow C$ is $[\alpha_{\otimes} \circ g \otimes \text{id}_G \circ f, G' \otimes G]$.

Before proceeding with the universal property of this construction, we first establish that it succeeds in constructing $\mathbf{FHilb}_{\text{CPTN}}$.

Theorem 21. *There is an equivalence $L_{\otimes}^t(\mathbf{Contraction}) \simeq \mathbf{FHilb}_{\text{CPTN}}$ of monoidal categories.*

Proof. Construct a functor $L_{\otimes}^t(\mathbf{Contraction}) \rightarrow \mathbf{FHilb}_{\text{CPTN}}$ acting as the identity on objects, by sending $[f, G]: A \rightarrow B$ to the map $\rho \mapsto \text{tr}_G(f^\dagger \rho f)$, which is CPTN by Proposition 19 and since the partial trace is trace-preserving. This is well-defined since dagger monomorphisms in $\mathbf{Contraction}$ are precisely the isometries, and since Stinespring dilations are invariant under isometric manipulation of the ancilla system G . This functor is essentially surjective since it is identity on objects and $L_{\otimes}^t(\mathbf{Contraction})$ and $\mathbf{FHilb}_{\text{CPTN}}$ have the same objects; it is full since every CPTN map admits a Stinespring dilation (by Proposition 16); it is faithful since different Stinespring dilations of the same CPTN map are always connected by an isometry on the ancilla by Proposition 18; and it is (strict) monoidal since it preserves coherence isomorphisms. \square

An immediate consequence of the definition of $L_{\otimes}^t(\mathbf{C})$ is that each object comes equipped with a discarding map and chosen projections.

Proposition 22. *Every object A of $L_{\otimes}^t(\mathbf{C})$ has a discarding map $\dagger: A \rightarrow I$, giving canonical projections $\pi_1: A \otimes B \rightarrow A$ and $\pi_2: A \otimes B \rightarrow B$.*

Proof. As in L_{\oplus} , construct $\dagger: A \rightarrow I$ as the equivalence class of the pair $[\lambda_{\otimes}^{-1}, A]$ where $\lambda_{\otimes}^{-1}: A \rightarrow I \otimes A$ is the inverse left unitor of \mathbf{C} . Projections are given by the equivalence class of $[\text{id}_{A \otimes B}, B]: A \otimes B \rightarrow A$ and that of $[\sigma_{\otimes}, A]: A \otimes B \rightarrow B$. \square

We state some basic properties of this construction, shown analogously to those for the L_{\oplus} -construction.

Proposition 23. *When \mathbf{C} is a symmetric monoidal dagger category, $L_{\otimes}^t(\mathbf{C})$ is symmetric monoidal.*

Proposition 24. *There is a strict monoidal functor $\mathcal{E}_t: \mathbf{C} \rightarrow L_{\otimes}^t(\mathbf{C})$ given by $\mathcal{E}_t(A) = A$ on objects and $\mathcal{E}_t(f) = [\rho_{\otimes}^{-1} \circ f, I]$ on morphisms.*

The connection between the L_{\otimes} and the L_{\otimes}^t construction is made clear by the following inclusion.

Proposition 25. *There is a strict monoidal inclusion functor $I_t: L_{\otimes}(\text{DagMon}(\mathbf{C})) \rightarrow L_{\otimes}^t(\mathbf{C})$.*

Indeed, as we will see momentarily, the L_{\otimes}^t -construction is characterised by this inclusion. Interestingly, this construction can be seen as a particular instance of the *monoidal indeterminates*-construction [9]. This gives a very useful factorisation lemma, which may be regarded as an instance of *purification*.

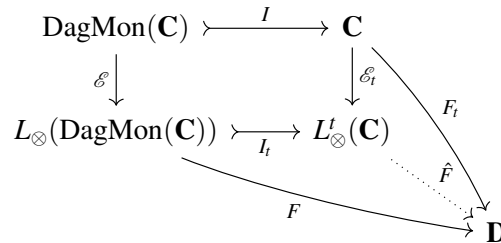
Lemma 26. *Let \mathbf{C} be a symmetric monoidal dagger category. Then*

- (i) *morphisms $[f, E]$ of $L_{\otimes}(\text{DagMon}(\mathbf{C}))$ factor uniquely as $\pi_1 \circ \mathcal{E}(f')$ for some f' in $\text{DagMon}(\mathbf{C})$,*
- (ii) *morphisms $[f, E]$ of $L_{\otimes}^t(\mathbf{C})$ factor uniquely as $\pi_1 \circ \mathcal{E}_t(f')$ for some f' in \mathbf{C} , and*
- (iii) *morphisms $[f, E]$ of $L_{\otimes}^t(\mathbf{C})$ factor uniquely as $I_t(\pi_1) \circ \mathcal{E}(f')$ for some f' in \mathbf{C} .*

Proof. (i,ii) are the expansion-raw factorisation of [9], and (iii) holds as $\pi_1 = I_t(\pi_1)$ in $L_{\otimes}^t(\mathbf{C})$. □

We are now ready to establish the universal property of $L_{\otimes}^t(\mathbf{C})$.

Theorem 27. *$L_{\otimes}^t(\mathbf{C})$ is a pushout of $\mathcal{E}: \text{DagMon}(\mathbf{C}) \rightarrow L_{\otimes}(\text{DagMon}(\mathbf{C}))$ along the inclusion functor $\text{DagMon}(\mathbf{C}) \rightarrow \mathbf{C}$ in the category of locally small symmetric monoidal categories and strong monoidal functors.*

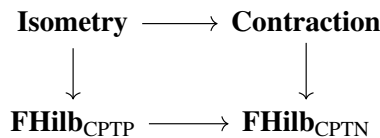


Proof sketch. Notice first that the upper square commutes since all functors involved are strict monoidal, and I and I_t are merely inclusions behaving as the identity on objects and morphisms, while \mathcal{E} is precisely \mathcal{E}_t restricted to dagger monomorphisms of \mathbf{C} .

Next, since objects on \mathbf{C} , $\text{DagMon}(\mathbf{C})$, $L_{\otimes}^t(\mathbf{C})$, and $L_{\otimes}(\text{DagMon}(\mathbf{C}))$ all coincide, F and F_t must agree on objects, so we may define $\hat{F}(X) = F(X) = F_t(X)$ on objects, and $\hat{F} \circ I_t = F$ and $\hat{F} \circ \mathcal{E}_t = F_t$ on objects follows immediately. On morphisms we define $\hat{F}([f, E]) = F(\pi_1) \circ F_t(f)$. That this definition satisfies $\hat{F} \circ I_t = F$ and $\hat{F} \circ \mathcal{E}_t = F_t$ on morphisms as well, and that \hat{F} is unique, follows using the factorisation lemma above (full proof in the appendix). □

Instantiating this property to the case of **Contraction**, where the functors **Contraction** \rightarrow **Isometry** and **FHilb**_{CPTP} \rightarrow **FHilb**_{CPTN} are inclusions, and the functors **Isometry** \rightarrow **FHilb**_{CPTP} and **Contraction** \rightarrow **FHilb**_{CPTN} conjugating by an isometry or contraction, we obtain the following characterisation.

Corollary 28. *The following square is a pushout of symmetric monoidal categories:*



Proof. That $\text{DagMon}(\mathbf{Contraction}) \simeq \mathbf{Isometry}$ follows by the fact that every isometry f is contractive (it satisfies $\|f(x)\| = \|x\|$, so specifically $\|f(x)\| \leq \|x\|$) and satisfies $f^\dagger \circ f = \text{id}$. Finally, $L_\otimes(\mathbf{Isometry}) \simeq \mathbf{FHilb}_{\text{CPTN}}$ is known [14], and $L_\otimes(\mathbf{Contraction}) \simeq \mathbf{FHilb}_{\text{CPTN}}$ by Theorem 21. \square

5 Splitting measurements

The category $\mathbf{FHilb}_{\text{CPTN}}$ does not have coproducts, but it can nevertheless be instructive to see why a given candidate for cotupling fails. Working in $L_\otimes^t(\mathbf{Contraction})$, given two Stinespring dilations $f: A \rightarrow C \otimes G$ and $g: B \rightarrow C \otimes G'$, a candidate for the Stinespring dilation of their cotupling is the map $\delta_L^{-1} \circ f \oplus g: A \oplus B \rightarrow C \otimes (G \oplus G')$. In particular, if we were to try to cotuple the trivial Stinespring dilations of the canonical injections $i_1: A \rightarrow (A \oplus B) \otimes I$ and $i_2: B \rightarrow (A \oplus B) \otimes I$ in $\mathbf{Contraction}$, the result would be the map $\delta_L^{-1} \circ i_1 \oplus i_2: A \oplus B \rightarrow (A \oplus B) \otimes (I \oplus I)$. Notice the non-trivial nature of the ancilla of this dilation, which tracks whether the result came from i_1 or i_2 . This is not simply the identity, as it should be if this were an actual cotupling. Computing, we see that this map acts on block diagonal density matrices on $A \oplus B$ by measuring whether the result falls in A or in B , *i.e.*, by the mapping

$$\left(\begin{array}{c|c} X & Y \\ \hline Z & W \end{array} \right) \mapsto \left(\begin{array}{c|c} X & 0 \\ \hline 0 & W \end{array} \right).$$

Clearly, maps e such as these are both idempotent and *causal* (in that they satisfy $\dagger \circ e = \dagger$), but interestingly they do not split in $\mathbf{FHilb}_{\text{CPTN}}$. However, they *do* have a very natural splitting in the category $\mathbf{FCstar}_{\text{CPTN}}$ of finite-dimensional C^* -algebras and CPTN maps, via $m: \mathcal{B}(A \oplus B) \rightarrow \mathcal{B}(A) \oplus \mathcal{B}(B)$ and $p: \mathcal{B}(A) \oplus \mathcal{B}(B) \rightarrow \mathcal{B}(A \oplus B)$ given by

$$m \left(\begin{array}{c|c} X & Y \\ \hline Z & W \end{array} \right) = (X, W) \quad \text{and} \quad p(X, W) = \left(\begin{array}{c|c} X & 0 \\ \hline 0 & W \end{array} \right) \quad (1)$$

as we then have $m \circ p = \text{id}$ and $p \circ m = e$. Indeed, as argued in [3], this is a defining characteristic of finite-dimensional C^* -algebras compared to Hilbert spaces.

Definition 29 ([3]). The idempotent splitting of causal idempotents in \mathbf{C} (where \mathbf{C} is a category with discarding) is the category $\mathbf{Split}^\dagger(\mathbf{C})$ whose

- objects are pairs (A, e) of an object A of \mathbf{C} and an idempotent $e: A \rightarrow A$ satisfying $\dagger \circ e = \dagger$,
- morphisms $f: (A, e) \rightarrow (B, e')$ are morphisms $f: A \rightarrow B$ of \mathbf{C} satisfying $e' \circ f \circ e = f$,
- each identity $(A, e) \rightarrow (A, e)$ is e , and
- composition is as in \mathbf{C} .

There is an inclusion of \mathbf{C} in $\mathbf{Split}^\dagger(\mathbf{C})$ sending objects A to (A, id) and leaving morphisms unchanged. This construction is well-known to be the free splitting of these idempotents; more abstractly, it is the completion of a category with regards to certain absolute colimits [23]. That this constructs finite dimensional C^* -algebras out of Hilbert spaces follows by [3]:

Proposition 30. *There is a (monoidal, causal) equivalence $\mathbf{Split}^\dagger(\mathbf{FHilb}_{\text{CPTN}}) \simeq \mathbf{FCstar}_{\text{CPTN}}$.*

Proof. It suffices to argue that the objects of $\mathbf{Split}^\dagger(\mathbf{FHilb}_{\text{CPTN}})$ are finite-dimensional C^* -algebras, which is immediate by Corollary 3.4 of [3]. \square

6 Discussion

We have presented a universal construction that constructs the category of finite-dimensional C*-algebras and completely positive trace-nonincreasing maps from the category of finite-dimensional Hilbert spaces and unitaries. Though we have kept to the finite-dimensional case in this paper, there is reason to suspect that many of these results will generalise to infinite dimensions. For example, *Halmos dilation* has a far stronger statement as Sz. Nagy dilation in the infinite-dimensional case, and the usual Stinespring dilation theorem generalises to the infinite-dimensional case as well.

The key application that we envision for this work is in the design and semantics of quantum programming languages. One application of the LR_{\oplus} -construction is in the quantisation of reversible classical programs. It can be shown that applying the LR_{\oplus} -construction to the category **FinBij** of finite sets and bijections yields the category **FinPInj** of finite sets and partial injective functions. In particular, this means that the quantisation functor **FinBij** \rightarrow **Unitary** lifts uniquely to a functor **FinPInj** \rightarrow **Contraction**. This is interesting since **FinPInj** is the setting for (finite) reversible classical computing (see e.g. [18, 6, 7]), in particular *reversible (classical) flowcharts* [32, 5] (with a finite state space). This suggests that **Contraction** may be similarly considered as a setting for *reversible quantum flowcharts*, a kind of quantum flowcharts (see also [26]) which eschew measurements in favour of quantum control.

Another application concerns extending the quantum programming language $\mathcal{U}\Pi_a^{\chi}$ (“yuppie-chia”) [11] with *classical types*. $\mathcal{U}\Pi_a^{\chi}$ is an effectful extension of $\mathcal{U}\Pi$, a quantum extension to the strongly typed classical reversible programming language Π [17]. The effectful part of $\mathcal{U}\Pi_a^{\chi}$ is that it uses the L_{\otimes} and R_{\oplus} constructions in a very direct way to add measurement as an effect to an otherwise measurement-free language, taking its semantics from **Unitary** to **FHilb**_{CPTN}. Since the causal idempotents described in Section 5 can all be described as $\mathcal{U}\Pi_a^{\chi}$ programs, the **Split**[±]-construction can similarly be used very directly to add classical types to this language as a computational effect. This addresses a known shortcoming of the language by allowing a distinction between e.g. the type of *bits* and the type of *qubits* (as in e.g. [26]), but also allows for a much more fine-grained type-level separation of measurement maps in *how much* they measure.

Acknowledgements We are indebted to the anonymous reviewers for their comments and suggestions, to John van der Wetering for a lively discussion at the event, and to Sean Tull for discussions and pointing out the equivalence between splitting measurements and splitting causal idempotents in **FHilb**_{CPTN}.

References

- [1] V. Cappellini, H.-J. Sommers & K. Życzkowski (2007): *Subnormalized states and trace-nonincreasing maps*. *Journal of Mathematical Physics* 48(5), p. 052110, doi:10.1063/1.2738359.
- [2] K. Cho (2015): *Total and Partial Computation in Categorical Quantum Foundations*. In C. Heunen, P. Selinger & J. Vicary, editors: *Proceedings 12th International Workshop on Quantum Physics and Logic (QPL 2015)*, EPTCS 195, pp. 116–135, doi:10.4204/EPTCS.195.9.
- [3] B. Coecke, J. Selby & S. Tull (2017): *Two Roads to Classically*. In: *Proceedings of the 14th International Workshop on Quantum Physics and Logic (QPL 2017)*, Open Publishing Association, pp. 104–118, doi:10.4204/EPTCS.266.7.
- [4] C. Comfort (2021): *The ZX& calculus: A complete graphical calculus for classical circuits using spiders*. In B. Valiron, S. Mansfield, P. Arrighi & P. Panangaden, editors: *Proceedings 17th International Conference on Quantum Physics and Logic (QPL 2020)*, Electronic Proceedings in Theoretical Computer Science 340, OPA, pp. 60–90, doi:10.4204/EPTCS.340.4.

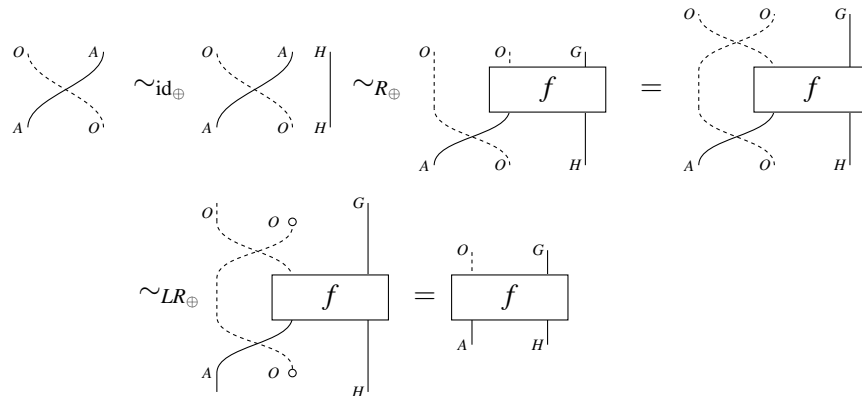
- [5] R. Glück & R. Kaarsgaard (2018): *A categorical foundation for structured reversible flowchart languages: Soundness and adequacy*. *Logical Methods in Computer Science* 14(3:16), pp. 1–38, doi:10.23638/LMCS-14(3:16)2018.
- [6] R. Glück, R. Kaarsgaard & T. Yokoyama (2019): *Reversible programs have reversible semantics*. In: *International Symposium on Formal Methods: FM 2019 International Workshops*, Springer, pp. 413–427, doi:10.1007/978-3-030-54997-8_26.
- [7] R. Glück, R. Kaarsgaard & T. Yokoyama (2022): *From Reversible Programming Languages to Reversible Metalanguages*. *Theoretical Computer Science* 920, pp. 46–63, doi:10.1016/j.tcs.2022.02.024.
- [8] P. R. Halmos (1950): *Normal Dilations and Extensions of Operators*. *Summa Brasiliensis Mathematicae* 2, pp. 125–134, doi:10.1007/978-1-4613-8208-9_9.
- [9] C. Hermida & R. D. Tennent (2012): *Monoidal indeterminates and categories of possible worlds*. *Theoretical Computer Science* 430, pp. 3–22, doi:10.1016/j.tcs.2012.01.001.
- [10] C. Heunen & R. Kaarsgaard (2021): *Bennett and Stinespring, Together at Last*. In C. Heunen & M. Backens, editors: *Proceedings 18th International Conference on Quantum Physics and Logic (QPL 2021)*, *Electronic Proceedings in Theoretical Computer Science* 343, OPA, pp. 102–118, doi:10.4204/EPTCS.343.5.
- [11] C. Heunen & R. Kaarsgaard (2022): *Quantum Information Effects*. *Proceedings of the ACM on Programming Languages* 6(POPL), doi:10.1145/3498663.
- [12] C. Heunen, A. Kissinger & P. Selinger (2014): *Completely positive projections and biproducts*. In: *Proceedings of the 10th International Workshop on Quantum Physics and Logic (QPL 2013)*, Open Publishing Association, pp. 71–83, doi:10.4204/EPTCS.171.7.
- [13] C. Heunen & J. Vicary (2019): *Categories for Quantum Theory*. Oxford University Press, doi:10.1093/oso/9780198739623.001.0001.
- [14] M. Huot & S. Staton (2018): *Universal properties in quantum theory*. In P. Selinger & G. Chiribella, editors: *Proceedings of the 15th International Conference on Quantum Physics and Logic (QPL 2018)*, *Electronic Proceedings in Theoretical Computer Science* 287, Open Publishing Association, pp. 213–224, doi:10.4204/EPTCS.287.12.
- [15] M. Huot & S. Staton (2019): *Quantum channels as a categorical completion*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019)*, IEEE, pp. 1–13, doi:10.1109/LICS.2019.8785700.
- [16] G. Hutton (1999): *A tutorial on the universality and expressiveness of fold*. *Journal of Functional Programming* 9(4), pp. 355–372, doi:10.1017/S0956796899003500.
- [17] R. P. James & A. Sabry (2012): *Information Effects*. In: *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of programming languages (POPL '12)*, ACM, pp. 73–84, doi:10.1145/2103656.2103667.
- [18] R. Kaarsgaard, H. B. Axelsen & R. Glück (2017): *Join inverse categories and reversible recursion*. *Journal of Logical and Algebraic Methods in Programming* 87, pp. 33–50, doi:10.1016/j.jlamp.2016.08.003.
- [19] M. Laplaza (1972): *Coherence for Distributivity*. *Lecture Notes in Mathematics* 281, pp. 29–72, doi:10.1007/BFb0059555.
- [20] E. Levy & O. Shalit (2014): *Dilation theory in finite dimensions: The possible, the impossible, and the unknown*. *Rocky Mountain Journal of Mathematics* 44(1), pp. 203–221, doi:10.1216/RMJ-2014-44-1-203.
- [21] M. A. Nielsen & I. L. Chuang (2010): *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, doi:10.1017/CB09780511976667.
- [22] A. J. Parzygnat (2019): *Stinespring's construction as an adjunction*. *Compositionality* 1(2), pp. 1–42, doi:10.32408/compositionality-1-2.
- [23] R. Paré (1971): *On absolute colimits*. *Journal of Algebra* 19(1), pp. 80–95, doi:10.1016/0021-8693(71)90116-5.

- [24] M. Rennela, S. Staton & R. Furber (2017): *Infinite-dimensionality in quantum foundations: W^* -algebras as presheaves over matrix algebras*. In: *13th International Conference on Quantum Physics and Logic (QPL 2017)*, Open Publishing Association, pp. 161–173, doi:10.4204/EPTCS.236.11.
- [25] P. L. Robinson (2018): *Julia operators and Halmos dilations*. arXiv preprint arXiv:1803.09329, doi:10.48550/arXiv.1803.09329.
- [26] P. Selinger (2004): *Towards a Quantum Programming Language*. *Mathematical Structures in Computer Science* 14(4), pp. 527–586, doi:10.1017/S0960129504004256.
- [27] S. Staton & P. B. Levy (2013): *Universal properties of impure programming languages*. In: *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '13)*, ACM, pp. 179–192, doi:10.1145/2429069.2429091.
- [28] D. Stein & S. Staton (2021): *Compositional semantics for probabilistic programs with exact conditioning*. In: *36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2021)*, IEEE, pp. 1–13, doi:10.1109/LICS52264.2021.9470552.
- [29] W. F. Stinespring (1955): *Positive functions on C^* -algebras*. *Proceedings of the American Mathematical Society* 6(2), pp. 211–216, doi:10.2307/2032342.
- [30] P. Wadler (1989): *Theorems for free!* In: *Proceedings of the fourth international conference on Functional programming languages and computer architecture (FPCA '89)*, ACM, pp. 347–359, doi:10.1145/99370.99404.
- [31] A. Westerbaan & B. Westerbaan (2016): *Paschke Dilations*. In R. Duncan & C. Heunen, editors: *Proceedings of the 13th International Conference on Quantum Physics and Logic (QPL 2016)*, *Electronic Proceedings in Theoretical Computer Science* 236, Open Publishing Association, pp. 229–244, doi:10.4204/EPTCS.236.15.
- [32] T. Yokoyama, H. B. Axelsen & R. Glück (2016): *Fundamentals of reversible flowchart languages*. *Theoretical Computer Science* 611, pp. 87–115, doi:10.1016/j.tcs.2015.07.046.

A Deferred proofs

Proposition 9. The additive unit O is a zero object in $LR_{\oplus}(\mathbf{C})$.

Proof. Define maps $A \rightarrow O$ and $O \rightarrow A$ as the equivalence classes of the symmetry $\sigma_{\oplus} : A \oplus O \rightarrow O \oplus A$ and $\sigma_{\oplus} : O \oplus A \rightarrow A \oplus O$. To see that the map $A \rightarrow O$ is unique, let $f : A \oplus H \rightarrow O \oplus G$ be any other morphism of \mathbf{C} . Then $[O, \sigma, A] \sim_{LR_{\oplus}} [H, f, G]$ can be seen using the graphical language [13]:



That the map $O \rightarrow A$ is unique follows analogously, so O is both initial and terminal, *i.e.*, a zero object. □

Proposition 10. When \mathbf{C} is a dagger rig category, so is $LR_{\oplus}(\mathbf{C})$.

Proof. For any morphism $[H, f, G]: A \rightarrow B$ in $LR_{\oplus}(\mathbf{C})$, its dagger is defined to be

$$[H, f, G]^{\dagger} = [G, f^{\dagger}, H].$$

First, we need to check this is well-defined, *i.e.*, given $[H, f, G] = [H', f', G']$, verify $[H, f, G]^{\dagger} = [H', f', G']^{\dagger}$. This is straightforward, as $f \sim_{LR_{\oplus}} g$ implies $f^{\dagger} \sim_{LR_{\oplus}} g^{\dagger}$; to see this, notice that

$$\begin{aligned} f \sim_{L_{\oplus}} g &\implies f^{\dagger} \sim_{R_{\oplus}} g^{\dagger} \\ f \sim_{R_{\oplus}} g &\implies f^{\dagger} \sim_{L_{\oplus}} g^{\dagger} \end{aligned}$$

by using the dagger of the mediators and, furthermore, $f \sim_{\text{id}_{\oplus}} g$ trivially implies $f^{\dagger} \sim_{\text{id}_{\oplus}} g^{\dagger}$. Clearly, this dagger is involutive and

$$\text{id}_A^{\dagger} = [O, \text{id}_{A \oplus O}, O]^{\dagger} = [O, \text{id}_{A \oplus O}^{\dagger}, O] = \text{id}_A.$$

Checking explicitly that $(g \circ f)^{\dagger} = f^{\dagger} \circ g^{\dagger}$ is more involved, but conceptually trivial: flip the diagram and use the fact that \mathbf{C} is a dagger category. This works because, in $LR_{\oplus}(\mathbf{C})$ there is symmetry between input and output: the dagger turns hidden parts of the input into a hidden part of the output, and vice versa. That coherence isomorphisms are unitary follows immediately by the fact that they are inherited from \mathbf{C} , where they are unitary by \mathbf{C} a dagger rig category. \square

Proposition 12. When \mathbf{C} is a rig category, $L_{\oplus}(R_{\oplus}(\mathbf{C})) \cong LR_{\oplus}(\mathbf{C}) \cong R_{\oplus}(L_{\oplus}(\mathbf{C}))$.

Proof. The fact that O is initial in $LR_{\oplus}(\mathbf{C})$ together with the universal property of R_{\oplus} implies that the top triangle of the diagram below commutes, for a unique functor Φ . Then, the fact that O is also terminal in $LR_{\oplus}(\mathbf{C})$ and the universal property of L_{\oplus} implies that the bottom triangle also commutes, for a unique functor \mathcal{F} .

$$\begin{array}{ccc} \mathbf{C} & \xrightarrow{\mathcal{D}} & R_{\oplus}(\mathbf{C}) \\ \mathcal{F} \downarrow & \swarrow \Phi & \downarrow \mathcal{E} \\ LR_{\oplus}(\mathbf{C}) & \xleftarrow{\widehat{\mathcal{F}}} & L_{\oplus}(R_{\oplus}(\mathbf{C})) \end{array}$$

Moreover, O is a zero object in $L_{\oplus}(R_{\oplus}(\mathbf{C}))$ and, thus, the universal property of LR_{\oplus} implies there is a unique strong monoidal functor $\widehat{\mathcal{E}} \circ \widehat{\mathcal{D}}: LR_{\oplus}(\mathbf{C}) \rightarrow L_{\oplus}(R_{\oplus}(\mathbf{C}))$ such that $\mathcal{E} \circ \mathcal{D} = \widehat{\mathcal{E}} \circ \widehat{\mathcal{D}} \circ \mathcal{F}$. This, together with the commuting diagram above implies:

$$\widehat{\mathcal{E}} \circ \widehat{\mathcal{D}} \circ \widehat{\mathcal{F}} \circ \mathcal{E} \circ \mathcal{D} = \widehat{\mathcal{E}} \circ \widehat{\mathcal{D}} \circ \Phi \circ \mathcal{D} = \widehat{\mathcal{E}} \circ \widehat{\mathcal{D}} \circ \mathcal{F} = \mathcal{E} \circ \mathcal{D}$$

which is captured in the outer triangle of the diagram below

$$\begin{array}{ccccc} \mathbf{C} & \xrightarrow{\mathcal{D}} & R_{\oplus}(\mathbf{C}) & \xrightarrow{\mathcal{E}} & L_{\oplus}(R_{\oplus}(\mathbf{C})) \\ & \searrow \mathcal{E} \circ \mathcal{D} & \swarrow & \downarrow \widehat{\mathcal{E}} \circ \widehat{\mathcal{D}} \circ \widehat{\mathcal{F}} & \\ & & & & L_{\oplus}(R_{\oplus}(\mathbf{C})) \end{array}$$

Due to the universal property of R_{\oplus} , the functor shown above as a diagonal dashed line is unique, so it must be \mathcal{E} ; then, $\widehat{\mathcal{E} \circ \mathcal{D}} \circ \widehat{\mathcal{F}} = 1_{L_{\oplus}(R_{\oplus}(\mathbf{C}))}$ as both functors make the inner right triangle commute but, by the universal property of L_{\oplus} , there is only one such functor. By a similar argument $\widehat{\mathcal{F}} \circ \widehat{\mathcal{E} \circ \mathcal{D}} = 1_{LR_{\oplus}(\mathbf{C})}$ and hence $L_{\oplus}(R_{\oplus}(\mathbf{C})) \cong LR_{\oplus}(\mathbf{C})$. This is an equivalence of rig categories, as $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{E} \circ \mathcal{D}}$ are rig functors by construction. The same strategy proves $R_{\oplus}(L_{\oplus}(\mathbf{C})) \cong LR_{\oplus}(\mathbf{C})$. \square

Theorem 27. $L'_{\otimes}(\mathbf{C})$ is a pushout of $\mathcal{E}: \text{DagMon}(\mathbf{C}) \rightarrow L_{\otimes}(\text{DagMon}(\mathbf{C}))$ along the inclusion functor $\text{DagMon}(\mathbf{C}) \rightarrow \mathbf{C}$ in the category of locally small symmetric monoidal categories and strong monoidal functors.

$$\begin{array}{ccc}
 \text{DagMon}(\mathbf{C}) & \xrightarrow{I} & \mathbf{C} \\
 \mathcal{E} \downarrow & & \downarrow \mathcal{E}_t \\
 L_{\otimes}(\text{DagMon}(\mathbf{C})) & \xrightarrow{I_t} & L'_{\otimes}(\mathbf{C}) \\
 & \searrow F & \downarrow \hat{F} \\
 & & \mathbf{D}
 \end{array}$$

(Note: In the original image, there is also a curved arrow F_t from \mathbf{C} to \mathbf{D} and a dotted arrow \hat{F} from $L'_{\otimes}(\mathbf{C})$ to \mathbf{D} .)

Proof. Notice first that the upper square commutes since all functors involved are strict monoidal, and I and I_t are merely inclusions behaving as the identity on objects and morphisms, while \mathcal{E} is precisely \mathcal{E}_t restricted to dagger monomorphisms of \mathbf{C} .

Next, since objects on \mathbf{C} , $\text{DagMon}(\mathbf{C})$, $L'_{\otimes}(\mathbf{C})$, and $L_{\otimes}(\text{DagMon}(\mathbf{C}))$ all coincide, F and F_t must agree on objects, so we may define $\hat{F}(X) = F(X) = F_t(X)$ on objects, and $\hat{F} \circ I_t = F$ and $\hat{F} \circ \mathcal{E}_t = F_t$ on objects follows immediately. On morphisms we define $\hat{F}([f, E]) = F(\pi_1) \circ F_t(f)$. Then

$$\begin{aligned}
 \hat{F}(I_t([f, E])) &= \hat{F}(I_t(\pi_1 \circ \mathcal{E}(f))) = \hat{F}(I_t(\pi_1) \circ I_t(\mathcal{E}(f))) = \hat{F}(I_t(\pi_1) \circ \mathcal{E}_t(I(f))) \\
 &= \hat{F}([I(f), E]) = F(\pi_1) \circ F_t(I(f)) = F(\pi_1) \circ F(\mathcal{E}(f)) \\
 &= F(\pi_1 \circ \mathcal{E}(f)) = F([f, E])
 \end{aligned}$$

so $\hat{F} \circ I_t = F$ on morphisms as well. For the other triangle,

$$\begin{aligned}
 \hat{F}(\mathcal{E}_t(f)) &= \hat{F}(\pi_1 \circ \rho^{-1} \circ \mathcal{E}_t(f)) = \hat{F}(\pi_1 \circ \mathcal{E}_t(\rho^{-1}) \circ \mathcal{E}_t(f)) = \hat{F}(\pi_1 \circ \mathcal{E}_t(\rho^{-1} \circ f)) \\
 &= \hat{F}([\rho^{-1} \circ f, I]) = F(\pi_1) \circ F_t(\rho^{-1} \circ f) = F(\pi_1) \circ F(\rho^{-1}) \circ F_t(f) \\
 &= F(\pi_1) \circ F_t(I(\rho^{-1})) \circ F_t(f) = F(\pi_1) \circ F(\mathcal{E}(\rho^{-1})) \circ F_t(f) \\
 &= F(\pi_1 \circ \mathcal{E}(\rho^{-1})) \circ F_t(f) = F([\rho^{-1}, I]) \circ F_t(f) = F(\text{id}) \circ F_t(f) = F_t(f)
 \end{aligned}$$

establishing $\hat{F} \circ \mathcal{E}_t = F_t$ on morphisms. Finally, suppose G satisfies $G \circ I_t = F$ and $G \circ \mathcal{E}_t = F_t$. Then

$$G([f, E]) = G(I_t(\pi_1) \circ \mathcal{E}_t(f)) = G(I_t(\pi_1)) \circ G(\mathcal{E}_t(f)) = F(\pi_1) \circ F_t(f) = \hat{F}([f, E])$$

on morphisms, and since G and \hat{F} must agree on objects as well (e.g. $\hat{F}(X) = F(X) = G(I_t(X)) = G(X)$), we get $G = \hat{F}$. Finally, \hat{F} is strong monoidal since \mathcal{E}_t is strict monoidal and F_t and \hat{F} agree on objects, so \hat{F} may reuse the coercions $I \cong F_t(I)$ and $F_t(A \otimes B) \cong F_t(A) \otimes F_t(B)$. \square

The Causal Structure of Semantic Ambiguities

Daphne Wang

University College London

Mehrnoosh Sadrzadeh

University College London

Ambiguity is a natural language phenomenon occurring at different levels of syntax, semantics, and pragmatics. It is widely studied; in Psycholinguistics, for instance, we have a variety of competing studies for the human disambiguation processes. These studies are empirical and based on eye-tracking measurements. Here we take first steps towards formalizing these processes for semantic ambiguities where we identified the presence of two features: (1) joint plausibility degrees of different possible interpretations, (2) causal structures according to which certain words play a more substantial role in the processes. The novel sheaf-theoretic model of definite causality developed by Gogioso and Pinzani in QPL 2021 offers tools to model and reason about these features. We applied this theory to a dataset of ambiguous phrases extracted from Psycholinguistics literature and their human plausibility judgements collected by us using the Amazon Mechanical Turk engine. We measured the causal fractions of different disambiguation orders within the phrases and discovered two prominent orders: from subject to verb in the subject-verb and from object to verb in the verb object phrases. We also found evidence for delay in the disambiguation of polysemous vs homonymous verbs, again compatible with Psycholinguistic findings.

1 Introduction

Discovering, studying, and formalising the ambiguities of natural language is an area of the field of Computational Linguistics. Natural language ambiguities occur at different levels of syntax, semantics, and pragmatics. Syntactic ambiguities are due to single words with multiple grammatical roles, e.g. both noun and verb, as in ‘book’ and ‘cook’. They also occur at the phrase level, e.g. in the phrase ‘old men and women’ the adjective ‘old’ can be modifying both the word ‘men’ or the conjunction ‘men and women’. Other examples are ‘Show me the meals on the flight from SF’, and ‘We saw the Eiffel Tower flying to Paris’. Ambiguities surpass sentential boundaries, as in the anaphorically ambiguous discourse ‘I put the CD in the computer. It broke.’. Another prominent type of ambiguity is due to different interpretations of words and phrases. Words/phrases that are ‘homonymous’ have more than one unrelated interpretation, due to historical incidence or other reasons, as in ‘plant’, ‘pitcher’, or ‘coach’. Polysemous words/phrases have more than one related interpretations, as in ‘newspaper’, which can refer to the collection of papers stapled together (literal) or the content conveyed by these (figurative).

The process of semantic disambiguation and the role of the context has been studied in the field of Psycholinguistics via devices such as eye-tracking. Here, the delay in reading and the trajectory of the gaze shows that context plays different roles when disambiguating different types of semantically ambiguous words. These results show that ambiguous verbs get disambiguated late, e.g. after reading the whole sentence, whereas certain ambiguous nouns get disambiguated almost immediately and without much reliance on context. They also show that polysemous verbs are disambiguated later than homonymous verbs, and the same applies for nouns, i.e. that polysemous nouns are disambiguated later than homonymous nouns. Semantic ambiguities are plentiful and universal. They also easily lift from word to phrase level, where the number and complexity of potential interpretations increase exponentially. Assuming only two interpretations per word (which is the minimal criteria for ambiguity; resources such

as WordNet list 78695 senses for a total of 128321 words of English), a 2-word phrase, the simplest ambiguous combination of two ambiguous words, can have up to four different interpretations, a 3-word phrase up to 8, an n -word one up to 2^n . Instances of ambiguous 2-word phrases are ‘(the) cabinet reflects’, ‘(the) pitcher threw’, and ‘(the) plant bored’, where ‘pitcher threw’ has 3 interpretations: a jug throwing a shadow, a baseball player throwing a shadow, or a baseball player throwing a ball. ‘A plant bore’ has 4 interpretations: a factory that makes holes (e.g. in metal bars), a factory that made its workers weary, a house plant that was uninteresting, a house plant that pierced its pot.

Whether word or phrase, the interpretations of ambiguous natural language expressions depends on their contexts and this can be formalised in different ways. We use the fact that interpretations are context-dependent and that this gives rise to context-dependent probability distributions, corresponding to the likelihood that a certain meaning of a word is selected in a context. By context we refer to any linguistic or non-linguistic information, e.g. knowledge-based, resources and background information. This gives rise to two questions that we aim at investigating. (1) Given these resources, can the single distributions of the interpretations of each word within a phrase be used to compute a distribution for the meaning combination of the full phrase? For example, if we see the word “pitcher” in the corpus mostly as a baseball player and “threw” as throwing a ball, when we next come across the phrase “pitcher threw”, can we be sure that it means a baseball player threw a ball and not a jug threw a shadow? (2) Since there is a temporal order in the disambiguation process, is there a causal order in the process, and if so how can it be quantified used to replicate the Psycholinguistics findings? The first question can be formalised in terms of quantum-like contextuality and indeed previous research has been done on whether cognitive processes are contextual in this way, (see for example the work of Bruza et. al. focusing on concept combinations[3] and the “mental lexicon”[2]).

We answered the first question in previous work [14, 13, 12] using the sheaf theoretic model of contextuality of [1] and its generalisation to signalling scenarios in the Contextuality by Default (CbD) setting of [5]. Using these tools, we formalised the first question as: “Is there a global joint probability distribution that describes the probabilistic distributions of phrases where we can maximise the probability distributions of each word within the phrase?”. We hypothesised that, similar to the case in Quantum mechanics, the answer to this question is no, and found a few examples that witnessed it. This led us to the conclusion that the pre-existing value of the interpretation of a word in a phrase is not independent of the interpretations of other words in the context (including the phrase itself).

In this paper, we formalise and answer the second question using recent advances in causal sheaf theory and in particular the development of [8]. Our methodology is as follows. We first devise a dataset consisting of equal numbers of polysemous and homonymous nouns and verbs. The nouns and verbs are trimmed down from a larger such set, with the demarcation rule that both of their subject-verb and verb-object combinations in a phrase would make sense. We put these phrases on Amazon Mechanical Turk and collect human judgements for degrees of plausibility of each phrase. We compute probability distributions from our Amazon Turk human judgements and verify which proportion of the judgements are compatible with one of our four main causal orders: Object/Subject \rightarrow Verb, Verb \rightarrow Object/Subject. We then work within Subject-Verb and Verb-Object phrases and for each phrase type study which of the four causal order was higher than the other: polysemous verbs/nouns or homonymous verbs/nouns. Our findings confirm the Psycholinguistic research, that (1) the prominent causal order of phrases is from the noun to the verb, i.e. from Subject \rightarrow Verb in Subject-Verb phrases and from the Object \rightarrow Verb in the Verb-Object phrase. In other words, the verb is the last part of speech to be disambiguated in a sentence, (2) polysemous verbs are disambiguated later than homonymous verbs, (3) polysemous nouns are disambiguated later than homonymous nouns.

2 The causal framework and the causal fraction

The aim of this section is to introduce the causal framework and the causal fraction therein. In this entry paragraph we briefly review the relevance of these to linguistic scenarios.

The study of quantum contextuality relies on the *no-signalling* property of a given system; i.e. that choices of observables do not influence the outcomes of other observables measured at the same time. However, linguistic scenarios do not necessarily satisfy this property, and neither does it fit with our intuition: no-signalling in our linguistic scenarios would mean that the choice of a word does not influence the interpretation of other words in a phrase. There are mathematical models that generalise the notion of contextuality from no-signalling to all systems. An example is the Contextual-by-Default (CbD) framework [5]. In this framework, however, the source and nature of signalling property is completely disregarded, and the interpretation of what contextuality might say about the system is not clear anymore. In previous work [13, 14, 12], we nonetheless used the CbD framework and analysed ambiguous examples from natural language. Here, instead, we work with the extension of the sheaf-theoretic model of contextuality to causality. This extension was developed in [8]. The use of the causal framework not only enables us to allow for some signalling, but also determines whether the signalling observed in our systems has a direction; for example, does the choice of verb has an influence on the interpretation of its object, or similarly on the interpretation of its subject.

2.1 The causal framework

A causal scenario consists of a list of events and a set of causal relations associated to them. An event X is usually considered to be a generalised process with a set of possible inputs and outputs; these are respectively denoted as I_X and O_X . If we are considering a definite causal scenario, then the relations form a partial order where, e.g. $A \rightarrow B$ if A causally precedes B , as we do not allow causal loops (antisymmetry), and causality is clearly transitive and reflexive. If we are considering indefinite causal scenarios, then more exotic processes not compatible with the standard circuit model of quantum computation will be needed to describe the systems.

Formally, a *causal scenario* is defined to be a triple $\Sigma = (\Omega, \underline{I}, \underline{O})$ where Ω is a poset representing the causal relations between events, $\underline{I} = (I_\omega)_{\omega \in \Omega}$ includes all possible inputs for all events, and $\underline{O} = (O_\omega)_{\omega \in \Omega}$ includes all the possible outputs for all events.

Given the poset Ω , its associated set of *lower sets* is denoted by $\Lambda(\Omega)$. These are downwards closed subsets of Ω . In terms of causal events, each element of $\Lambda(\Omega)$ corresponds to a set of events that admits a description independent to other events.

The *locale of events* is the set:

$$\mathcal{L}_\Sigma = \{(\lambda, (U_\omega)_{\omega \in \lambda}) \mid \lambda \in \Lambda(\Omega), U_\omega \subseteq I_\omega, U_\omega \neq \emptyset\} \quad (1)$$

with which there is associated a partial order $U = (\lambda_U, \underline{U}) \leq V = (\lambda_V, \underline{V})$ iff $\lambda_U \subseteq \lambda_V$ and $U_\omega \subseteq V_\omega$ for all $\omega \in \lambda_U$. The meets and joins of this locale are defined as follows:

$$U \cup V = (\lambda_U \cup \lambda_V, (U_\omega \cup V_\omega)) \quad (2)$$

$$U \cap V = (\{\omega \in \lambda_U \cap \lambda_V \mid U_\omega \cap V_\omega \neq \emptyset\}, (U_\omega \cap V_\omega)) \quad (3)$$

Over a (causal) poset Ω , we introduce *causal functions*; these are the functions from inputs to outputs that respect the causal order, i.e. the inputs of succeeding events do not influence the outputs of preceding events. Formally, this means that if we order the events $\omega_1 \rightarrow \omega_2 \dots \rightarrow \omega_n$, the function $f : I_1 \times I_2 \times \dots \times I_n \rightarrow O_1 \times O_2 \times \dots \times O_n$ satisfies the following for all k 's:

$$f(a_1, \dots, a_k, a_{k+1}, \dots, a_n) \Big|_{\omega_k \downarrow} = f(a_1, \dots, a_k, a'_{k+1}, \dots, a'_n) \Big|_{\omega_k \downarrow} \quad (4)$$

We can now define the *event sheaf* as follows:

$$\begin{aligned} \mathcal{E}_\Sigma : \mathcal{L}^{op} &\rightarrow \mathbf{Set} \\ (\lambda_U, \underline{U}) &\mapsto \left\{ f : \underline{U} \rightarrow \prod_{\omega \in \lambda_U} O_\omega \mid f \text{ causal} \right\} \end{aligned} \quad (5)$$

and the restriction map is given by:

$$(U \leq V) \mapsto (f \mapsto f|_U) \quad (6)$$

The above means that the event sheaf associates to a lower set and a set of inputs on the lower set, the set of all functions which respect the causal order of the scenario. This sheaf encodes the required conditions for definite causality.

An *empirical model* is a specific distribution on causal functions and is defined over all strings of inputs. Formally, it is the following element:

$$e \in \prod_{\underline{i} \in \mathcal{I}_\Sigma} \mathcal{D}_R \mathcal{E}_\Sigma(\Omega, \underline{i}) \quad (7)$$

where \underline{i} is – by abus de langage – the string of singletons $(\{i_\omega\})_{i_\omega \in \underline{i}}$, and $\mathcal{I}_\Sigma = \prod_{I_\omega \in \Omega} I_\omega$ is the set of all strings of inputs (for all events). Also, \mathcal{D}_R is the R -distribution monad and \mathcal{E}_Σ is the event sheaf defined above. This needs to be a compatible family of distributions (by definition), i.e. the marginals on lower sets should be well-defined. For the rest of this paper, we will only make use of probabilistic distributions (i.e. $R = \mathbb{R}_+$).

2.1.1 Example

Let's consider an empirical model with only two events $\Omega = \{A, B\}$ with the single causal relation $A \rightarrow B$, such that $I_A = I_B = O_A = O_B = \{0, 1\}$, and the probability distribution on $\Omega \in \Lambda(\Omega)$ given as follows:

(A, B)		Output			
		(0,0)	(0,1)	(1,0)	(1,1)
Input	(0,0)	0	6/13	0	7/13
	(0,1)	24/65	6/65	7/13	0
	(1,0)	23/65	0	14/65	28/65
	(1,1)	23/260	69/260	42/65	0

(8)

These types of models are called (2,2,2) Bell-type scenarios (2 parties, 2 possible inputs each, 2 possible outcomes each) and are the simplest non-trivial empirical models with definite causality (up to relabelling). (2,2,2) Bell-type scenarios are the models explored further below, in the main body of this

work. Note that this is indeed a compatible family on the given causal scenario as the restriction of the lower set $\{A\} \in \Lambda(\Omega)$ is well defined, i.e. we have:

$$e_{(\Omega, I_A, \{0\})} \Big|_{(\{A\}, I_A)} = e_{(\Omega, I_A, \{1\})} \Big|_{(\{A\}, I_A)} = \begin{array}{c|cc} & \text{A} & \text{Output} \\ & & \begin{array}{cc} 0 & 1 \end{array} \\ \text{Input} & \begin{array}{c} 0 \\ 1 \end{array} & \begin{array}{cc} 6/13 & 7/13 \\ 23/65 & 42/65 \end{array} \end{array} \quad (9)$$

2.2 The causal fraction

So far, we assumed that the causal order of our system is known, but this is not generally the case. We now describe how given a known final distribution such as the one depicted in (8), one can decide what the underlying causal order is. In particular, one define the *causal fraction* of a (final) family of distributions which corresponds to the proportion of the model which is compatible with a given causal order. In the phenomena we are modelling, i.e. semantic ambiguity in natural language, the causal order of the system is in general unknown and so we have to start from the set of probability distributions over all events and only decide later which of these is the most likely causal order.

If a model e is not fully compatible with a given causal order Ω , we can calculate how much of the model can be explained by the causal order Ω . By definition, this is defined as the maximal $\gamma \in [0, 1]$ s.t.:

$$\gamma \cdot e^\Omega \preceq e \quad (10)$$

where the partial order \preceq is defined component-wise on empirical models defined on the same causal scenarios. Equivalently, this is to say that the causal fraction corresponds to the maximal proportion of the model which can be explained by an empirical model compatible with a given causal scenario.

In general, finding γ in 10 is a hard optimisation problem, as one needs to consider all empirical models e^Ω compatible with the causal order Ω . We here prove that there are simpler ways to approximate, using 11, or in some cases even calculate the causal fraction (see Proposition 2).

Proposition 1. *For a family of probability distributions where the causal order is not known, an upper bound of the causal fraction can be calculated as follows¹:*

$$\gamma \leq \min_{U, V} 1 - \left| e_i|_U|_{U \cap V}(\underline{o}) - e_i|_V|_{U \cap V}(\underline{o}) \right| \quad (11)$$

where $e_i|_U|_{U \cap V}$ corresponds to the restriction of e_i to first U and then from U to $U \cap V$ (and similarly for $e_i|_V|_{U \cap V}$).

Proof. For every causal empirical causal model e^Ω w.r.t. a causal scenario $\Sigma = (\Omega, I, \underline{O})$, if we have $\gamma \cdot e^\Omega \preceq e$, then both:

$$\gamma \cdot e_i^\Omega|_{U \cap V}(\underline{o}) \leq e_i|_U|_{U \cap V}(\underline{o}) \quad (12)$$

and

$$\gamma \cdot e_i^\Omega|_{U \cap V}(\underline{o}) \leq e_i|_V|_{U \cap V}(\underline{o}) \quad (13)$$

So:

$$\gamma \cdot e_i^\Omega|_{U \cap V}(\underline{o}) \leq \min_{X \in \{U, V\}} e_i|_X|_{U \cap V}(\underline{o}) \quad (14)$$

¹Note: the order of the restrictions is from left to right.

Now, since e_i are probability distributions:

$$1 - e_i|_X|_{U \cap V}(\underline{o}) = \sum_{\underline{o}' \neq \underline{o}} e_i|_X|_{U \cap V}(\underline{o}') \quad (15)$$

and similarly for e^Ω . Therefore, using $\gamma e^\Omega \preceq e$ once again:

$$\gamma \left(1 - e_i^\Omega|_X|_{U \cap V}(\underline{o}) \right) \leq \min_{X \in \{U, V\}} 1 - e_i|_X|_{U \cap V}(\underline{o}) = 1 - \max_{X \in \{U, V\}} e_i|_X|_{U \cap V}(\underline{o}) \quad (16)$$

Then, writing $m_- = \min_{X \in \{U, V\}} e_i|_X|_{U \cap V}(\underline{o})$ and $m_+ = \max_{X \in \{U, V\}} e_i|_X|_{U \cap V}(\underline{o})$ for simplicity, we use (14) and (16) to get:

$$\gamma \leq 1 - m_+ + m_- \quad (17)$$

Now, using binary minima and maxima this reduces to:

$$\gamma \leq 1 - \left| e_i|_U|_{U \cap V}(\underline{o}) - e_i|_V|_{U \cap V}(\underline{o}) \right| \quad (18)$$

And since this has to be the case for all $U, V \in \mathcal{L}$, the claimed inequality has to hold. \square

In certain cases, such as the models described in Section 2.1.1, the above inequality becomes an equality as the upper bound is attained. This is expressed and proven below.²

Proposition 2. *For the causal order $A \rightarrow B$ in a (2,2,2) Bell-type scenario, the causal fraction is given by:*

$$\gamma = \min_{i_A \in \{0,1\}, \underline{o} \in \{0,1\}} 1 - \left| e_{(i_A,0)}|_{A \rightarrow B}|_A(\underline{o}) - e_{(i_A,1)}|_{A \rightarrow B}|_A(\underline{o}) \right| \quad (19)$$

Proof. Let's describe a construction of a causal empirical model e^Ω which satisfies $\gamma \cdot e^\Omega \preceq e$, for any given (2,2,2) Bell-type model e , where γ is given as in (19).

We start by constructing a probability distribution for the event A as follows. For any $i_A \in I_A$, we select $o_A^* \in O_A$ s.t.:

$$\min_{i_B \in I_B} e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A^*) = \min_{o_A \in O_A} \min_{i_B \in I_B} e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A) \quad (20)$$

and set:

$$e_{(i_A, i_B)}^\Omega|_A(o_A^*) = \frac{\min_{i_B \in I_B} e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A^*)}{\gamma} \quad (21)$$

and $e_{(i_A, i_B)}^\Omega|_A(\neg o_A^*) = 1 - e_{(i_A, i_B)}^\Omega|_A(o_A^*)$. Then we have:

$$\gamma \cdot e_{(i_A, i_B)}^\Omega|_{A \rightarrow B}|_A(o_A) \leq e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A) \quad (22)$$

for all $(i_A, i_B) \in I_A \times I_B$, and for all possible outcome $o_A \in O_A$.

One can then extend this distribution to the lowerset $A \rightarrow B = \Omega$ by setting, for example:

$$e_{(i_A, i_B)}^\Omega(o_A, o_B) = \frac{e_{(i_A, i_B)}^\Omega|_{A \rightarrow B}(o_A, o_B)}{e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A)} e_{(i_A, i_B)}|_{A \rightarrow B}|_A(o_A) \quad (23)$$

It is routine to check that this construction leads to a valid empirical model e^Ω , which does indeed satisfy $\gamma \cdot e^\Omega \preceq e$. \square

²We believe that this equality in fact holds for a larger range of systems. Proving a more general version of the proposition is left to future work.

2.2.1 Example

Let's consider another (2,2,2) Bell-type example, where this time the final family of distributions are:

		(A,B)	Output			
			(0,0)	(0,1)	(1,0)	(1,1)
e =	Input	(0,0)	0	1/7	0	6/7
		(0,1)	2/3	1/6	1/6	0
		(1,0)	1/4	0	1/4	1/2
		(1,1)	1/5	3/5	1/5	0

(24)

The marginal distributions for the two choices of inputs for B are given by:

		A	Output	
			0	1
Input	0	1/7	6/7	
	1	1/4	3/4	

(25)

if $i_B = 0$ and:

		A	Output	
			0	1
Input	0	5/6	1/6	
	1	4/5	1/5	

(26)

if $i_B = 1$. Therefore, the model is not compatible with the causal scenario $A \rightarrow B$ and causal fraction can be calculated as $\gamma = 13/42$ using (19). In fact, the model in Section 2.1.1, which we will denote as $e^{A \rightarrow B}$ does satisfy:

$$\frac{13}{42} e^{A \rightarrow B} \preceq e \quad (27)$$

This means that less than 31% of this scenario can be explained as process where the choice of input of A can influence the output of B .

3 The experiment

We started from a list of ambiguous nouns (homonymous and polysemous) and list of ambiguous verbs (homonymous and polysemous) and manually selected the verb-noun pairs for which several possible interpretations of both the verb-object and subject-verb phrases were possible. From these, we randomly selected 50 phrases that had a homonymous verb and a homonymous noun, 50 phrases with a homonymous verb and a polysemous noun, 50 phrases with a polysemous verb and a homonymous noun, and finally 50 phrases with a polysemous verb and a polysemous noun. This resulted in a dataset of 200 ambiguous phrases with an equal number of different types of ambiguous (polysemous or homonymous) verbs and nouns.

We launched this dataset on the Amazon Mechanical Turk (AMT) engine to collect human judgments. AMT workers were tasked to rate the plausibility of the different interpretations of the ambiguous phrases of the dataset. Each worker was provided with all interpretations of each of the words of each phrase and only saw a subset of the dataset with 8 phrases in it. These sub-datasets are referred to as *HIT* by AMT. The phrases of each HIT only contained either subject-verb or verb-object combinations. We ranged the plausibility scores over the discrete 0 to 7 interval and had 8 degrees of plausibility. In Psycholinguistics, a 7 grade scale has been deemed as most effective for human subjects, however, in order to avoid randomly chosen and accidental answers due to indecision, we also allowed for an 8th

neutral grade. We positioned this option in the middle of the scale and designed an 8 grade scale. Each worker was thus tasked to choose one of the 8 provided “scale-description” degrees of plausibility for each phrase. These were as follows:

0: impossible 1: extremely unlikely 2: very unlikely 3: somewhat unlikely
4: neutral 5: somewhat likely 6: very likely 7: extremely likely

The annotations were used to compute a probability distribution for all the possible interpretations of a phrase. This was done by averaging the scores of all the workers for a particular phrase, and then normalising the obtained average score. After this step, we combined the probability distributions to form “Bell-type scenarios”, in which we then studied the causality and contextuality of the empirical models.

Each phrase was annotated by 25 workers and in total we had 1250 annotators. An annotator could choose to annotate multiple HIT’s, and spent on average 10 minutes per HIT. We paid the workers based on the minimum wage in the UK. The probability distributions obtained from the workers’ plausibility scales were used to form 322 (2,2,2) Bell-type scenarios corresponding to subject-verb phrases, and the same number of verb-object phrases.

An examples of a subject-verb phrase was ‘pitcher threw’. The annotators were provided with all possible meaning combinations of the phrase, which were as follows:

- combination 1: ‘pitcher’ is a type of jug and ‘throw’ is the literal action of sending something through the air, e.g. a ball.
- combination 2: ‘pitcher’ is a type of jug and ‘throw’ is the figurative action of sending something into a different state, e.g. a shadow.
- combination 3: ‘pitcher’ is a baseball player and ‘throw’ is the literal action of sending something through the air, e.g. a ball.
- combination 4: ‘pitcher’ is a baseball player and ‘throw’ is the figurative action of sending something into a different state, e.g. a shadow.

A typical annotation for ‘pitcher threw’ was as follows:

combination 1: 0, combination 2: 5, combination 3: 6, combination 4: 5

4 Causal relations in natural language models

4.1 Causality of SVO phrases

We analyse the causality of subject-verb and verb-object phrases separately. Phrases were then combined in (2,2,2) Bell-type scenarios as described in the previous section. In these scenarios, an event corresponds to choosing a grammatical type for a word, i.e. subject of a verb or object of a verb or a verb, as input and then select an interpretation of that word as output. An example of such an event is choosing the word *plant* as the subject of a verb (i.e. *plant* will be the input) and then picking the “factory” interpretation of it as the output. An example of a subject-verb and object verb empirical model is shown in Fig. 1; in turns, the full dataset can be found in [15].

What we have access to is the final distribution of possible interpretations of a phrase. What we are interested in is whether our empirical models are compatible with a definite causal order and indeed if so, which one. For each subject-verb model (resp. verb-object model) we have two events: *S* and *V* (resp. *V* and *O*); these corresponds to choosing the subject and the verb (resp. verb and object) in a

	(0,0)	(0,1)	(1,0)	(1,1)
<i>the paper bored</i>	0.21	0.13	0.51	0.15
<i>the paper launched</i>	0.18	0.23	0.16	0.43
<i>the plant bored</i>	0.17	0.30	0.16	0.37
<i>the plant launched</i>	0.19	0.20	0.28	0.33

(a) A subject-verb model.

	(0,0)	(0,1)	(1,0)	(1,1)
<i>bored the paper</i>	0.19	0.23	0.29	0.29
<i>bored the plant</i>	0.18	0.21	0.32	0.29
<i>launched the paper</i>	0.26	0.23	0.21	0.30
<i>launched the plant</i>	0.29	0.18	0.23	0.30

(b) A verb-object model.

Figure 1: Examples of empirical models. Note: *plant* and *bore* have multiple meanings (respectively the living organism or the factory and making people loose interest or making holes) and *paper* and *launch* have multiple senses (respectively the material or content of an article and literally setting something in motion or starting an activity.)

given phrase. There are 3 possible definite (acyclic) causal orders associated to each subject-verb phrase, namely $S \rightarrow V$, $V \rightarrow S$ and S and V being no-signalling (resp. $V \rightarrow O$, $O \rightarrow V$ and once again having V and O no-signalling in verb-object phrases). The obtained causal fractions for all subject-verb and verb-object models are shown in Figs. 2a and 2b respectively.

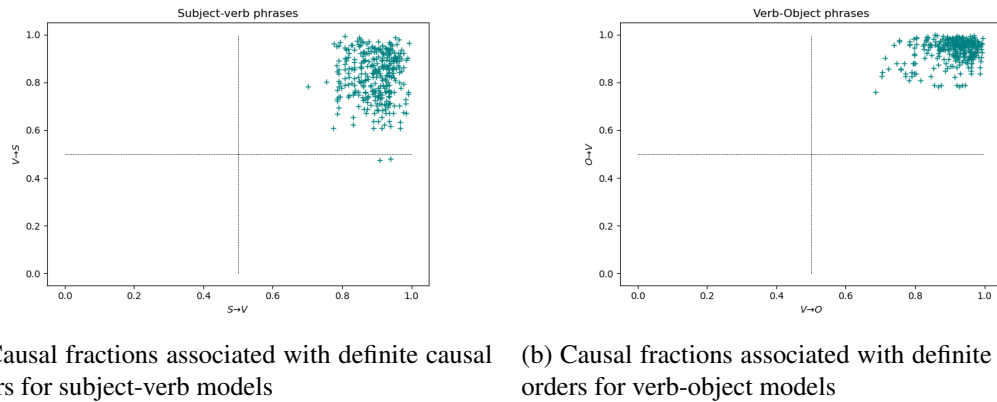


Figure 2

What emerged from the data is that subject-verb phrases are predominantly compatible with the $S \rightarrow V$ causal order. All of the models had a causal fraction > 0.7 . Both the $V \rightarrow S$ and the no-signalling fractions achieved lower causal fraction values, see Fig.3a. A similar result held for all of the verb-object models: these also achieved a causal fraction with the $O \rightarrow V$ order higher than 70%, and their other causal fractions reached significantly lower scores, see Fig.3b. What these results suggest is that the choice of nouns (subject or object) have more influence on which interpretation of the verb is selected, rather than the other way around, i.e. that the choice of verb has more influence on the interpretation of its subject or object.

Finally the causal fractions (in either direction) were also found to be higher for verb-object phrases compared to subject-verb phrases. This would suggests that verb-object phrases are in general easier to disambiguate than subject-verb phrases (see Fig.4).

4.2 Causality and levels of ambiguity

After determining the causal order of the phrases of our dataset, we would like to establish a relationship between the type of ambiguity of each word within a phrase, i.e. whether they are homonymous or

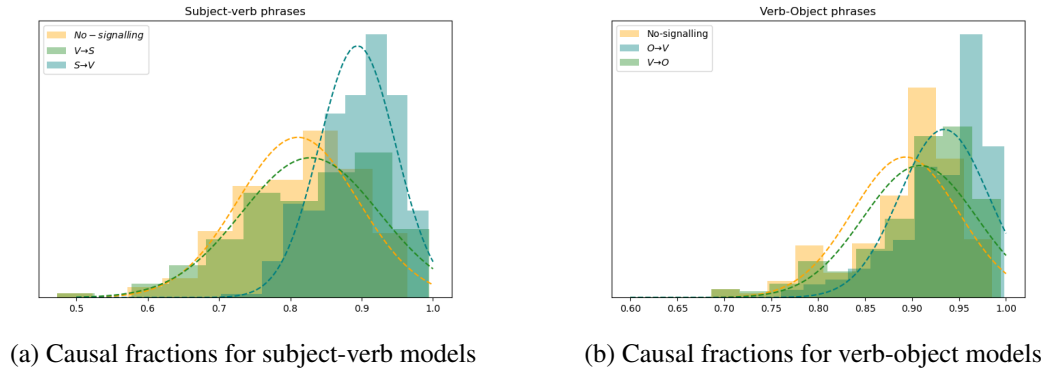


Figure 3

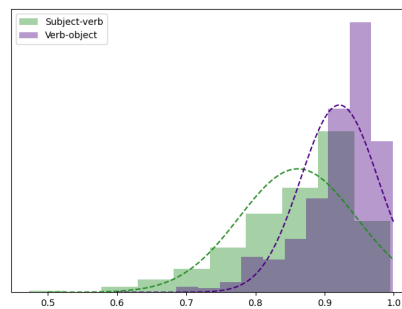


Figure 4: Causal fractions in subject-verb models and in verb-object models.

polysemous, and the causal fraction of the phrase. In other words, we would like to know whether the type of ambiguity has an effect on the causal fraction.

After investigating, we found out that the types of ambiguities of the words in the model do not play a major role in the value of the different causal fractions. Indeed, no apparent correlation was observed in the verb-object models, where we computed a Spearman R-coefficient $\rho = 0.009$, with a p -value $> 87\%$. There was only a mild effect for the subject-verb models, here the more polysemous the words in a model, the higher the $S \rightarrow V$ causal fraction (Spearman R-coefficient $\rho = 0.15$, p -value $< 0.7\%$). These results are depicted in Fig. 5a.

The more significant difference was related to the type of ambiguities of the *noun* and the *verb* of the phrase. In both subject-verb and verb-object phrases, the $S \rightarrow V$ and $O \rightarrow V$ causal fractions were higher when the verb was polysemous (Spearman R-coefficient $\rho = 0.17$, p -value $< 0.2\%$ for subject-verb phrases, R-coefficient $\rho = 0.16$, p -value $< 0.4\%$ for verb-object phrases), as depicted in see Fig.6c and Fig.6b respectively. In addition, the causal fraction associated with $O \rightarrow V$ was higher whenever objects were homonymous (Spearman R-coefficient $\rho = 0.14$, p -value $< 2\%$), as depicted in Fig.6d, no such effect was observed for the subject-verb phrases (Spearman R-coefficient $\rho = 0.04$, p -value $> 50\%$).

One may not that the Spearman coefficients found above are fairly low ($\rho < 0.2$), which does suggest that the correlations observed are quite mild. However, the p -values showed that the correlations claimed in the above paragraph are statistically significant, i.e. it is highly unlikely that no correlation exist between the causal fraction and levels of ambiguity.

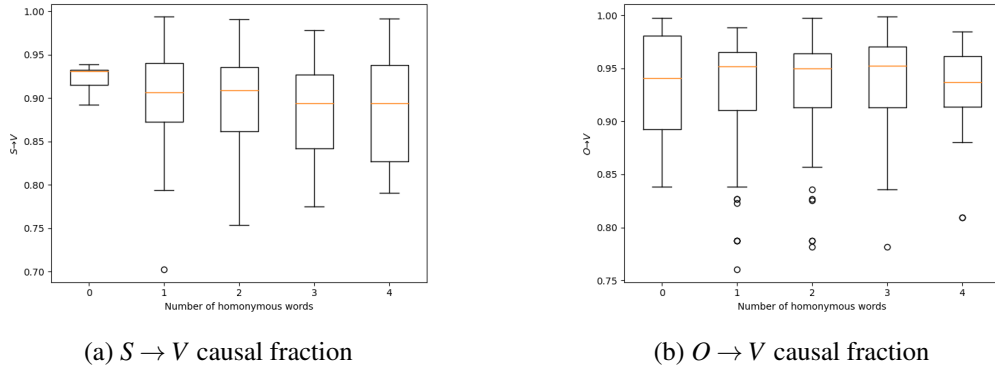


Figure 5: Causal fraction with respect to number of homonymous words in the model

5 Conclusion

We believe that our data reflects the findings of the Psycholinguistics and eye-tracking experiments. Indeed, the study of Pickering and Frisson in [10] observed a delay in disambiguation of ambiguous (transitive) verbs in comparison to the processing of ambiguous nouns. Hence, it would make sense that in a subject-verb or verb-object phrase, where each word is ambiguous, the verb would be disambiguated last, thus explaining the dominance of the $S \rightarrow V$ and $O \rightarrow V$ causal fractions.

It has also been shown in [10] that polysemous verbs are disambiguated even later than homonymous verbs. This is consistent with our finding in the subject-verb phrases, where models with several polysemous verbs tend to have a higher causal fraction, as the verb would be disambiguated even later than homonymous verbs would. Similarly, several similar studies have shown that homonymous nouns are (partially) disambiguated instantly [6], whilst polysemous nouns generally require a larger context in order to be (even partially) disambiguated [6, 9, 7]. This fits with our data for the verb-object models, where it was observed that $O \rightarrow V$ causal fractions are higher whenever the object had multiple meanings (which is then disambiguated even faster than when the object has multiple senses).

What remains to find out is why the effect of the ambiguity of nouns was different in subject-verb and verb-object models. One way to interpret this would be by taking into account the difference between pre and post contexts of the phrases. It was shown in [6] that homonymous nouns were disambiguated a lot faster than polysemous nouns, when the disambiguation context occurs before the target words. This would nicely explain the difference between verb-object and subject-verb phrases. The only possible disambiguation context for nouns in verb-object phrases is the verb, and therefore in such cases, it should be even clearer that the verb would be disambiguated after its object when the latter has multiple meanings. In the case of disambiguation context for nouns, it was shown that reading times were longer for both homonymous and polysemous nouns. This explains why we did not detect any particular effect in the subject-verb phrases.

5.1 Future Work

The framework developed in [8] allows us to distinguish quantum-like (i.e. contextual) and classical processes assuming that a given model is compatible with a definite causal structure. Due to the approximate nature of the probability distribution obtained, i.e. that all the probabilities calculated are not exact, the causal fractions are not exactly 1, and there are infinitely many sub-distributions that would

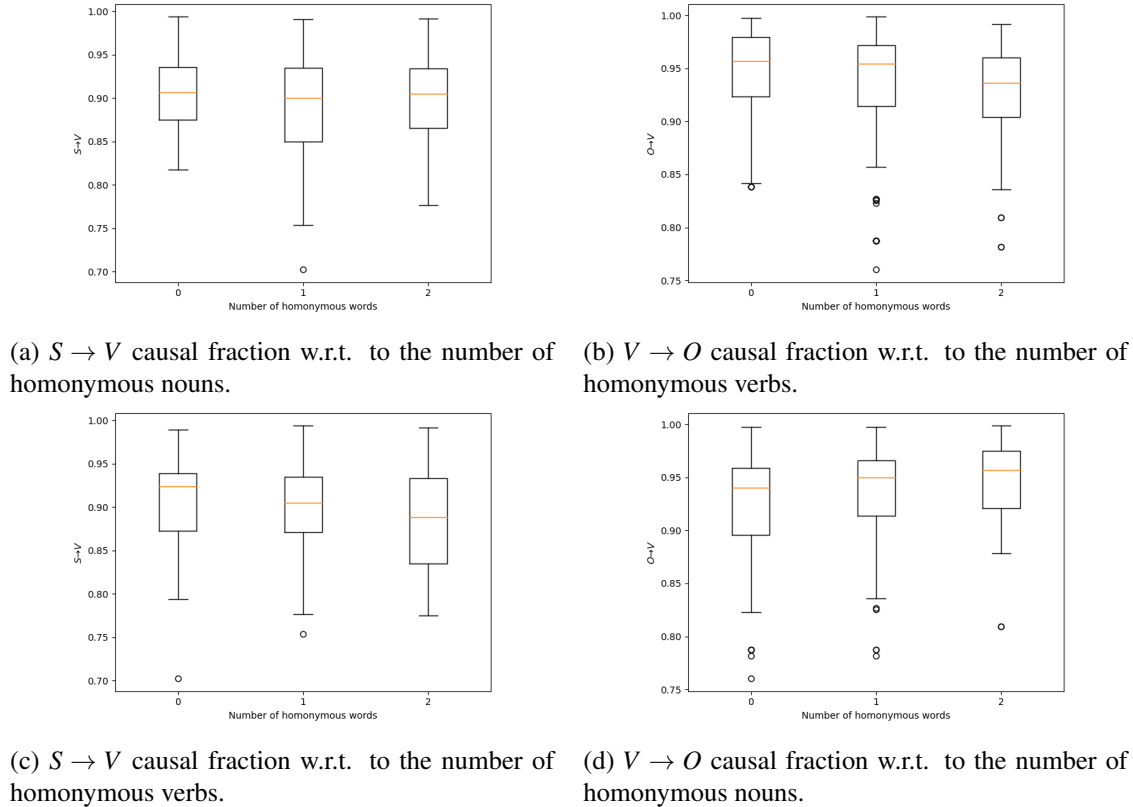


Figure 6

be compatible with a given causal order. In such cases, it is not easy to determine whether our empirical model could be contextual or not. Investigating how to extend the model to include these is an interesting future direction. Also, we would like to collect probabilities using state of the art neural embeddings such as BERT and extending the current study to ambiguities arising from syntax and discourse. Finally, computing casual fragments is much cheaper and quicker than collecting data in labs with eye-tracking equipment and exploring this impact case is our future ambition. We would also like to relate our work with related work such as the density matrices model of lexical ambiguity[11]. Similarly, we could also study other datasets from psychology and behavioural science which showed signalling but no “true contextuality”[4], such as the concept combination dataset of Bruza et. al.[3].

References

- [1] Samson Abramsky & Adam Brandenburger (2011): *The sheaf-theoretic structure of non-locality and contextuality*. *New J. Phys.* 13, p. 113036, doi:10.1088/1367-2630/13/11/113036. arXiv:<https://arxiv.org/abs/1102.0264v7>.
- [2] Peter Bruza, Kirsty Kitto, Douglas Nelson & Cathy McEvoy (2009): *Is there something quantum-like about the human mental lexicon?* *Journal of mathematical psychology* 53(5), pp. 363–377, doi:10.1016/j.jmp.2009.04.004. Available at <https://pubmed.ncbi.nlm.nih.gov/20224806>. 20224806[pmid].

- [3] Peter D. Bruza, Kirsty Kitto, Brentyn J. Ramm & Laurianne Sitbon (2015): *A probabilistic framework for analysing the compositionality of conceptual combinations*. *Journal of Mathematical Psychology* 67, pp. 26–38, doi:10.1016/j.jmp.2015.06.002. Available at <https://www.sciencedirect.com/science/article/pii/S002224961500036X>.
- [4] E. N. Dzhafarov, Ru Zhang & Janne Kujala (2016): *Is there contextuality in behavioural and social systems?* *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374(2058), p. 20150099, doi:10.1098/rsta.2015.0099. arXiv:<https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2015.0099>.
- [5] Ehtibar N. Dzhafarov & Janne V. Kujala (2016): *Context–content systems of random variables: The Contextuality-by-Default theory*. *Journal of Mathematical Psychology* 74, pp. 11 – 33, doi:10.1016/j.jmp.2016.04.010. arXiv:<https://arxiv.org/abs/1511.03516v6>. Foundations of Probability Theory in Psychology and Beyond.
- [6] Lyn Frazier & Keith Rayner (1990): *Taking on semantic commitments: Processing multiple meanings vs. multiple senses*. *Journal of Memory and Language* 29(2), pp. 181–200, doi:10.1016/0749-596X(90)90071-72. Available at <https://www.sciencedirect.com/science/article/pii/0749596X90900717>.
- [7] Steven Frisson & Martin Pickering (2009): *Semantic Underspecification in Language Processing*. *Language and Linguistics Compass* 3, pp. 111–127, doi:10.1111/j.1749-818X.2008.00104.x.
- [8] Stefano Gogioso & Nicola Pinzani (2021): *The Sheaf-Theoretic Structure of Definite Causality*. *Electronic Proceedings in Theoretical Computer Science* 343, p. 301–324, doi:10.4204/eptcs.343.13. arXiv:<http://dx.doi.org/10.4204/EPTCS.343.13>.
- [9] Martin Pickering & Steven Frisson (2001): *Obtaining a figurative interpretation of a word: support for underspecification*. *Metaphor and Symbol* 16, pp. 149–171, doi:10.1080/10926488.2001.9678893
- [10] Martin Pickering & Steven Frisson (2001): *Processing Ambiguous Verbs: Evidence from Eye Movements*. *Journal of experimental psychology. Learning, memory, and cognition* 27, pp. 556–73, doi:10.1037/0278-7393.27.2.556.
- [11] Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke & Mehrnoosh Sadrzadeh (2015): *Open System Categorical Quantum Semantics in Natural Language Processing*, doi:10.48550/ARXIV.1502.00831. Available at <https://arxiv.org/abs/1502.00831>.
- [12] D. Wang, M. Sadrzadeh, S. Abramsky & V. Cervantes (2021): *Analysing Ambiguous Nouns and Verbs with Quantum Contextuality Tools*. *Journal of Cognitive Science* 22(3), pp. 391–420, doi:10.17791/jcs.2021.22.3.391.
- [13] D. Wang, M. Sadrzadeh, S. Abramsky & V. Cervantes (2021): *In Search of True Contextuality in Natural Language*. Fourth Workshop on Quantum Contextuality and Quantum Mechanics and Beyond (QCQMB). Outstanding Paper Award.
- [14] D. Wang, M. Sadrzadeh, S. Abramsky & V. Cervantes (2021): *On the Quantum-like Contextuality of Ambiguous Phrases*. In: *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science*, Association for Computational Linguistics, p. 42–52, doi:10.48550/arXiv.2107.14589. arXiv:<https://arxiv.org/abs/2107.14589>.
- [15] Daphne Wang (2022): *Causality of lexically ambiguous phrases*. Available at <https://github.com/wangdaphne/Causality-of-lexically-ambiguous-phrases>.

Tunable Quantum Neural Networks in the QPAC-Learning Framework

Viet Pham Ngoc

Imperial College London
London, United Kingdom

viet.pham-ngoc17@imperial.ac.uk

David Tuckey

Imperial College London
London, United Kingdom

david.tuckey17@imperial.ac.uk

Herbert Wiklicky

Imperial College London
London, United Kingdom

h.wiklicky@imperial.ac.uk

In this paper, we investigate the performances of tunable quantum neural networks in the Quantum Probably Approximately Correct (QPAC) learning framework. Tunable neural networks are quantum circuits made of multi-controlled X gates. By tuning the set of controls these circuits are able to approximate any Boolean functions. This architecture is particularly suited to be used in the QPAC-learning framework as it can handle the superposition produced by the oracle. In order to tune the network so that it can approximate a target concept, we have devised and implemented an algorithm based on amplitude amplification. The numerical results show that this approach can efficiently learn concepts from a simple class.

1 Introduction

Machine learning is believed to be an application of quantum computing that will yield promising results. It is the reason why, in the past years, significant efforts have been put into developing machine learning techniques suited to quantum computers. One interesting approach to this task consists in adapting existing classical techniques [16, 20, 18, 13] to take advantage of quantum properties and gain some speed-up.

Introduced by [21], probably approximately correct (PAC) learning provides a mathematical framework to analyse classical machine learning techniques. This framework revolves around the existence of an oracle that provides samples drawn from the instance space. These examples are then used as examples for an algorithm to learn a target concept. The efficiency of this learning algorithm can then be characterised thanks to its sampling complexity. Given the momentum gained by quantum techniques for machine learning, it seems natural for a quantum equivalence of PAC-learning to be introduced. This is exactly what was done in [6] with QPAC-learning. In this framework, instead of providing samples from the instance space, the oracle generates a superposition of all the examples. Similarly to PAC-learning, QPAC-learning can be used to evaluate the learning efficiency of quantum algorithm.

In this paper we are using this framework to study a learning algorithm based on quantum amplitude amplification [5]. This fundamental quantum procedure allows us both to compare the error rate to a threshold as well as increase the probability of measuring the errors produced by the learner. These measured errors are then used to tune the learner in order to decrease the error rate. In our case, this learner is a tunable quantum neural network [15] which is essentially a quantum circuit made of multi-controlled X gates. We prove that this setup is able to learn efficiently by implementing it and testing it against a simple class of concepts.

2 Related Works

Studied in [15], the tunable quantum neural networks (TNN) are an interesting kind of quantum circuits in the sense that they naturally express Boolean functions hence are able to approximate any target function. At the core of this concept is the fact that a Boolean function can be expanded into a polynomial form called the algebraic normal form (ANF). This polynomial form can then easily be transposed into a quantum circuit.

Let $u = u_0 \dots u_{n-1} \in \mathbb{B}^n$, we denote $\mathbf{1}_u = \{i \in [0, n-1] \mid u_i = 1\}$ and for $x \in \mathbb{B}^n$, $x^u = \prod_{i \in \mathbf{1}_u} x_i$. Let $f \in \mathbb{B}^{\mathbb{B}^n}$ then its algebraic normal form is:

$$f(x) = \bigoplus_{u \in \mathbb{B}^n} \alpha_u x^u \quad (1)$$

Where $\alpha_u \in \{0, 1\}$ denotes the absence or the presence of the monomial x^u in the expansion. As an example, let us consider the function $g \in \mathbb{B}^{\mathbb{B}^2}$ defined by $g(x_0, x_1) = 1 \oplus x_1 \oplus x_0.x_1$. With the previously introduced notations, its ANF can be written: $g(x) = x^{00} \oplus x^{01} \oplus x^{11}$ with $\alpha_{00} = \alpha_{01} = \alpha_{11} = 1$ and $\alpha_{10} = 0$. Using this form, any Boolean function can be expressed with a quantum circuit made of multi-controlled **X** gates.

Let $f \in \mathbb{B}^{\mathbb{B}^n}$ with ANF as in (1) and consider a quantum circuit of $n+1$ qubits with the first n qubits being denoted q_0, \dots, q_{n-1} and the last one being the ancillary qubit, denoted a . Now for $u \in \mathbb{B}^n$ such that $\alpha_u = 1$ in the ANF of f , place an **X** gate controlled by the qubits $\{q_i \mid i \in \mathbf{1}_u\}$ on the ancillary qubit. Let **QC** be the resulting circuit, then it is such that for $x \in \mathbb{B}^n$ and $b \in \mathbb{B}$:

$$\mathbf{QC} |x\rangle |b\rangle = |x\rangle |b \oplus f(x)\rangle$$

Let us consider the function $g \in \mathbb{B}^{\mathbb{B}^2}$ introduced previously, then the circuit pictured in Figure 1 is expressing g .

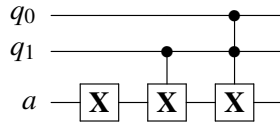


Figure 1: Circuit expressing $g(x) = 1 \oplus x_1 \oplus x_0.x_1$.

A tunable neural network is then a quantum circuit of this type for which the set of controlled **X** gates can be tuned so that the expressed function is an approximation of a target function. Now let $f \in \mathbb{B}^{\mathbb{B}^n}$ and consider the network $\mathbf{TNN}(f)$ expressing f , then for a superposition of the form $|\phi\rangle = \sum_{x \in \mathbb{B}^n} d_x |x\rangle |0\rangle$ we have:

$$\mathbf{TNN}(f) |\phi\rangle = \sum_{x \in \mathbb{B}^n} d_x |x\rangle |f(x)\rangle$$

Which is reminiscent of the output from the oracle encountered in the QPAC-learning framework.

QPAC-Learning has been introduced in [6] and is a quantum version of the work presented in [21]. In this framework, we call $\mathcal{C} \subseteq \mathbb{B}^{\mathbb{B}^n}$ a class of concept and the aim is to learn $c \in \mathcal{C}$. A probability distribution D is placed over \mathbb{B}^n and for a hypothesis $h \in \mathbb{B}^{\mathbb{B}^n}$ the error is defined by

$$err_D(h, c) = P_{x \sim D}(h(x) \neq c(x)) \quad (2)$$

For $0 < \varepsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$, the goal is then to produce $h \in \mathcal{C}$ such that:

$$P(\text{err}_D(h, c) < \varepsilon) > 1 - \delta \tag{3}$$

A class of concept \mathcal{C} is said to be PAC-learnable with an algorithm A if for $c \in \mathcal{C}$, for all distribution D and for $0 < \varepsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, A will output a hypothesis $h \in \mathbb{B}^{\mathbb{B}^n}$ verifying (3).

During the learning process, we are given access to an oracle $\mathbf{EX}(c, D)$ such that each call to this oracle will produce the superposition $|\Psi(c, D)\rangle$:

$$|\Psi(c, D)\rangle = \sum_{x \in \mathbb{B}^n} \sqrt{D(x)} |x\rangle |c(x)\rangle$$

For the rest of the paper we will assume that $\mathbf{EX}(c, D)$ is a quantum gate such that:

$$\mathbf{EX}(c, D) |0\rangle = |\Psi(c, D)\rangle$$

With these notations, for a concept c and a hypothesis h we have:

$$\text{err}_D(h, c) = \sum_{h(x) \neq c(x)} D(x)$$

A learning algorithm is then said to be an efficient PAC-learner when the number of calls to \mathbf{EX} that are necessary to attain (3) is polynomial in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$.

The learning algorithm presented in this paper is based on quantum amplitude amplification. Introduced by [5] the quantum amplitude amplification algorithm is a generalisation of Grover's algorithm [9]. Let G be the set of states we want to measure and \mathcal{A} be a quantum circuit such that $\mathcal{A} |0\rangle = |\Phi\rangle$ with:

$$|\Phi\rangle = \sum_{x \notin G} d_x |x\rangle + \sum_{x \in G} d_x |x\rangle$$

This state can be rewritten as

$$|\Phi\rangle = \cos(\theta) |\Phi_B\rangle + \sin(\theta) |\Phi_G\rangle \tag{4}$$

With $\theta \in [0, \frac{\pi}{2}]$, $\cos(\theta) = \sqrt{\sum_{x \notin G} |d_x|^2}$, $|\Phi_B\rangle = \frac{1}{\cos(\theta)} \sum_{x \notin G} d_x |x\rangle$, $\sin(\theta) = \sqrt{\sum_{x \in G} |d_x|^2}$ and $|\Phi_G\rangle = \frac{1}{\sin(\theta)} \sum_{x \in G} d_x |x\rangle$. We also have $\langle \Phi_B | \Phi_G \rangle = 0$

Now let \mathcal{X}_G be such that:

$$\mathcal{X}_G |x\rangle = \begin{cases} -|x\rangle & \text{if } x \in G \\ |x\rangle & \text{otherwise} \end{cases}$$

Then we have $\mathcal{X}_G |\Phi_G\rangle = -|\Phi_G\rangle$ and $\mathcal{X}_G |\Phi_B\rangle = |\Phi_B\rangle$. We also define $\mathcal{X}_0 = \mathbf{I} - 2|0\rangle\langle 0|$ then by denoting $\mathbf{Q} = -\mathcal{A} \mathcal{X}_0 \mathcal{A}^{-1} \mathcal{X}_G$, \mathbf{Q} is the diffusion operator and acts on $|\Phi\rangle$ in the following way:

$$\mathbf{Q}^m |\Phi\rangle = \cos((2m+1)\theta) |\Phi_B\rangle + \sin((2m+1)\theta) |\Phi_G\rangle \text{ for } m \in \mathbb{N}$$

Suppose we know θ , then by choosing m such that $(2m+1)\theta \approx \frac{\pi}{2}$, for example $m = \lfloor \frac{1}{2} (\frac{\pi}{2\theta} - 1) \rfloor$, we are ensured that when measuring, a state in G will almost certainly be measured.

This amplitude amplification procedure is at the heart of numerous quantum amplitude estimation algorithms [5, 14, 1]. In this setup, a state of a form similar to that of (4) is given but $\theta \in [0, \frac{\pi}{2}]$ is unknown and these algorithms seek to approximate $\sin(\theta)$. Where [5] makes use of quantum Fourier transforms an integrant part of the algorithm, [14, 1] do without while still maintaining quantum speed-up. If $a = \sin(\theta) \in [0, 1]$ is the amplitude to be evaluated and $\varepsilon, \delta > 0$, then these algorithms output an estimation \tilde{a} such that $P(|\tilde{a} - a| < a\varepsilon) > 1 - \delta$.

3 TNN in the QPAC-Learning Framework

As said previously, tunable networks are particularly well suited to be employed in the QPAC-learning framework. Let $\mathcal{C} \subseteq \mathbb{B}^n$ be a class of concept and $c \in \mathcal{C}$. Let D be a probability distribution over \mathbb{B}^n and $\mathbf{EX}(c, D)$ the oracle such that:

$$\mathbf{EX}(c, D) |0\rangle = |\Psi(c, D)\rangle = \sum_{x \in \mathbb{B}^n} \sqrt{D(x)} |x\rangle |c(x)\rangle$$

Now let $\mathbf{TNN}(h)$ be a tunable neural network expressing $h \in \mathbb{B}^n$ in its current state. Then we have:

$$\begin{aligned} |\Phi\rangle &= \mathbf{TNN}(h) |\Psi(c, D)\rangle \\ &= \sum_{x \in \mathbb{B}^n} \sqrt{D(x)} |x\rangle |c(x) \oplus h(x)\rangle \\ &= \sum_{h(x)=c(x)} \sqrt{D(x)} |x\rangle |0\rangle + \sum_{h(x) \neq c(x)} \sqrt{D(x)} |x\rangle |1\rangle \end{aligned}$$

The corresponding circuit is given in Figure 2.

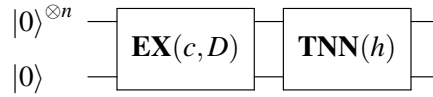


Figure 2: Circuit resulting in the state $|\Phi\rangle$.

The state $|\Phi\rangle$ can then be rewritten as:

$$|\Phi\rangle = \cos(\theta_e) |\phi_g\rangle |0\rangle + \sin(\theta_e) |\phi_e\rangle |1\rangle$$

With $\langle \phi_g, 0 | \phi_e, 1 \rangle = 0$. This state is similar to the one encountered in the amplitude amplification procedure. Because $\sin(\theta_e) = \sqrt{\sum_{h(x) \neq c(x)} D(x)}$, we also have

$$\sin^2(\theta_e) = \sum_{h(x) \neq c(x)} D(x) = \text{err}_D(h, c)$$

In order to use tunable neural networks in the QPAC-learning framework we thus need a mean to estimate $\sin^2(\theta_e)$ and compare it to $0 < \varepsilon < 1/2$. If the error is smaller than ε , stop. Otherwise, tune the network so that it expresses another hypothesis h' and repeat. The algorithm proposed in this paper does just that by using amplitude amplification.

4 Tuning Algorithm

At its core, the algorithm is similar to quantum amplitude estimation, the difference being that it compares the unknown amplitude $\text{err}_D(h, c)$ to ε instead of estimating the amplitude to a relative error of ε . Let θ_{err} and θ_ε such that $\sin^2(\theta_{err}) = \text{err}_D(h, c)$ and $\sin^2(\theta_\varepsilon) = \varepsilon$. Because D is a distribution we can take $\theta_{err} \in [0, \frac{\pi}{2}]$ and because $0 < \varepsilon < \frac{1}{2}$ we can assume $\theta_\varepsilon \in]0, \frac{\pi}{4}[$. There are then two possible cases: either $\text{err}_D(h, c) \geq \varepsilon$ or $\text{err}_D(h, c) < \varepsilon$. Because \sin is increasing on $[0, \frac{\pi}{2}]$, these translate into $\theta_{err} \geq \theta_\varepsilon$ or $\theta_{err} < \theta_\varepsilon$ respectively. Now let m_{\max} be the smallest integer such that $(2m_{\max} + 1)\theta_\varepsilon \geq \frac{\pi}{4}$, we introduce this following simple lemma:

Lemma 4.1. Let $\theta \in [0, \frac{\pi}{2}]$. If for all $m \leq m_{\max}$, we have $\sin^2((2m+1)\theta) < \frac{1}{2}$ then $\theta < \theta_\epsilon$

Proof. We show this by contraposition.

Let $\theta \geq \theta_\epsilon$.

Suppose $\theta \in [\frac{\pi}{4}, \frac{\pi}{2}]$, then taking $m = 0 \leq m_{\max}$, we have $\sin^2(\theta) \geq \frac{1}{2}$.

Now suppose that $\theta \in [0, \frac{\pi}{4}]$. By definition we have $m_{\max} = \lceil \frac{1}{2} (\frac{\pi}{4\theta_\epsilon} - 1) \rceil$ and we denote $m = \lceil \frac{1}{2} (\frac{\pi}{4\theta} - 1) \rceil$, then $m \leq m_{\max}$. Let $0 \leq \alpha < 1$ such that $m = \frac{1}{2} (\frac{\pi}{4\theta} - 1) + \alpha$. Then we have $(2m+1)\theta = \frac{\pi}{4} + 2\alpha\theta$ and $\frac{\pi}{4} \leq (2m+1)\theta \leq \frac{3\pi}{4}$. This leads to $\sin^2((2m+1)\theta) \geq \frac{1}{2}$. Hence Lemma 4.1. □

This result is illustrated in Figures 3 and 4.

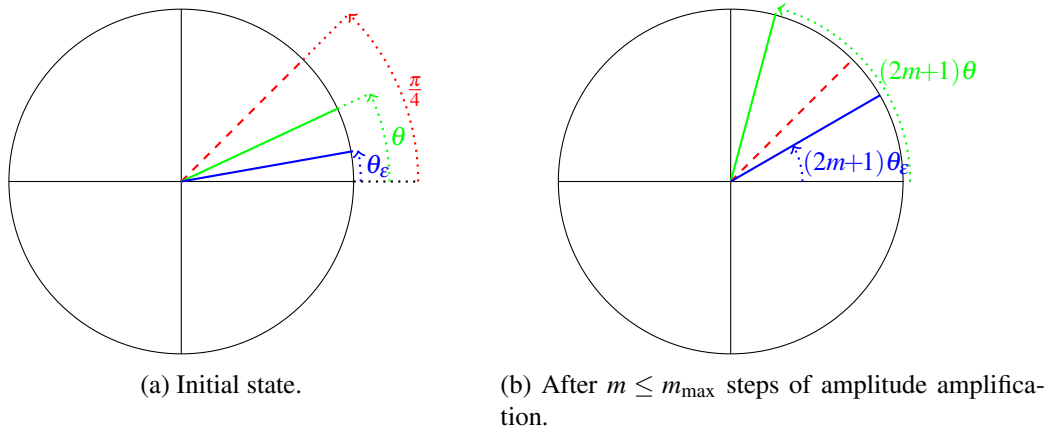


Figure 3: Case where $\theta \geq \theta_\epsilon$.

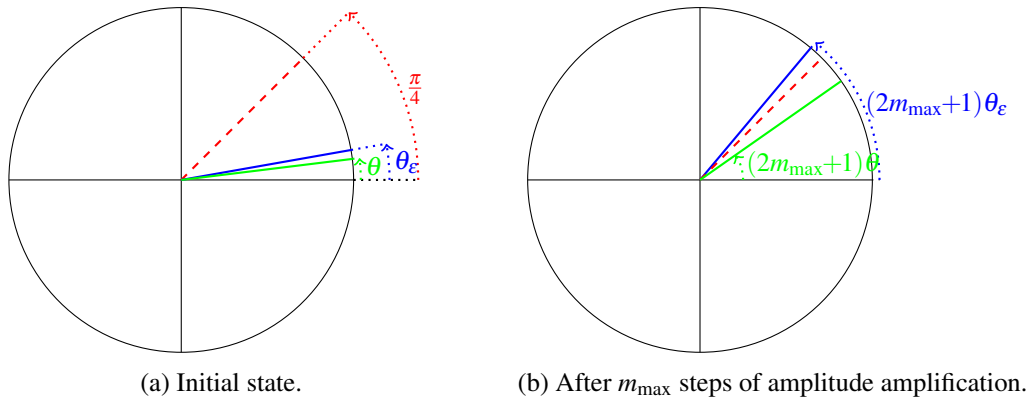


Figure 4: Case where $\theta < \theta_\epsilon$.

The tuning algorithm works as follow. After each update of the tunable network, the error is compared to ϵ thanks to Lemma 4.1: after $m_0 < m_{\max}$ steps of amplification, the resulting state is measured N times and the number S of measurements for which the ancillary qubit is in state $|1\rangle$ is counted.

If $S > \frac{N}{2}$, then the error is greater than ϵ and the network is updated using the measurements on the first n qubits. If on the other hand $S \leq \frac{N}{2}$, the process is repeated with $m_0 + 1 \leq m_{\max}$ steps of amplification.

The algorithm stops when the network reaches a state such that even after m_{\max} steps of amplification we have $S \leq \frac{N}{2}$. It is thus necessary to choose N such that when the algorithm stops the error is most probably lower than ε .

To avoid the angle $(2m_{\max} + 1)\theta_\varepsilon$ from overshooting $\frac{\pi}{4}$ by too much, we have chosen to limit ε to $]0, \frac{1}{10}[$. This limitation can be achieved without loss of generality by scaling the problem by a factor of $\frac{1}{5}$. This can be done by introducing a second ancillary qubit and applying a controlled rotation gate \mathbf{R} such that $\mathbf{R}|10\rangle = \frac{2}{\sqrt{5}}|10\rangle + \frac{1}{\sqrt{5}}|11\rangle$. The states of interest are then the ones for which the two ancillary qubits are in state $|11\rangle$. This means that for $\varepsilon \in]0, \frac{1}{2}[$, we are effectively working with $\varepsilon' = \frac{\varepsilon}{5} \in]0, \frac{1}{10}[$ as required. With this procedure, the effective error is itself limited to $[0, \frac{1}{5}]$. This also ensures that we have $\theta_{err} \in [0, \arcsin(\frac{1}{\sqrt{5}})]$ meaning that $\theta_{err} \neq \frac{\pi}{4}$ which is a stationary point of the amplitude amplification process so the error will always be amplified.

To account for this additional procedure, the diffusion operator has to be redefined. We now denote $\mathcal{A}(h) = (\mathbf{I}^{\otimes n} \otimes \mathbf{R})((\mathbf{TNN}(h)\mathbf{EX}(c,D)) \otimes \mathbf{I})$, $\mathcal{X}_G = \mathbf{CZ}(a_0, a_1)$, the controlled \mathbf{Z} gate for which the control is the first ancillary qubit and the target is the second ancillary qubit and $\mathcal{X}_0 = \mathbf{I} - 2|0\rangle\langle 0|$. These allow us to define the diffusion operator \mathbf{Q} in a similar way as previously with:

$$\mathbf{Q}(h) = -\mathcal{A}(h)\mathcal{X}_0\mathcal{A}^{-1}(h)\mathcal{X}_G$$

The corresponding circuit is represented in Figure 5 and the tuning algorithm is given in Algorithm 1.

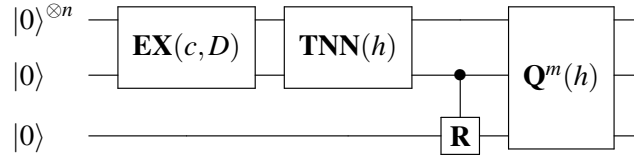


Figure 5: Circuit used to tune the network.

In Algorithm 1 the instruction "Update TNN according to errors" is given without further specification as the way it is done might depend on the class of concepts that is being learnt. The update strategy used to learn the class of concepts introduced in Section 6 is given in Algorithm 2

5 Proof and Analysis of the Algorithm

The algorithm stops when the network is in such a state that even after m_{\max} rounds of amplitude amplification, if S is the number of measurements for which the ancillary qubits are in state $|11\rangle$, then $S \leq \frac{N}{2}$ where N is the number of measurements. This stage is only reached if for all the m in the schedule we also have $S \leq \frac{N}{2}$ after m rounds of amplification. Let $|\Phi\rangle = \mathcal{A}|0\rangle$, then it can be written:

$$|\Phi\rangle = \cos(\theta_e)|\phi_\perp\rangle + \sin(\theta_e)|\phi_e\rangle|11\rangle$$

Where $|\phi_\perp\rangle$ is orthogonal to $|\phi_e\rangle|11\rangle$ and $|\phi_e\rangle$ contains the inputs that have been misclassified by the network. After m_{\max} rounds of amplification we have:

$$\mathbf{Q}^{m_{\max}}|\Phi\rangle = \cos((2m_{\max} + 1)\theta_e)|\phi_\perp\rangle + \sin((2m_{\max} + 1)\theta_e)|\phi_e\rangle|11\rangle$$

Algorithm 1: Tuning Algorithm

Data: $0 < \varepsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, $\mathbf{EX}(c, D)$
Result: TNN expressing h^* such that $P(\text{err}_D(h^*, c) < \varepsilon) > 1 - \delta$
 $N \leftarrow 2 \left(\lfloor \frac{1}{\pi \delta^2} \rfloor // 2 \right) + 2$;
 $m_{\max} \leftarrow$ smallest integer such that $(2m_{\max} + 1) \arcsin\left(\sqrt{\frac{\varepsilon}{5}}\right) \geq \frac{\pi}{4}$;
TNN \leftarrow **I**;
 $m \leftarrow -1$;
 $\text{errors} \leftarrow []$;
while $m < m_{\max}$ **or** $\text{length}(\text{errors}) > \frac{N}{2}$ **do**
 $m \leftarrow m + 1$;
 $\text{errors} \leftarrow []$;
 $\mathcal{A} \leftarrow (\mathbf{I}^{\otimes n} \otimes \mathbf{R})((\mathbf{TNN}.\mathbf{EX}(c, D)) \otimes \mathbf{I})$;
 $|\Phi\rangle \leftarrow \mathcal{A}|0\rangle$;
 $\mathbf{Q} \leftarrow -\mathcal{A}(h)\mathcal{X}_0\mathcal{A}^{-1}(h)\mathcal{X}_G$;
 for $1 \leq n \leq N$ **do**
 Measure $\mathbf{Q}^m|\Phi\rangle$;
 if *11 is measured on the ancillary qubits* **then**
 append the string of the first n qubits to errors
 end
 end
 if $\text{length}(\text{errors}) > \frac{N}{2}$ **then**
 Update **TNN** according to errors ;
 $m \leftarrow -1$;
 end
end

If p is the probability of measuring 11 on the ancillary qubits then $p = \sin^2((2m_{\max} + 1)\theta_e)$ and we want to compare it to $\frac{1}{2}$ in order to apply Lemma 4.1. This boils down to estimating $P(p < \frac{1}{2} \mid S \leq \frac{N}{2})$. We have:

$$P(p < 1/2 \mid S \leq N/2) = \frac{P(S \leq N/2 \mid p < 1/2)P(p < 1/2)}{P(S \leq N/2)}$$

By placing a uniform marginal distribution on p we get:

$$P(p < 1/2 \mid S \leq N/2) = \frac{\sum_{k=0}^{N/2} \binom{N}{k} \int_0^{\frac{1}{2}} \theta^k (1-\theta)^{N-k} d\theta}{\sum_{k=0}^{N/2} \binom{N}{k} \int_0^1 \theta^k (1-\theta)^{N-k} d\theta} \quad (5)$$

From now on we assume that N is even. For $a \in [0, 1]$ it can be shown by integrating by parts that:

$$\int_0^a \theta^k (1-\theta)^{N-k} d\theta = \frac{k!(N-k)!}{(N+1)!} - (1-a)^{N-k+1} \sum_{i=0}^k \frac{k!(N-k)!}{(k-i)!(N-k+i+1)!} a^{k-i} (1-a)^i$$

This leads to:

$$\int_0^1 \theta^k (1-\theta)^{N-k} d\theta = \frac{k!(N-k)!}{(N+1)!}$$

And

$$\sum_{k=0}^{N/2} \binom{N}{k} \int_0^1 \theta^k (1-\theta)^{N-k} d\theta = \frac{N+2}{2(N+1)} \quad (6)$$

Similarly we have:

$$\int_0^{\frac{1}{2}} \theta^k (1-\theta)^{N-k} d\theta = \frac{k!(N-k)!}{(N+1)!} - \left(\frac{1}{2}\right)^{N+1} \sum_{i=0}^k \frac{k!(N-k)!}{(k-i)!(N-k+i+1)!}$$

And

$$\sum_{k=0}^{N/2} \binom{N}{k} \int_0^{\frac{1}{2}} \theta^k (1-\theta)^{N-k} d\theta = \frac{N+2}{2(N+1)} - \left(\frac{1}{2}\right)^{N+1} \sum_{k=0}^{N/2} \sum_{i=0}^k \frac{N!}{(k-i)!(N-k+i+1)!} \quad (7)$$

Putting (5), (6) and (7) together yields:

$$\begin{aligned} P(p < 1/2 \mid S \leq N/2) &= 1 - \left(\frac{1}{2}\right)^N \frac{N+1}{N+2} \sum_{k=0}^{N/2} \sum_{i=0}^k \frac{N!}{(k-i)!(N-k+i+1)!} \\ &= 1 - \left(\frac{1}{2}\right)^N \frac{1}{N+2} \sum_{k=0}^{N/2} \sum_{i=0}^k \frac{(N+1)!}{(k-i)!(N-k+i+1)!} \\ &= 1 - \left(\frac{1}{2}\right)^N \frac{1}{N+2} \sum_{k=0}^{N/2} \sum_{i=0}^k \binom{N+1}{k-i} \\ &= 1 - \left(\frac{1}{2}\right)^N \frac{1}{N+2} \sum_{k=0}^{N/2} \sum_{i=0}^k \binom{N+1}{i} \\ &= 1 - \left(\frac{1}{2}\right)^N \frac{1}{N+2} \sum_{k=0}^{N/2} \binom{N+1}{k} \left(\frac{N}{2} + 1 - k\right) \end{aligned} \quad (8)$$

Now

$$\sum_{k=0}^{N/2} \binom{N+1}{k} \left(\frac{N}{2} + 1 - k\right) = \frac{N+2}{2} \sum_{k=0}^{N/2} \binom{N+1}{k} - \sum_{k=0}^{N/2} \binom{N+1}{k} k \quad (9)$$

For $k > 0$ we have:

$$\binom{N+1}{k} k = \binom{N}{k-1} (N+1)$$

And N being even, we also have:

$$\sum_{k=0}^{N/2} \binom{N+1}{k} = 2^N$$

Plugging these back into (9) we get:

$$\sum_{k=0}^{N/2} \binom{N+1}{k} \left(\frac{N}{2} + 1 - k\right) = 2^{N-1} (N+2) - (N+1) \sum_{k=0}^{\frac{N}{2}-1} \binom{N}{k}$$

Coming back to (8), we thus have:

$$P(p < 1/2 \mid S \leq N/2) = \frac{1}{2} + \left(\frac{1}{2}\right)^N \frac{N+1}{N+2} \sum_{k=0}^{\frac{N}{2}-1} \binom{N}{k} \quad (10)$$

Because N is even:

$$\sum_{k=0}^{\frac{N}{2}-1} \binom{N}{k} = \frac{1}{2} \left(2^N - \binom{N}{N/2} \right)$$

Together with (10), this leads to:

$$P(p < 1/2 \mid S \leq N/2) = \frac{1}{2} + \left(\frac{1}{2}\right)^{N+1} \frac{N+1}{N+2} \left(2^N - \binom{N}{N/2} \right)$$

But [8]:

$$\binom{N}{N/2} \leq \sqrt{2} \frac{2^N}{\sqrt{\pi N}}$$

So:

$$P(p < 1/2 \mid S \leq N/2) \geq \frac{1}{2} + \frac{N+1}{2(N+2)} \left(1 - \frac{\sqrt{2}}{\sqrt{\pi N}} \right)$$

Because $\frac{N+1}{N+2} \sim 1$ we look for a lower bound of the form $1 - \frac{\alpha}{\sqrt{N}}$. For example, we have:

$$P(p < 1/2 \mid S \leq N/2) > 1 - \frac{1}{\sqrt{\pi N}}$$

So for $0 < \delta < \frac{1}{2}$, in order to have $P(p < 1/2 \mid S \leq N/2) > 1 - \delta$, it suffices to take for N an even integer that is greater than $\frac{1}{\pi\delta^2}$, hence:

$$N = 2 \left(\left\lceil \frac{1}{\pi\delta^2} \right\rceil // 2 \right) + 2$$

We have shown that when the algorithm stops, we have:

$$P \left(\sin^2((2m_{\max} + 1)\theta_e) < \frac{1}{2} \right) > 1 - \delta$$

Together with Lemma 4.1, it comes that:

$$P(\theta_e < \theta_\varepsilon) > 1 - \delta$$

Hence:

$$P(\text{err} < \varepsilon) > 1 - \delta$$

So each sampling phase requires $\frac{1}{\pi\delta^2}$ calls to the oracle $\mathbf{EX}(c, d)$. In order to perform one update of the network, the algorithm requires at most $m_{\max} \approx \frac{1}{2} \left(\frac{\pi}{4 \arcsin(\sqrt{\varepsilon/5})} - 1 \right)$ of these sampling phases. Now for $\varepsilon \in]0, \frac{1}{2}[$, we have $\arcsin(\sqrt{\varepsilon/5}) \approx \sqrt{\varepsilon/5}$ so for one update of the network, the total number of call to $\mathbf{EX}(c, d)$ is $O\left(\frac{1}{\delta^2} \frac{1}{\sqrt{\varepsilon}}\right)$. The number of updates needed to reach the learning target is dependant on the class of concept.

It is possible to reduce the number of calls to $\mathbf{EX}(c, d)$ during one update phase by incrementing m not by 1 but with powers of a given number. In this case the total number for an update is $O\left(\frac{1}{\delta^2} \log\left(\frac{1}{\varepsilon}\right)\right)$ but this comes at the cost of possibly having a lower probability of success as we will see in Section 6.

6 Learning a Particular Class

Let $n \in \mathbb{N}$, for $s \in \mathbb{B}^n$, we define the parity function $p_s : x \mapsto s \cdot x = s_0 \cdot x_0 \oplus \dots \oplus s_{n-1} \cdot x_{n-1}$ and we are interested in learning the class of the parity functions \mathcal{C}_p :

$$\mathcal{C}_p = \{p_s \mid s \in \mathbb{B}^n\}$$

Any concept from this class can easily be expressed by a TNN: for each non-zero bit of s it suffices to apply the **X** gate controlled by the corresponding qubit. In order to learn a parity function, we are applying the update strategy shown in Algorithm 2. Because the gates to be updated are all controlled by a single qubit, we are assured that the final hypothesis will be a parity function.

Algorithm 2: Update strategy

Data: *errors* and *corrects* the lists of measured inputs that are respectively misclassified and correctly classified

Result: *gates* the list of gates to be updated

Gather the measurements in *errors* by group of same Hamming weight so that *errors*[i] is the list of misclassified inputs of Hamming weight i ;

Do the same with *corrects*;

gates \leftarrow *errors*[1];

for $1 \leq i \leq n - 1$ **do**

for e in *errors*[i] **do**

for c in *corrects*[$i + 1$] **do**

if $e \oplus c$ has Hamming weight 1 **then**

 Add $e \oplus c$ to *gates* if it is not already in;

end

end

end

for c in *corrects*[i] **do**

for e in *errors*[$i + 1$] **do**

if $c \oplus e$ has Hamming weight 1 **then**

 Add $c \oplus e$ to *gates* if it is not already in;

end

end

end

end

Return *gates*;

This approach has been implemented for $n = 4, 6$ and 8 using the Qiskit library¹. For a given n , a probability distribution over \mathbb{B}^n has been created randomly by applying to each qubit a **R_y** rotation gate with an angle randomly chosen in $[0, \pi]$. The controlled **X** gates corresponding to the target concept are then added to the circuit. These two blocks taken together are constituting the oracle **EX**(c, D). The construction of such oracles for $n = 4$ is illustrated in Figure 6. The TNN has then been trained to learn the concepts of the class \mathcal{C}_p according to the training algorithm and the update strategy introduced in

¹The code can be found in the following repository: <https://github.com/vietphamngoc/QPAC>

this paper. For $n = 4$ the experiments have been run for all the concepts of the class while for $n = 6$ and 8 the experiments have been performed for 16 randomly selected concepts from this class. In all cases different values of ε and δ . Once the training has been completed, the error is evaluated using the Qiskit statevector simulator by running the circuit composed of the oracle and the tuned network and getting the amplitude of all the inputs for which the network's output is wrong.

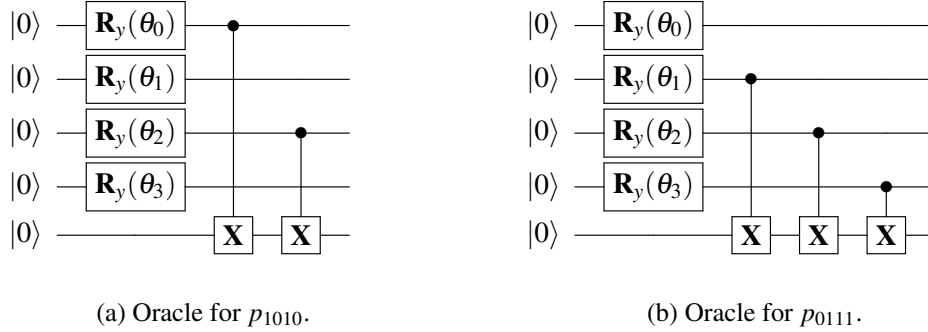


Figure 6: Implementation of the oracle for $n = 4$ and different target concepts.

To account for the overall randomness of the process, each training has been performed 50 times and the results are represented with violin plots where the width of the plot is proportional to their distribution within the repetitions. In Figure 7 are represented the results of some of these experiments. For given function and ε , in general, as δ decreases, the distribution of the final error rates tend to accumulate toward a minimum value, which was expected. Regarding the maximum value, we notice that in our set up, it never went above ε , except for the case $n = 6$, $\varepsilon = 0.05$ and $\delta = 0.1$ for the function p_{010001} as can be seen in Figure 7d. In this specific case the final error rate was greater than ε once in the 50 experiments. This event did not happen again for lower values of δ .

As said previously, the number of update steps necessary until a state with an error rate lower than ε depends on the class of concept. In the case of \mathcal{C}_p and with the experiments realised, we have found that the number of update steps decreases with decreasing δ . This can be explained by the fact that decreasing δ will increase the number of samples, hence the diversity of the measurements. This in turn allows for more accurate updates which result in less updates. This trend is depicted in Figure 8 where the number of updates is plotted in the cases $n = 4$ and 8 , $\varepsilon = 0.05$ and for different values of δ . Although there are few experiments, we can see that the number of updates does not significantly increase with n .

Finally, we wished to investigate how the increment schedule of m , the number of amplification round, could impact the behaviour of the tuning algorithm. In Algorithm 1, m was incremented by 1 after each sampling round. To speed-up the algorithm, the increment schedule was changed to increase m with powers of 2. We have compared the two schedule for $\varepsilon = 0.01$, that is $m \in [0, 9]$ against $m \in \{0, 1, 2, 4, 8, 9\}$, in the case $n = 6$ and with different values of δ . As previously, for each different value of δ the experiments have been repeated 50 times. The results of these experiments, as reported in Figure 9, show that incrementing m with powers of 2 yields similar performances compared to when m is incremented by 1. This means that the total number of calls to the oracle could be reduced from $O\left(\frac{1}{\delta^2} \frac{1}{\sqrt{\varepsilon}}\right)$ to $O\left(\frac{1}{\delta^2} \ln\left(\frac{1}{\varepsilon}\right)\right)$.

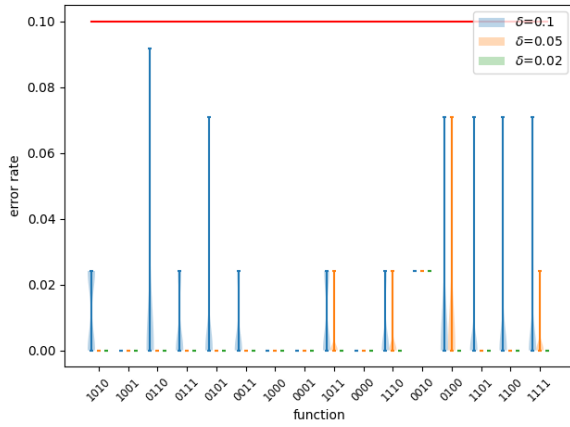
7 Conclusion

In this paper we have studied tunable quantum neural networks in the context of QPAC-learning. To do so, we have devised and proved a learning algorithm that uses quantum amplitude amplification. Amplitude amplification is used to both compare the error rate to the threshold ε and to better measure the errors. These measurements are then used to update the network. We have implemented this approach and tested it against the class of parity functions and found that this algorithm is indeed an efficient learner as its sample complexity is $O\left(\frac{1}{\delta^2} \frac{1}{\sqrt{\varepsilon}}\right)$ with a possible reduction to $O\left(\frac{1}{\delta^2} \ln\left(\frac{1}{\varepsilon}\right)\right)$. Experimental results show that most of the final error rates are quite far away from the threshold ε , which could be explained by an excessive number of samples. As future work we will look for a tighter lower bound for the probability of success which should reduce the dependence in $\frac{1}{\delta}$ in the algorithm's complexity. Finally we will study the generalisation of this approach to more complex classes of concepts.

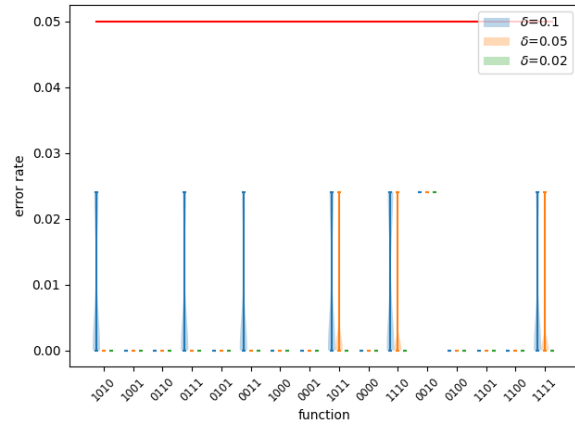
References

- [1] S. Aaronson & P. Rall (2020): *Quantum Approximate Counting, Simplified*. *Symposium on Simplicity in Algorithms*, p. 24–32, doi:10.1137/1.9781611976014.5.
- [2] S. Arunachalam & R. de Wolf (2017): *Guest Column: A Survey of Quantum Learning Theory*. *ACM SIGACT News* 48(2), pp. 41–67, doi:10.1145/3106700.3106710. Available at <http://dl.acm.org/citation.cfm?id=3106710>.
- [3] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin & H. Weinfurter (1995): *Elementary Gates for Quantum Computation*. *Physical review. A, Atomic, molecular, and optical physics* 52(5), pp. 3457–3467, doi:10.1103/PhysRevA.52.3457. Available at <https://search.proquest.com/docview/1859212069>.
- [4] Y. I. Bogdanov, N. A. Bogdanova, D. V. Fastovets & V. F. Lukichev (2019): *Representation of Boolean functions in terms of quantum computation*. In Vladimir F. Lukichev & Konstantin V. Rudenko, editors: *International Conference on Micro- and Nano-Electronics 2018*, 11022, International Society for Optics and Photonics; SPIE, p. 740, doi:10.1117/12.2522053.
- [5] G. Brassard, P. Hoyer, M. Mosca & A. Tapp (2002): *Quantum Amplitude Amplification and Estimation*, pp. 53–74. *Quantum computation and information* 305, Amer. Math. Soc., Providence, RI, doi:10.1090/conm/305/05215. 81P68 (81-02); 1947332.
- [6] N.H. Bshouty & J.C. Jackson (1998): *Learning DNF over the Uniform Distribution Using a Quantum Example Oracle*. *SIAM Journal on Computing* 28(3), pp. 1136–1153, doi:10.1137/S0097539795293123. Available at <https://search.proquest.com/docview/919130605>.
- [7] I. L. Chuang & M. A. Nielsen (2010): *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, doi:10.1017/CBO9780511976667. Available at <http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9780511994005&uid=none>.
- [8] T.M. Cover & J.A. Thomas (2005): *Elements of Information Theory*. John Wiley & Sons, Ltd, doi:10.1002/047174882X.
- [9] L. K. Grover (1996): *A Fast Quantum Mechanical Algorithm for Database Search*. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*, Association for Computing Machinery, New York, NY, USA, p. 212–219, doi:10.1145/237814.237866.
- [10] J. Haah, A. W. Harrow, Z. Ji, X. Wu & N. Yu (2017): *Sample-Optimal Tomography of Quantum States*. *IEEE Transactions on Information Theory* 63(9), pp. 5628–5641, doi:10.1109/TIT.2017.2719044.
- [11] K. Iwama, Y. Kambayashi & S. Yamashita (2002): *Transformation rules for designing CNOT-based quantum circuits*. In: *Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324)*, pp. 419–424, doi:10.1109/DAC.2002.1012662.

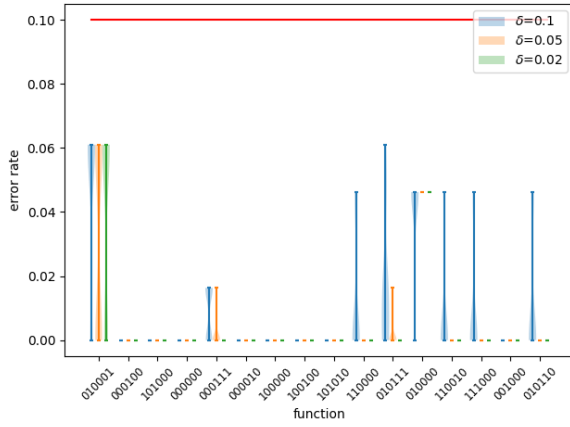
- [12] S. Kurgalin & S. Borzunov (2018): *The Discrete Math Workbook*. Springer International Publishing, Cham, doi:10.1007/978-3-319-92645-2.
- [13] S. Lloyd, M. Mohseni & P. Rebentrost (2013): *Quantum algorithms for supervised and unsupervised machine learning*. arXiv e-print. Available at <https://arxiv.org/abs/1307.0411>.
- [14] K. Nakaji (2020): *Faster Amplitude Estimation*. arXiv:2003.02417.
- [15] V. Pham Ngoc & H. Wiklicky (2020): *Tunable Quantum Neural Networks for Boolean Functions*. arXiv:2003.14122.
- [16] P. Rebentrost, T. R. Bromley, C. Weedbrook & S. Lloyd (2018): *Quantum Hopfield neural network*. *Physical Review A* 98(4), doi:10.1103/PhysRevA.98.042308.
- [17] J. E. Savage (2000): *Models of Computation*, reprinted with corr. edition. Addison-Wesley, Reading, Mass. [u.a.].
- [18] M. Schuld, I. Sinayskiy & F. Petruccione (2014): *The quest for a Quantum Neural Network*. *Quantum Information Processing* 13(11), pp. 2567–2586, doi:10.1007/s11128-014-0809-8. Available at <https://arxiv.org/abs/1408.7005>.
- [19] V. V. Shende, A. K. Prasad, I. L. Markov & J. P. Hayes (2003): *Synthesis of Reversible Logic Circuits*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(6), pp. 710–722, doi:10.1109/TCAD.2003.811448. Available at <https://ieeexplore.ieee.org/document/1201583>.
- [20] F. Tacchino, C. Macchiavello, D. Gerace & D. Bajoni (2019): *An Artificial Neuron Implemented on an Actual Quantum Processor*. *npj Quantum Information* 5(1), pp. 1–8, doi:10.1038/s41534-019-0140-4. Available at <https://search.proquest.com/docview/2199869978>.
- [21] L. Valiant (1984): *A theory of the learnable*. *Communications of the ACM* 27(11), pp. 1134–1142, doi:10.1145/1968.1972. Available at <http://dl.acm.org/citation.cfm?id=1972>.
- [22] S. Wallis (2013): *Binomial Confidence Intervals and Contingency Tests: Mathematical Fundamentals and the Evaluation of Alternative Methods*. *Journal of Quantitative Linguistics* 20(3), pp. 178–208, doi:10.1080/09296174.2013.799918. Available at <http://www.tandfonline.com/doi/abs/10.1080/09296174.2013.799918>.
- [23] A. Younes & J. Miller (2003): *Automated Method for Building CNOT Based Quantum Circuits for Boolean Functions*. arXiv e-prints, pp. quant-ph/0304099. Available at <https://ui.adsabs.harvard.edu/abs/2003quant.ph..4099Y>. Quant-ph/0304099; Provided by the SAO/NASA Astrophysics Data System.
- [24] A. Younes & J. F. Miller (2004): *Representation of Boolean Quantum Circuits as Reed-Muller Expansions*. *International Journal of Electronics* 91(7), pp. 431–444, doi:10.1080/00207210412331272643. Available at <http://www.tandfonline.com/doi/abs/10.1080/00207210412331272643>.



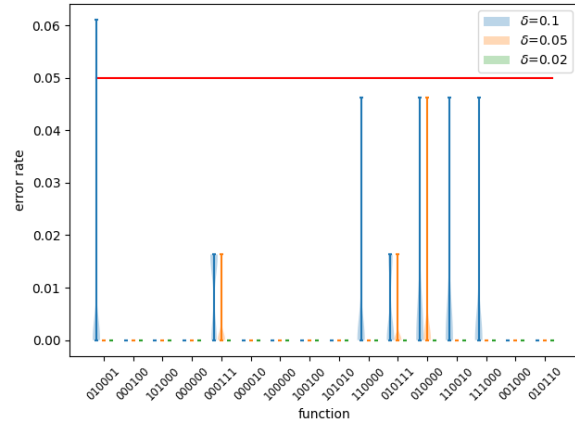
(a) $n = 4$ and $\epsilon = 0.1$



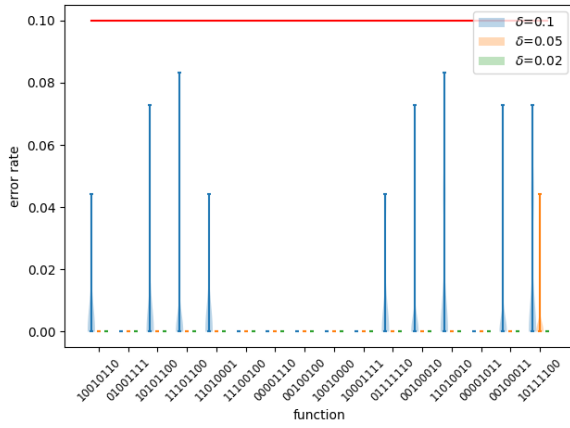
(b) $n = 4$ and $\epsilon = 0.05$



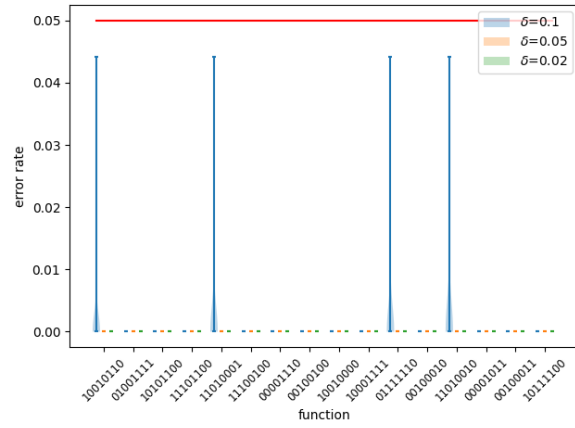
(c) $n = 6$ and $\epsilon = 0.1$



(d) $n = 6$ and $\epsilon = 0.05$

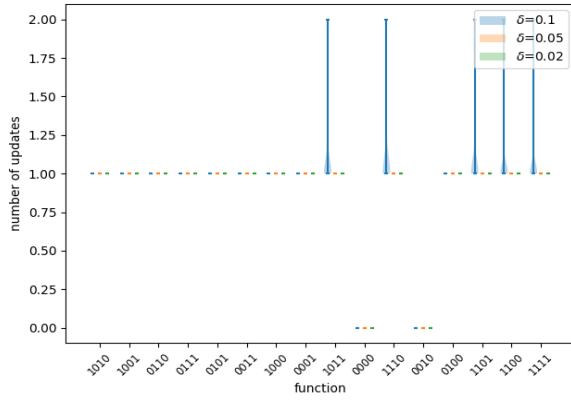


(e) $n = 8$ and $\epsilon = 0.1$

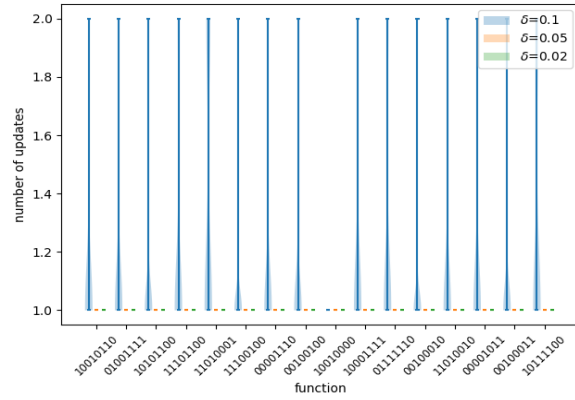


(f) $n = 8$ and $\epsilon = 0.05$

Figure 7: Final error rates for different values of n , ϵ and δ . The red line represents ϵ .

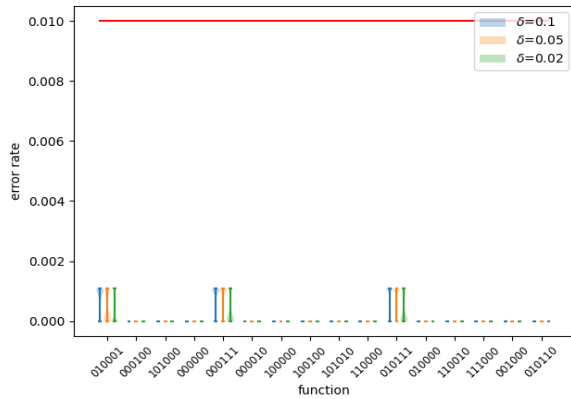


(a) $n = 4$

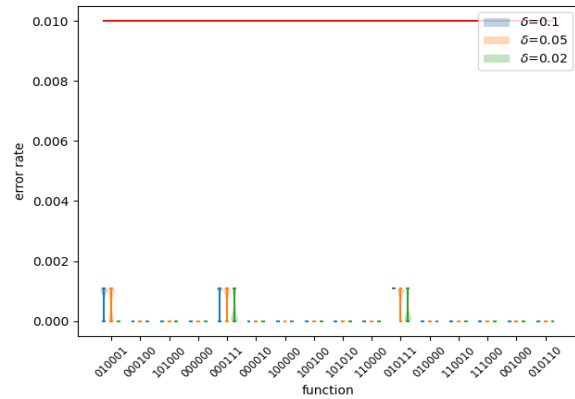


(b) $n = 8$

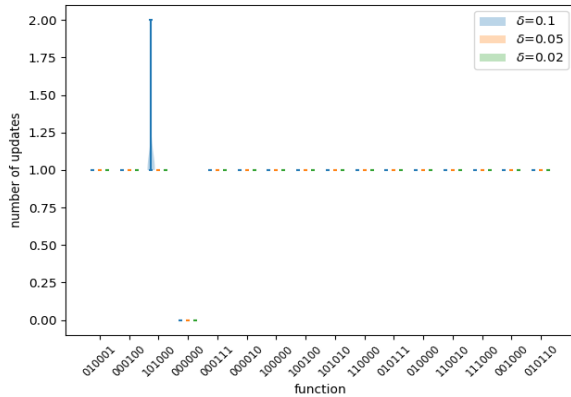
Figure 8: Number of updates for different values of n and $\epsilon = 0.05$



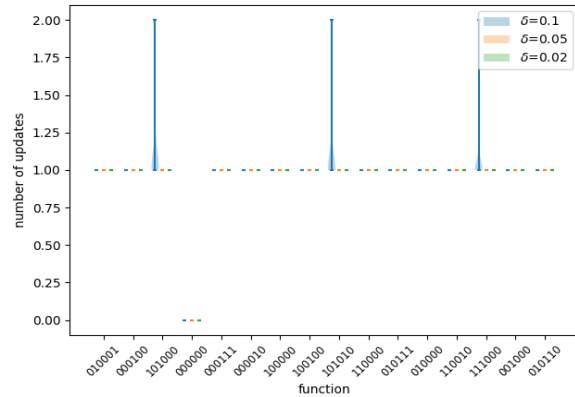
(a) m incremented by 1



(b) m incremented with powers of 2



(c) m incremented by 1



(d) m incremented with powers of 2

Figure 9: $n = 6$ and $\epsilon = 0.01$. Final error rate (Figures 9a and 9b) and number of updates (Figures 9c and 9d), for different values of δ , resulting from different increment schedules for m .

How to Sum and Exponentiate Hamiltonians in ZXW Calculus

Razin A. Shaikh

Quanlong Wang

Richie Yeung

Quantinuum, 17 Beaumont Street, Oxford OX1 2NA, United Kingdom

This paper develops practical summation techniques in ZXW calculus to reason about quantum dynamics, such as unitary time evolution. First we give a direct representation of a wide class of sums of linear operators, including arbitrary qubits Hamiltonians, in ZXW calculus. As an application, we demonstrate the linearity of the Schrödinger equation and give a diagrammatic representation of the Hamiltonian in Greene-Diniz et al [13], which is the first paper that models carbon capture using quantum computing. We then use the Cayley-Hamilton theorem to show in principle how to exponentiate arbitrary qubits Hamiltonians in ZXW calculus. Finally, we develop practical techniques and show how to do Taylor expansion and Trotterization diagrammatically for Hamiltonian simulation. This sets up the framework for using ZXW calculus to the problems in quantum chemistry and condensed matter physics.

1 Introduction

ZX calculus [4] is a graphical representation for quantum circuits and linear algebra in general: a diagram with n inputs and m outputs represents a $2^n \times 2^m$ linear map. ZX calculus comes with a complete set of rewrite rules such that two diagrams that represent the same linear map can always be rewritten to one another. Using this philosophy of *computation via rewriting*, ZX calculus has been successfully applied to a wide range of problems in quantum computing, including circuit compilation [1, 7, 8, 20], circuit equality validation [21, 24], circuit simulation [22], error correction [2, 3], natural language processing [5, 25] and quantum machine learning [30, 38, 39].

To solve a problem using ZX calculus, or other diagrammatic calculi, we first need to synthesise the expressions involved into diagrams. This can be done in an ad-hoc way or using general methods such as elementary matrices [36]. For example: in Hamiltonian simulation problems [26] the Hamiltonian is typically given as a sum of Pauli operators, and without an efficient representation of such Hamiltonians in ZX calculus, there is simply no way to proceed with the problem.

As of this writing, there is no published work on how to efficiently combine sums of ZX diagrams. Both [18] and [32] first require an inefficient conversion step to an intermediate form before the diagrams can be summed together. Furthermore, the resulting diagram is large and does not resemble the original symbolic expression, so the advantage of using diagrammatic reasoning is diminished.

To obtain intuitive sums of diagrams, we introduce the framework of *ZXW calculus*. Coecke and Kissinger [6] proposed the idea of developing a graphical calculus based on the interaction between GHZ and W states. Following up on that, Hadzihasanovic [15, 16] developed the ZW calculus. The ZW-calculus has found applications in linear optical quantum computing [9, 17], describing interactions in quantum field theory [29], and studying multi-partite entanglement [6]. In this paper, we combine the ZX and ZW calculi to create the *ZXW calculus*. The W-spider of the ZXW calculus plays an important

role in producing efficient and compact sums of diagrams. We define the ZXW calculus through a slight modification of the algebraic ZX calculus [33].

In this work, we modify and extend the notion of controlled states [19] to give direct representations of controlled diagrams for a wide class of matrices and show how to sum them within the framework of ZXW calculus. These representations, combined with the recently developed techniques of differentiating arbitrary ZX diagrams [18, 37], allow us to *practically reason about analytical problems that were previously inaccessible to ZX calculus*. Moreover, the addition, exponentiation and differentiation operations interact nicely, allowing us to work with ZX diagrams as naturally as the traditional matrix notation, while still keeping the compact representation and rewriting advantages of the ZX calculus.

Specifically, these techniques allow us to reason about quantum dynamics and quantum chemistry in the ZXW calculus. Previous diagrammatic approaches [10, 11, 12] to quantum dynamics have tackled the problem from an abstract categorical perspective. In this paper, we develop diagrammatic techniques to tackle problems of quantum chemistry and condensed matter physics in a concrete and practical way. We first devise a diagrammatic form for arbitrary Hamiltonians in the ZXW calculus. One of the main problems in quantum computational chemistry is that of Hamiltonian simulation: given a Hamiltonian, find an approximation to its unitary time evolution. The ideal evolution is given by $e^{-iHt/2}$, where H is the Hamiltonian and t is time. Using the diagrammatic form of Hamiltonian and the summation techniques developed in this paper, we show how to diagrammatically exponentiate Hamiltonians. This allows us to write the unitary time evolution graphically and apply the ZXW rewrite rules to extract a quantum circuit for Hamiltonian simulation.

Summary of results

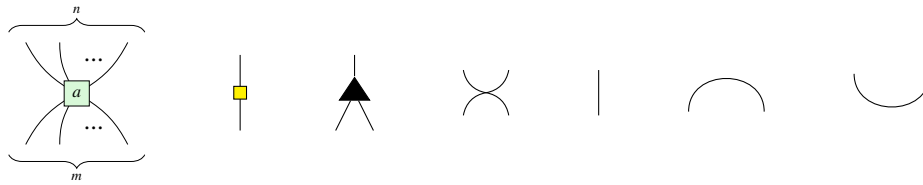
1. **Representing sums of square matrices and arbitrary vectors:** We show how to represent a sum of any square matrices of size $2^m \times 2^m$ or any vectors of size 2^m in ZXW calculus (Section 3 and Section 4). As an application, We formulate the Schrödinger equation in ZXW calculus and show the linearity of its solutions. (Section 5)
2. **Representing arbitrary Hamiltonians:** We show how to construct a diagram for any Hamiltonian defined on arbitrary number of qubits (Section 6). As an example, we express the Hamiltonian used in Greene-Diniz et. al. [13] (Figure 1).
3. **Representing Taylor expansion and Trotterization:** We show how Taylor expansion and Trotterization used for practical Hamiltonian exponentiation can be realised in ZXW calculus. (Section 7)

2 ZXW Calculus

In this section, we give an introduction to ZXW calculus which is a slight modification of the algebraic ZX calculus [33], including its generators and rewriting rules. Note that algebraic ZX calculus is complete [33] for matrices of size $2^m \times 2^n$ and hence, so is the ZXW calculus. In this paper diagrams are either read from left to right or top to bottom.

2.1 Generators

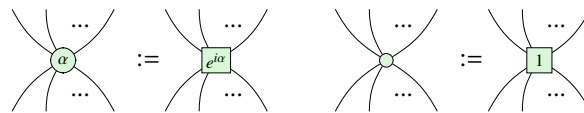
The diagrams in ZXW calculus are defined by freely combining the following generating objects. Note that a can be any complex number.



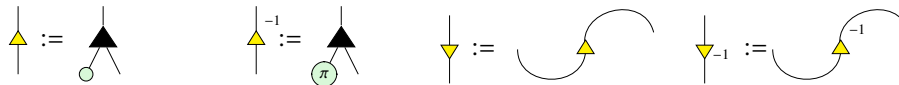
2.2 Additional notation

For simplicity, we introduce additional notation based on the given generators:

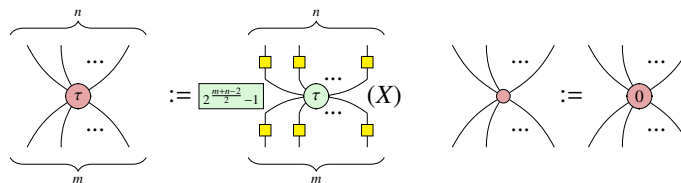
1. The green spider from the original ZX calculus can be defined using the green box spider in ZXW calculus.



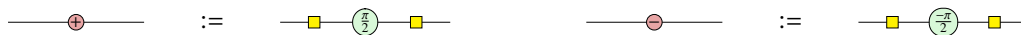
2. The triangle and the inverse triangle can be expressed as follows. The transposes of the triangle and the inverse triangle can be drawn as inverted triangles.



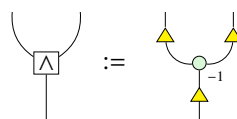
3. The red spider from the original ZX calculus can be defined by performing Hadamard conjugation on each leg of the green spider, and the pink spider is the algebraic equivalent of the red spider. It is only defined for $\tau \in \{0, \pi\}$, and is rescaled to have integer components in its matrix representation.



4. The plus gate (also known as V gate) and minus gate (also known as V^\dagger gate) can be used for constructing the Pauli Y gate, they are defined as $V = HSH$ and $V^\dagger = HS^\dagger H$ respectively:

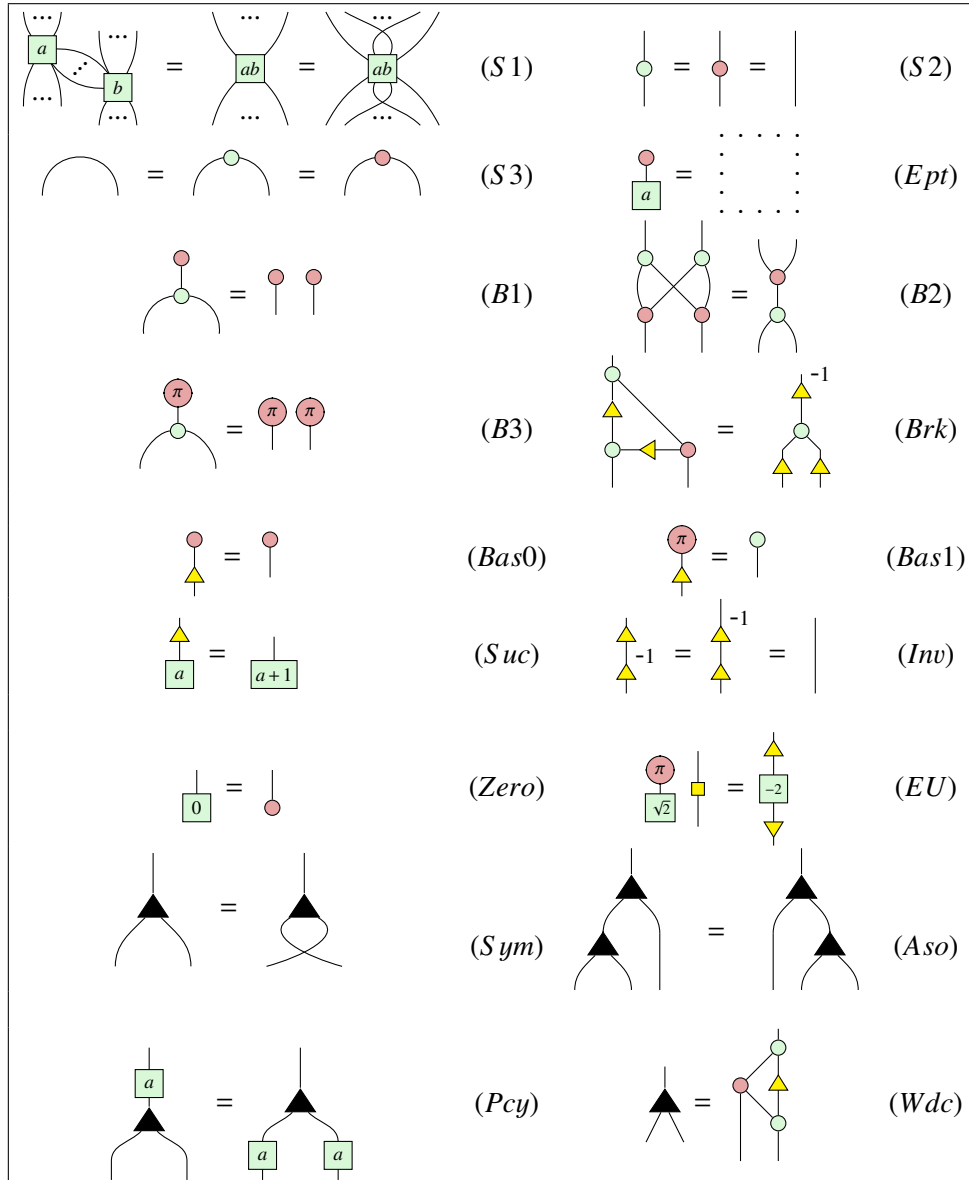


5. The “and” box is defined as the following.



2.3 Rules

Now we give the rewriting rules of ZXW calculus.



where $a, b \in \mathbb{C}$. The vertically flipped versions of the rules are assumed to hold as well.

2.4 Interpretation

Although the generators in ZXW calculus are formal mathematical objects in their own right, in the context of this paper we interpret the generators as linear maps, so each ZXW diagram is equivalent to a

vector or matrix.

$$\begin{array}{c} \overbrace{\quad\quad\quad}^n \\ \diagdown \quad \dots \quad \diagup \\ \text{---} \boxed{a} \text{---} \\ \diagup \quad \dots \quad \diagdown \\ \underbrace{\quad\quad\quad}_m \end{array} = |0\rangle^{\otimes m} \langle 0|^{\otimes n} + a |1\rangle^{\otimes m} \langle 1|^{\otimes n} \quad \text{---} \boxed{\text{yellow}} \text{---} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \blacktriangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\text{---} \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \text{---} \quad \text{---} \text{---} \text{---} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{---} \text{---} \text{---} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \text{---} \text{---} \text{---} = (1 \ 0 \ 0 \ 1),$$

Remark 2.1. Due to the associative rule (Aso), we can define the W spider and give its interpretation as follows [34]:

$$\begin{array}{c} \blacktriangle \\ \diagdown \quad \dots \quad \diagup \\ \dots \end{array} := \begin{array}{c} \blacktriangle \\ \diagdown \quad \dots \quad \diagup \\ \dots \\ \blacktriangle \\ \diagdown \quad \dots \quad \diagup \\ \dots \end{array} \quad \begin{array}{c} \blacktriangle \\ \diagdown \quad \dots \quad \diagup \\ \dots \\ \underbrace{\quad\quad\quad}_m \end{array} = \underbrace{|0 \dots 0\rangle}_{m} \langle 0| + \sum_{k=1}^m \underbrace{|0 \dots 0 1 0 \dots 0\rangle}_{k-1} \langle 1|.$$

As a consequence, we have

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

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

3 Controlled diagrams

We start by stating the definitions of controlled states and matrices, and how to perform operations on them. Note that our definition of controlled states are slightly different from that of Jeandel et al [18], as we send $|0\rangle$ to $|0\rangle^{\otimes n}$ instead of $|+\rangle^{\otimes n}$. They also have a notion of controlled matrices by applying the map-state duality to the controlled state, which maps $|0\rangle$ to $|+\rangle^{\otimes n} \langle +|^{\otimes n}$ instead of the identity matrix as used in our definition, so their definition is not equivalent to ours.

Definition 3.1 (Controlled matrix). The controlled matrix \widetilde{M} corresponding to the matrix M is a diagram

$$\begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{M}} \text{---} \\ \vdots \end{array} \quad \text{such that} \quad \begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{M}} \text{---} \\ \vdots \end{array} = \text{---} \text{---} \text{---} \quad \text{and} \quad \begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{M}} \text{---} \\ \vdots \end{array} = \begin{array}{c} \text{---} \\ \text{---} \boxed{M} \text{---} \\ \vdots \end{array}$$

Definition 3.2 (Controlled state). The controlled state \widetilde{V} corresponding to the state V is a diagram

$$\begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{V}} \text{---} \\ \vdots \end{array} \quad \text{such that} \quad \begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{V}} \text{---} \\ \vdots \end{array} = \begin{array}{c} \text{---} \\ \text{---} \text{---} \text{---} \\ \vdots \end{array} \quad \text{and} \quad \begin{array}{c} \text{---} \\ \text{---} \boxed{\widetilde{V}} \text{---} \\ \vdots \end{array} = \begin{array}{c} \text{---} \\ \text{---} \boxed{V} \text{---} \\ \vdots \end{array}$$

Now, we show how to construct control diagrams for sums and products of matrices, given the controlled diagrams for the original matrices.

Proposition 3.3 (Controlled product of matrices). *Given controlled matrices $\widetilde{M}_1, \dots, \widetilde{M}_k$ corresponding to matrices M_1, \dots, M_k , the controlled matrix for $\prod_i M_i$ is given by*

$$\text{Diagram (2): } \prod_i \widetilde{M}_i = \widetilde{M}_1 \dots \widetilde{M}_k \tag{2}$$

Proposition 3.4 (Controlled sum of matrices). *Given controlled matrices $\widetilde{M}_1, \dots, \widetilde{M}_k$ corresponding to matrices M_1, \dots, M_k and complex numbers c_1, \dots, c_k , the controlled matrix for $\sum_i c_i M_i$ is given by*

$$\text{Diagram (3): } \sum_i c_i \widetilde{M}_i = \widetilde{M}_1 \dots \widetilde{M}_k \tag{3}$$

Both of these can be verified by simply plugging in the standard basis states. Similarly, we can construct the controlled diagram for sums of states.

Proposition 3.5 (Controlled sum of states). *Given controlled states $\widetilde{V}_1, \dots, \widetilde{V}_k$ corresponding to states V_1, \dots, V_k and complex numbers c_1, \dots, c_k , the controlled state for $\sum_i c_i V_i$ is given by*

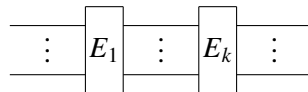
$$\text{Diagram (4): } \sum_i c_i \widetilde{V}_i = \widetilde{V}_1 \dots \widetilde{V}_k \tag{4}$$

4 Realising controlled diagrams

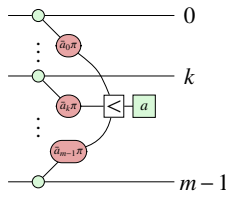
In this section, we show how to directly construct the controlled matrix \widetilde{M} given a square matrix M , and the controlled state $\widetilde{\psi}$ given a state ψ .

4.1 Controlled matrix

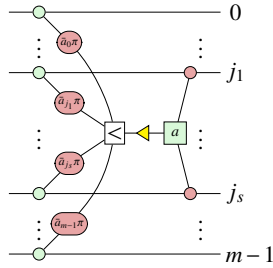
Consider a matrix M of size $2^m \times 2^m$. As given in [36], M can be represented in diagrams as follows:



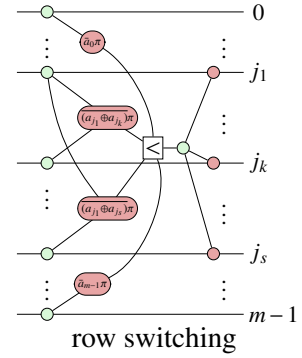
where $E_i, 1 \leq i \leq k$ is an elementary matrix. As a consequence of Proposition 3.3, if we know how to construct any controlled elementary matrix, then we are able to depict the controlled matrix M . It has been shown in [36] that the elementary matrix E_i must be of one of the following forms:



row multiplication



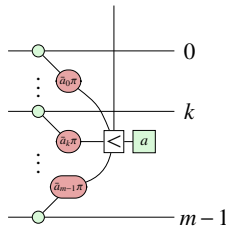
row addition



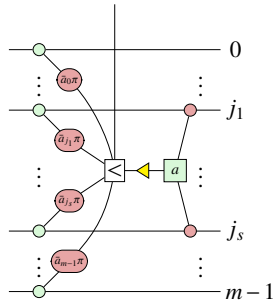
row switching

Then it can be verified by plugging standard basis that their corresponding controlled matrices can be obtained by simply adding a branch to the And-gate:

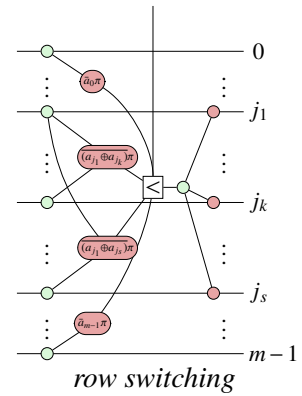
Proposition 4.1. *The controlled elementary matrices are given as:*



row multiplication



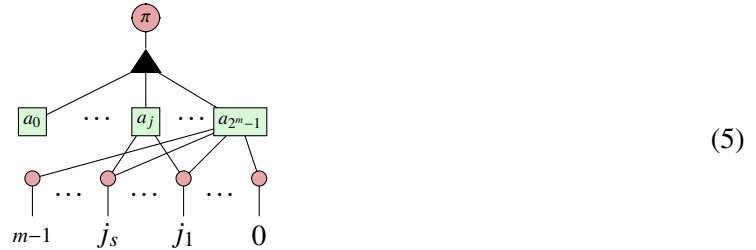
row addition



row switching

4.2 Controlled state

According to Wang [32, 35], a state vector $(a_0, \dots, a_{2^m-1})^T$ of dimension 2^m , $m \in \mathbb{N}$, can be represented in the following normal form:



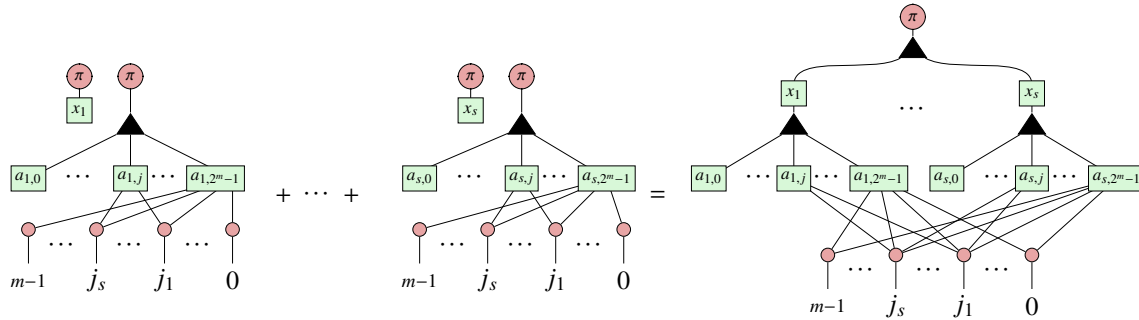
Hence, we can realise any controlled state by constructing the controlled diagram of the above normal form. This is given in the following proposition.

Proposition 4.2. *Controlled state for a vector $(a_0, \dots, a_{2^m-1})^T$ is given by*



Using the above result and Proposition 3.5, we can derive the expression for linear combination of states, represented in the normal form.

Proposition 4.3.



5 Schrödinger Equation

As an application of the results from the previous sections, we prove the linearity of the solutions of the Schrödinger equation. We recall the Schrödinger equation:

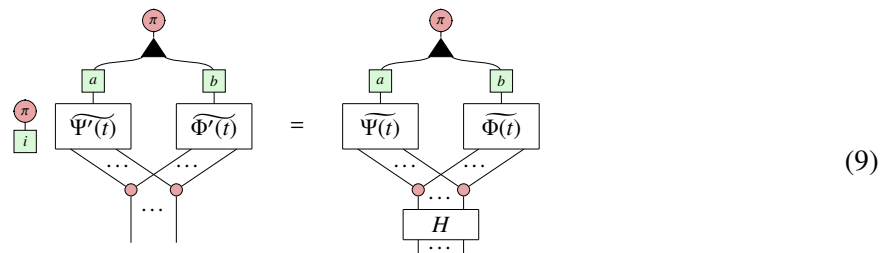
$$i \frac{\partial}{\partial t} |\Psi(t)\rangle = H |\Psi(t)\rangle \tag{7}$$

where t is time, $|\Psi(t)\rangle$ is the state vector of the quantum system in question, and H is a Hamiltonian operator. We can write the Schrödinger equation diagrammatically as:



where $|\Psi(t)\rangle$ can be expressed using the normal form and $|\Psi'(t)\rangle$ can be obtained by applying the differentiation gadget [37] to $|\Psi(t)\rangle$. Along with this, we use Proposition 4.3 to show that any linear combination of solutions of Schrödinger equation is also a solution.

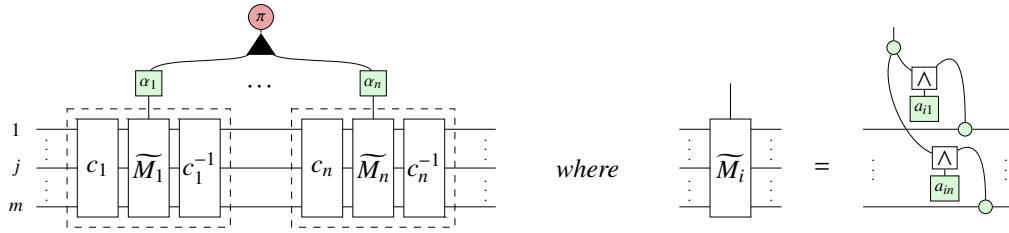
Proposition 5.1. Assume that $\Psi(t)$ and $\Phi(t)$ satisfy the Schrödinger equation (7) and a, b are arbitrary complex numbers, then so does $a\Psi(t) + b\Phi(t)$, i.e.,



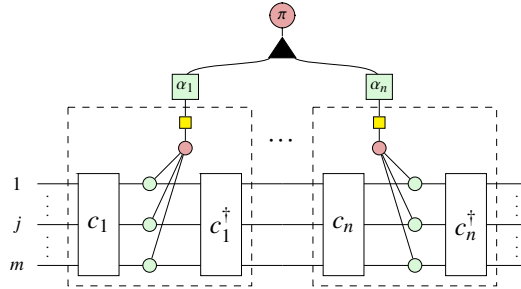
6 Representing Hamiltonians in ZXW

Here we give an efficient representation for a wide class of matrices using the ZXW calculus.

Lemma 6.1. Any matrix of the form $\sum_{i=1}^n \alpha_i c_i^{-1} \left(\bigotimes_{j=1}^m D(a_{ij}) \right) c_i$, where $D(a_{ij}) = |0\rangle\langle 0| + a_{ij}|1\rangle\langle 1|$, c_i is the conjugation, and α_i is a complex coefficient, can be expressed in ZXW calculus as

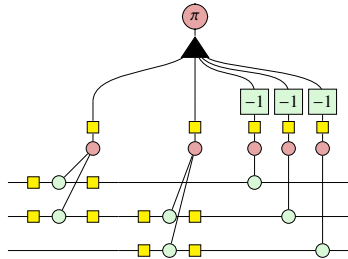


Theorem 6.2. Any Hamiltonian $\sum_{i=1}^n \alpha_i \otimes_{j=1}^m P_{ij}$ can be expressed in ZXW calculus using controlled-Paulis.



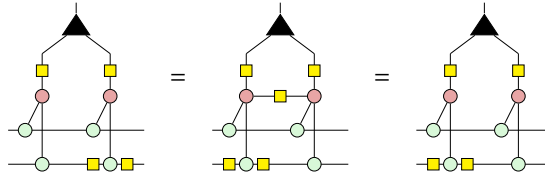
For each controlled-Pauli, there is a leg on the j -th qubit if $P_{ij} \neq I$, and c_i is the Clifford conjugation corresponding to the Pauli operator $P_{ij} = c_i^\dagger Z_{ij} c_i$.

Example 6.3. For the Hamiltonian $H = X_1 X_2 + X_2 X_3 - Z_1 - Z_2 - Z_3$, we have



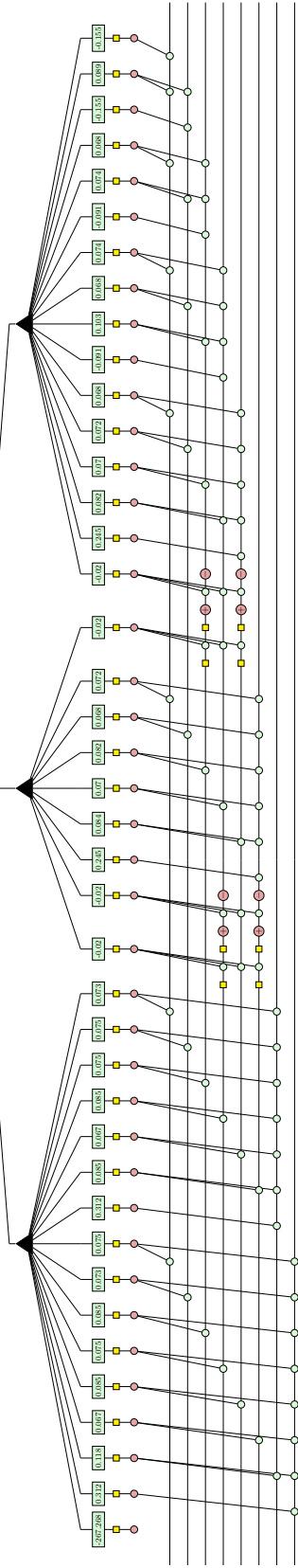
For an even larger example, the Hamiltonian used in [13] is shown in Figure 1.

Proposition 6.4. The diagrammatic representation of controlled sum of Hamiltonians, thus the sum of Hamiltonians, in Theorem 6.2 respects commutativity of addition (i.e. $\widehat{P_i + P_j} = \widehat{P_j + P_i}$):



While this proposition is obvious in non-diagrammatic calculations, our goal here is to diagrammatically characterise the commutative properties of controlled matrices.

1. Hamiltonian truncated using `hamiltonian.compress(abs_tol=0.019)`, 42 terms:



2. Hamiltonian truncated using `hamiltonian.compress(abs_tol=0.008)`, 70 terms:

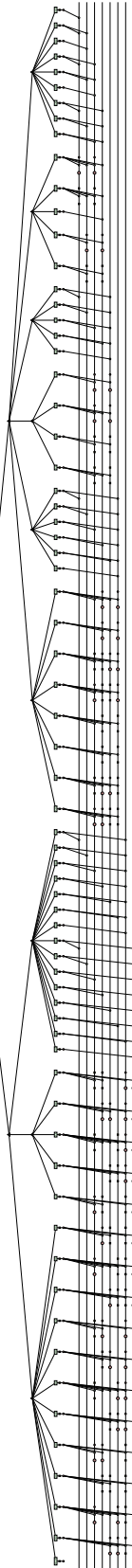


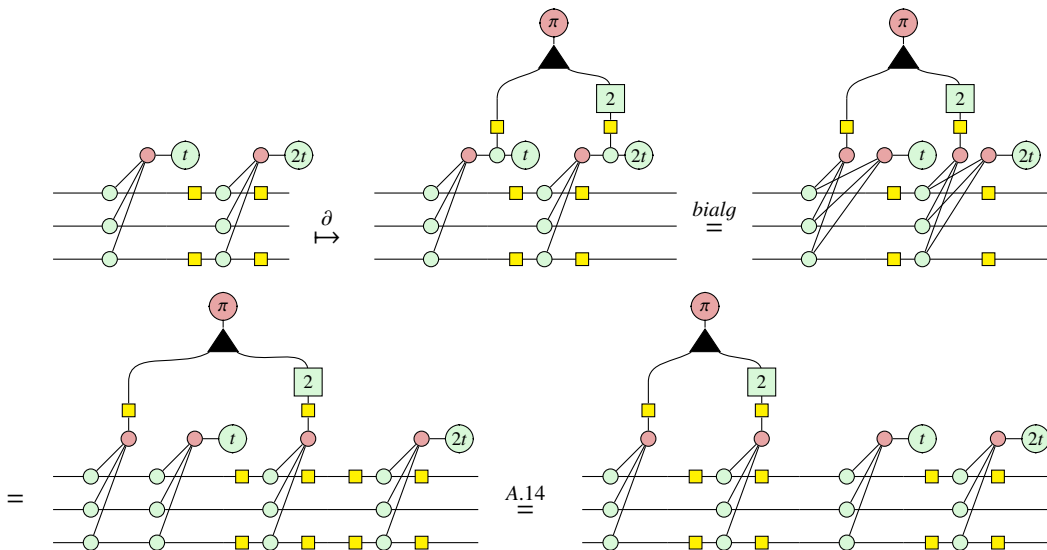
Figure 1: Hamiltonian in Greene-Diniz et al. [13]

7 Hamiltonian Exponentiation

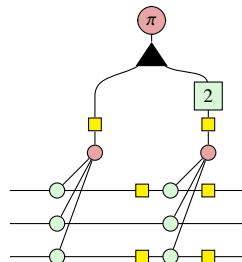
According to Stone’s theorem [31], the time-evolution operator of a Hamiltonian is the one-parameter unitary group (OPUG) generated by the Hamiltonian. Tasks such as Hamiltonian simulation require finding a unitary circuit that approximates the time evolution operator $e^{-iHt/2}$ for a given Hamiltonian H . In general, finding such unitary circuit is a hard problem. Developing a method to obtain the one-parameter unitary group of Hamiltonian diagrams in ZXW will allow us tackle problems from quantum chemistry and condensed matter physics using diagrammatic tools. We could further use the rules of ZXW calculus to rewrite a Hamiltonian exponential diagram to a unitary time-evolution circuit. In this section, we first talk about the exponentiation of Hamiltonians containing only commuting Pauli terms. We subsequently describe the general case where the Hamiltonian may contain non-commuting terms.

7.1 Hamiltonians with only commuting terms

For Hamiltonians comprised of only commuting Pauli terms, we have a simple correspondence between the diagrams of Hamiltonians and their exponentials. In the following example, we show how to obtain the Hamiltonian from its one-parameter unitary group. We begin by writing the exponential of Pauli strings of the Hamiltonian using the phase gadgets [7]. Then we differentiate the exponential diagram and set t to 0: $e^{-iHt/2} \xrightarrow{\partial} \frac{-i}{2} H e^{-iHt/2} \xrightarrow{t=0} \frac{-i}{2} H$. As an example, consider the Hamiltonian $H = ZZZ + 2XZX$. We will omit the global phase in the diagrams for simplicity.



We see from the above that the Hamiltonian diagrammatically commutes with its exponential. We set $t = 0$ to get the Hamiltonian:



In the following table, we show the correspondence between a few more Hamiltonians H and their OPUGs $\Phi_H(t)$.

$Z \leftrightarrow \Phi_Z(t)$	$ZXY \leftrightarrow \Phi_{ZXY}(t)$
$3(XZY) - (ZZX) \leftrightarrow \Phi_{XZY}(3t) \Phi_{ZZX}(-t)$	

For a more detailed reference of Pauli/phase gadgets and how to differentiate them, see [7] and [23] respectively.

7.2 General case

In most of the interesting examples, the Hamiltonian contains some non-commuting terms. Constructing an exact circuit for the OPUG of such Hamiltonians is difficult. We will use the Cayley-Hamilton theorem [14, 27] to construct an exact diagram for the OPUG of the Hamiltonian. But calculating the coefficients in such diagram is computationally hard. Hence, we will present approximate diagrams of the OPUGs via Taylor expansion and Trotterization, which we can use in practical applications.

7.2.1 Exact exponential diagram

To give an exact diagram for the exponential, we can use the Cayley-Hamilton theorem [14, 27]. For a $n \times n$ matrix of the Hamiltonian H , we can express its exponential as

$$e^{-iHt/2} = c_0(t)I + c_1(t)H + \dots + c_{n-1}(t)H^{n-1} \tag{10}$$

where c_0, \dots, c_{n-1} are some functions of t . Using this equation, we present the exact form of the Hamiltonian exponential function in ZXW:

$$\tag{11}$$

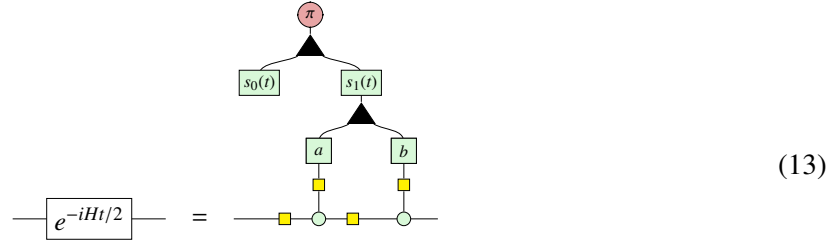
The coefficients c_0, \dots, c_{n-1} can be calculated using Putzer's algorithm [28], but its computational complexity is exponential in the number of qubits.

Now, we demonstrate the utility of the ZXW calculus by using its rules to rewrite a Hamiltonian exponential diagram to a quantum circuit.

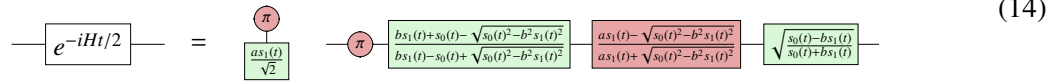
Example 7.1. Consider the following Hamiltonian:

$$H = aX + bZ \tag{12}$$

for any complex a and b . There exists $s_0(t)$ and $s_1(t)$ such that



Using ZXW, we can rewrite it to the following circuit. The full simplification steps are shown in Appendix B.

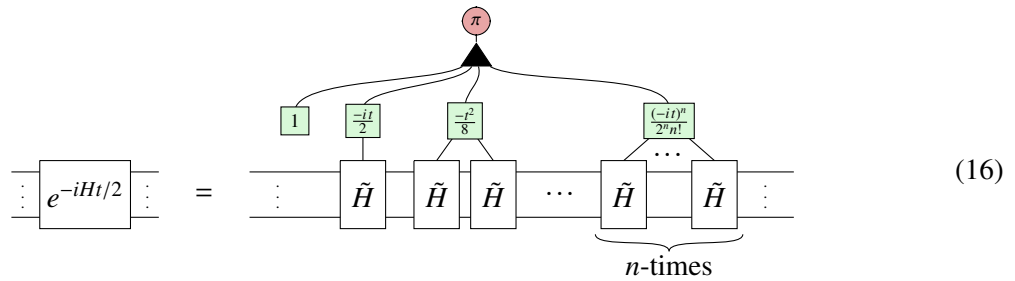


7.2.2 Taylor expansion

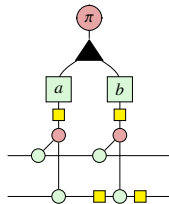
We consider Taylor expansion of the exponential to give a simple approximation of the OPUG in terms of ZXW diagram. The expansion is given as:

$$e^{-iHt/2} = \sum_{k=0}^{\infty} \left(\frac{-it}{2}\right)^k \frac{H^k}{k!} \tag{15}$$

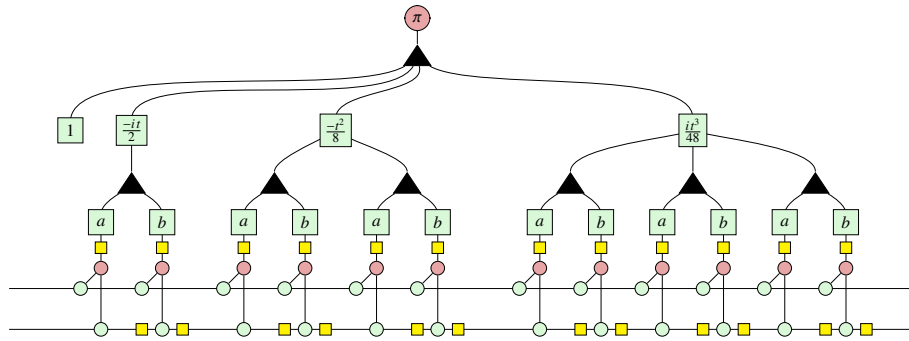
To construct a ZXW diagram, we need to take a finite number of terms from the expansion. Suppose, we take the n -th order approximation of the exponential. We can write it as the following diagram.



Example 7.2. Consider the Hamiltonian $H = aZ_1Z_2 + bZ_1X_2$. The diagram for this Hamiltonian is



The third order approximation of $e^{-iHt/2}$ is the following diagram



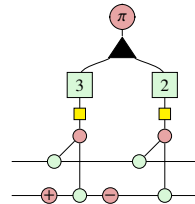
7.2.3 Trotterization

Using the Trotter-Suzuki formula, we can approximate the time evolution operator of the Hamiltonian by evolving the system in small time-steps. With small enough time-steps, we can treat the non-commuting terms in the Hamiltonian as commuting terms. Hence, for a Hamiltonian $H = \sum_k H_k$, we get

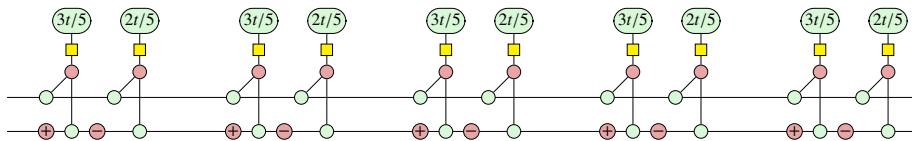
$$e^{-iHt/2} = \left(\prod_k e^{-iH_k t/2n} \right)^n + O\left(\frac{t^2}{n}\right) \tag{17}$$

where n is the number of Trotter steps. This allows us to apply the techniques developed in Section 7.1 to construct the diagram for the OPUG of Hamiltonian.

Example 7.3. Consider the Hamiltonian $H = 3ZY + 2ZZ$ represented by the following diagram.



By approximating $e^{-iHt/2}$ with 5 Trotter steps, we obtain the diagram shown below.



8 Future work

In this paper, we give direct representations of controlled diagrams for a wide class of matrices and show how to sum them. As applications, we express any Hamiltonian with a form of a sum of Pauli operators, including the Hamiltonians used for carbon capture in [13], convert between a Hamiltonian and its one-parameter unitary group, and represent Taylor expansion and Trotterization used for practical Hamiltonian exponentiation in ZXW calculus.

We would like to apply the summation techniques developed in this paper to practical problems, like quantum approximate optimization, integration on arbitrary ZX diagrams, or quantum machine learning. Also, we would like to develop tools to rewrite Hamiltonian exponentiation diagrams to quantum circuits.

Acknowledgements

We would like to thank Gabriel Greene-Diniz and David Zsolt Manrique for providing the Pauli operators form of the Hamiltonian from their paper [13]. We would also like to thank Bob Coecke, Pablo Andres-Martinez, Boldizsár Poór, Lia Yeh, Stefano Gogioso, Konstantinos Meichanetzidis, Vincent Wang and Robin Lorenz for the feedback on this paper.

References

- [1] 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, pp. 11:1–11:23, doi:10.4230/LIPIcs.TQC.2020.11.
- [2] 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, pp. 55:1–55:15, doi:10.4230/LIPIcs.MFCS.2019.55.
- [3] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren & Dominic Horsman (2016): *Graphical Structures for Design and Verification of Quantum Error Correction*.
- [4] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. *New Journal of Physics* 13(4), doi:10.1088/1367-2630/13/4/043016.
- [5] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis & Alexis Toumi (2020): *Foundations for Near-Term Quantum Natural Language Processing*.
- [6] Bob Coecke & Aleks Kissinger (2010): *The Compositional Structure of Multipartite Quantum Entanglement*. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide & Paul G. Spirakis, editors: *Automata, Languages and Programming*, pp. 297–308, doi:10.1007/978-3-642-14162-1_25.
- [7] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons & Seyon Sivarajah (2020): *Phase Gadget Synthesis for Shallow Circuits*. *Electronic Proceedings in Theoretical Computer Science* 318, p. 213?228, doi:10.4204/eptcs.318.13.
- [8] 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.
- [9] Giovanni de Felice & Bob Coecke (2022): *Quantum Linear Optics via String Diagrams*, doi:10.48550/arXiv.2204.12985.
- [10] Stefano Gogioso (2015): *Categorical Semantics for Schrödinger's Equation*.
- [11] Stefano Gogioso (2017): *Categorical Quantum Dynamics*. Ph.D. thesis, University of Oxford. arXiv:1709.09772.
- [12] Stefano Gogioso (2019): *A Diagrammatic Approach to Quantum Dynamics*. In Markus Roggenbach & Ana Sokolova, editors: *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019), Leibniz International Proceedings in Informatics (LIPIcs)* 139, pp. 19:1–19:23, doi:10.4230/LIPIcs.CALCO.2019.19.
- [13] Gabriel Greene-Diniz, David Zsolt Manrique, Wassil Sennane, Yann Magnin, Elvira Shishenina, Philippe Cordier, Philip Llewellyn, Michal Krompiec, Marko J Rančić & David Muñoz Ramo (2022): *Modelling carbon capture on metal-organic frameworks with quantum computing*. *EPJ Quantum Technology* 9(1), p. 37, doi:10.1140/epjqt/s40507-022-00155-w.
- [14] Werner Greub (1975): *Linear algebra*, 4 edition. Graduate Texts in Mathematics, Springer, doi:10.1007/978-1-4684-9446-4.
- [15] Amar Hadzihasanovic (2015): *A Diagrammatic Axiomatisation for Qubit Entanglement*. In: *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 573–584, doi:10.1109/LICS.2015.59.

- [16] Amar Hadzihasanovic (2017): *The algebra of entanglement and the geometry of composition*. Ph.D. thesis, University of Oxford. arXiv:1709.08086.
- [17] Amar Hadzihasanovic, Giovanni de Felice & Kang Feng Ng (2018): *A Diagrammatic Axiomatisation of Fermionic Quantum Circuits*. In H el ene Kirchner, editor: *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, LIPIcs 108, pp. 17:1–17:20, doi:10.4230/LIPIcs.FSCD.2018.17.
- [18] Emmanuel Jeandel, Simon Perdrix & Margarita Veshchezerova (2022): *Addition and Differentiation of ZX-diagrams*.
- [19] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2019): *A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pp. 1–10, doi:10.1109/LICS.2019.8785754.
- [20] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. *Electronic Proceedings in Theoretical Computer Science* 318, pp. 229–241, doi:10.4204/eptcs.318.14.
- [21] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Phys. Rev. A* 102, p. 022406, doi:10.1103/PhysRevA.102.022406.
- [22] Aleks Kissinger, John van de Wetering & Renaud Vilmart (2022): *Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions*. In Fran ois 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, Dagstuhl, Germany, pp. 5:1–5:13, doi:10.4230/LIPIcs.TQC.2022.5.
- [23] Mark Koch (2022): *Quantum Machine Learning using the ZXW-Calculus*. arXiv preprint arXiv:2210.11523.
- [24] Louis Lemonnier, John van de Wetering & Aleks Kissinger (2021): *Hypergraph Simplification: Linking the Path-sum Approach to the ZH-calculus*. *Electronic Proceedings in Theoretical Computer Science*, doi:10.4204/EPTCS.340.10.
- [25] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis & Bob Coecke (2023): *QNL 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.
- [26] Sam McArdle, Suguru Endo, Al an Aspuru-Guzik, Simon C. Benjamin & Xiao Yuan (2020): *Quantum computational chemistry*. *Rev. Mod. Phys.* 92, p. 015003, doi:10.1103/RevModPhys.92.015003.
- [27] H ector Manuel Moya-Cessa & Francisco Soto-Eguibar (2011): *Differential equations: an operational approach*. Rinton Press, Incorporated.
- [28] Eugene J Putzer (1966): *Avoiding the Jordan canonical form in the discussion of linear systems with constant coefficients*. *The American Mathematical Monthly* 73(1), pp. 2–7, doi:10.2307/2313914.
- [29] Razin A Shaikh & Stefano Gogioso (2022): *Categorical Semantics for Feynman Diagrams*. arXiv preprint arXiv:2205.00466, doi:10.48550/arXiv.2205.00466.
- [30] Tobias Stollenwerk & Stuart Hadfield (2022): *Diagrammatic Analysis for Parameterized Quantum Circuits*.
- [31] Marshall H Stone (1932): *On one-parameter unitary groups in Hilbert space*. *Annals of Mathematics*, pp. 643–648, doi:10.2307/1968538.
- [32] Quanlong Wang (2020): *Algebraic complete axiomatisation of ZX-calculus with a normal form via elementary matrix operations*. doi:10.48550/arXiv.2007.13739.
- [33] Quanlong Wang (2021): *An Algebraic Axiomatisation of ZX-calculus*. *Electronic Proceedings in Theoretical Computer Science* 340, pp. 303–332, doi:10.4204/eptcs.340.16.
- [34] Quanlong Wang (2021): *A non-anyonic qudit ZW-calculus*. doi:10.48550/arXiv.2109.11285.
- [35] Quanlong Wang (2022): *Qufinite ZX-calculus: A Unified Framework of Qudit ZX-calculi*, doi:10.48550/arXiv.2104.06429.

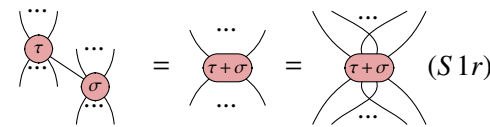
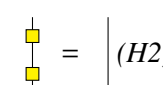
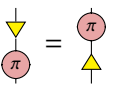

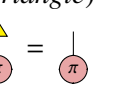
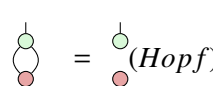
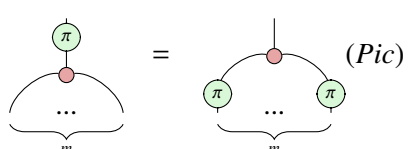
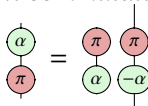
- [36] Quanlong Wang & Richie Yeung (2021): *Representing Matrices Using Algebraic ZX-calculus*. arXiv preprint arXiv:2110.06898, doi:10.48550/arXiv.2110.06898.
- [37] Quanlong Wang, Richie Yeung & Mark Koch (2022): *Differentiating and Integrating ZX Diagrams with Applications to Quantum Machine Learning*, doi:10.48550/arXiv.2201.13250.
- [38] Richie Yeung (2020): *Diagrammatic Design and Study of Ansatzes for Quantum Machine Learning*. arXiv preprint arXiv:2011.11073, doi:10.48550/arXiv.2011.11073.
- [39] Chen Zhao & Xiao-Shan Gao (2021): *Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus*. Quantum 5, p. 466, doi:10.22331/q-2021-06-04-466.

A Proofs and Lemmas

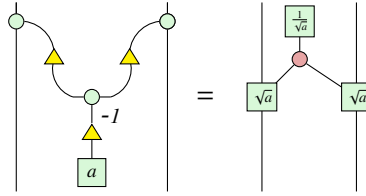
In this appendix, we include all the lemmas which have been essentially existed (up to scalars) in previous papers. The lemmas are given in the order which they appear in this paper.

A.1 Useful lemmas

We present several useful results in the following table.

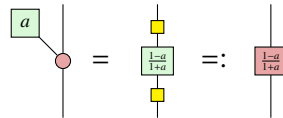
<p>Lemma A.1. [32] For $\tau, \sigma \in \{0, \pi\}$, pink spiders fuse.</p> 	<p>Lemma A.2. [32] Hadamard is involutive.</p> 
<p>Lemma A.3. [32] Pink π transposes the triangle.</p> 	<p>Lemma A.4. [32] Green π inverts the triangle.</p> 
<p>Lemma A.5. [32] (triangle)^T stabilises 1⟩.</p> 	<p>Lemma A.6. [32] Hopf rule.</p> 
<p>Lemma A.7. [32] π copy rule. For $m \geq 0$:</p> 	<p>Lemma A.8. [32] π commutation rule.</p> 

Lemma A.9. Suppose $a \neq 0, a \in \mathbb{C}$. Then



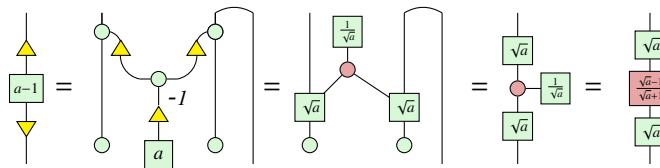
This equality can be verified by plugging in the standard basis on the inputs of the diagrams. Next, we have

Lemma A.10.

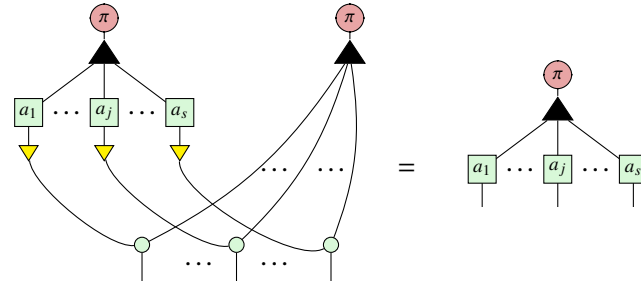


As a consequence of the above two lemmas, we have

Lemma A.11.

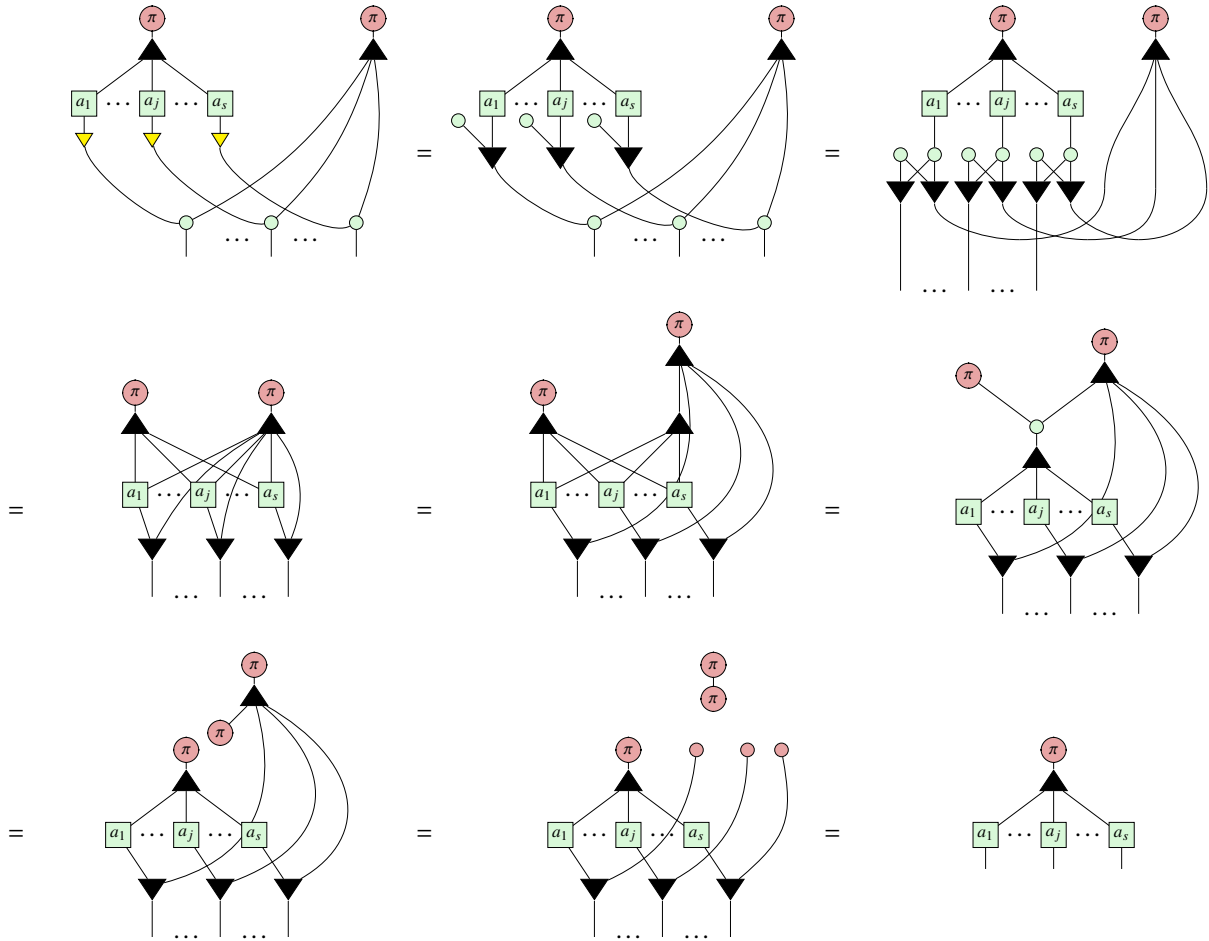


Lemma A.12.



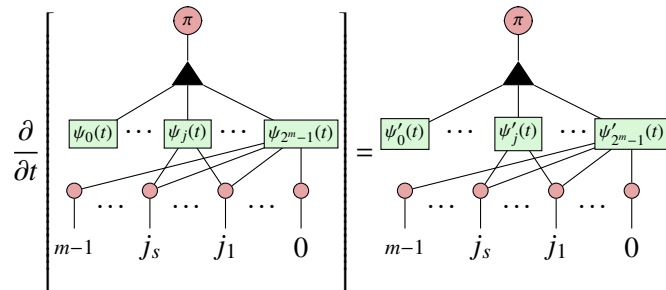
where a_1, \dots, a_s are arbitrary complex numbers.

Proof.

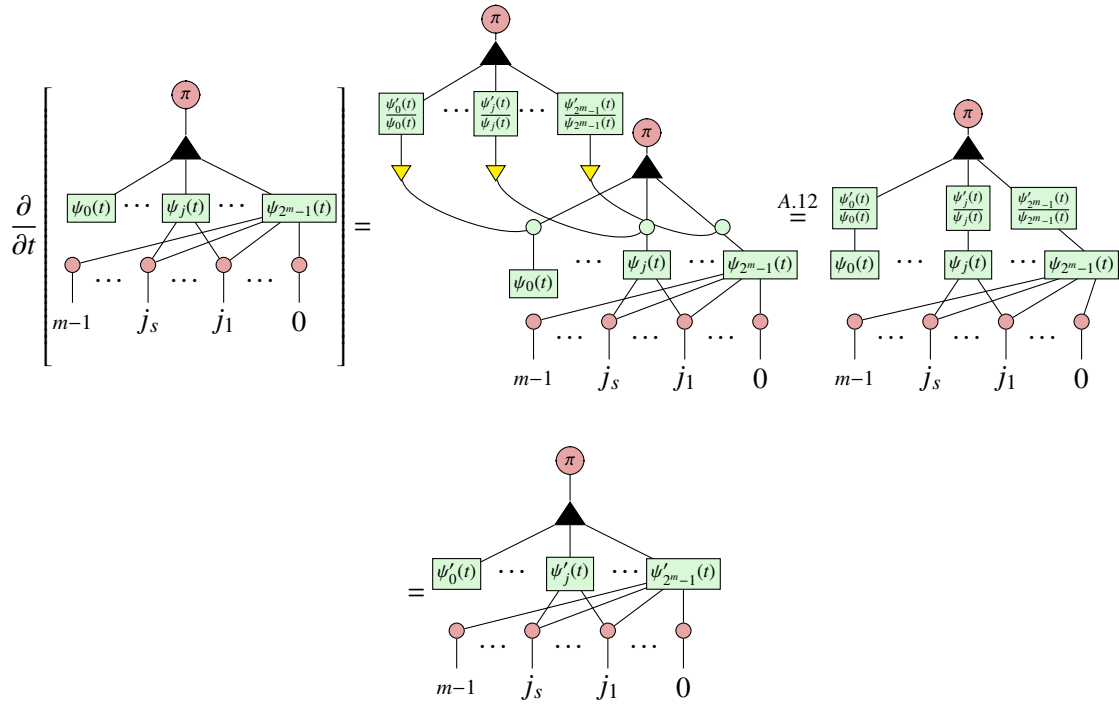


□

Lemma A.13.

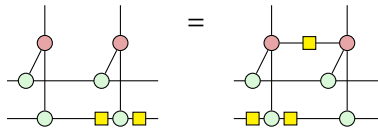


Proof.



□

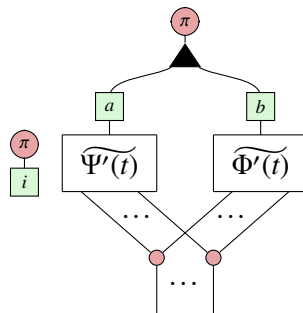
Lemma A.14.

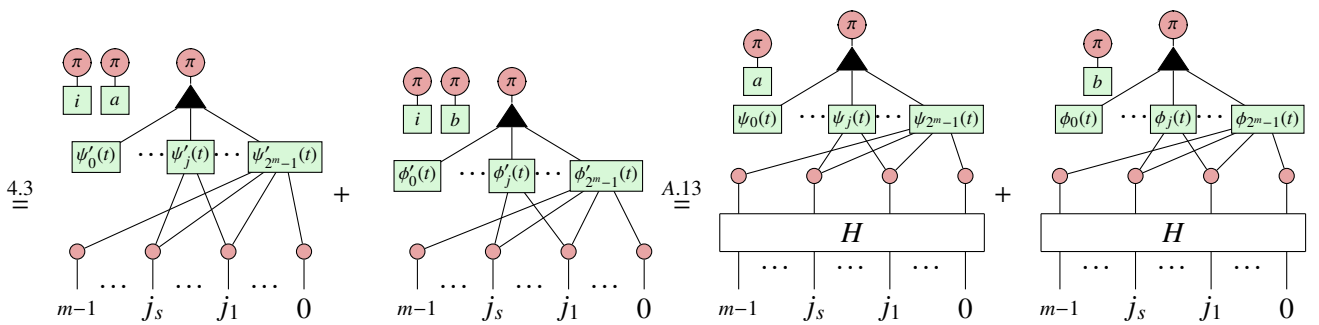
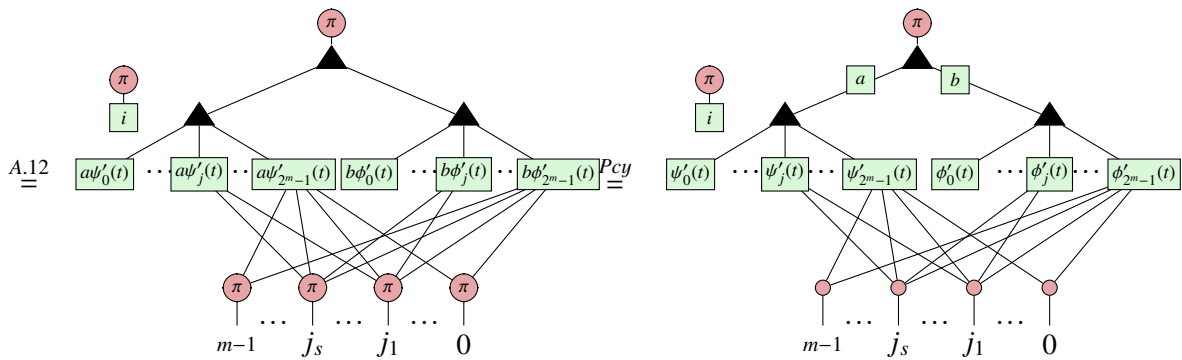
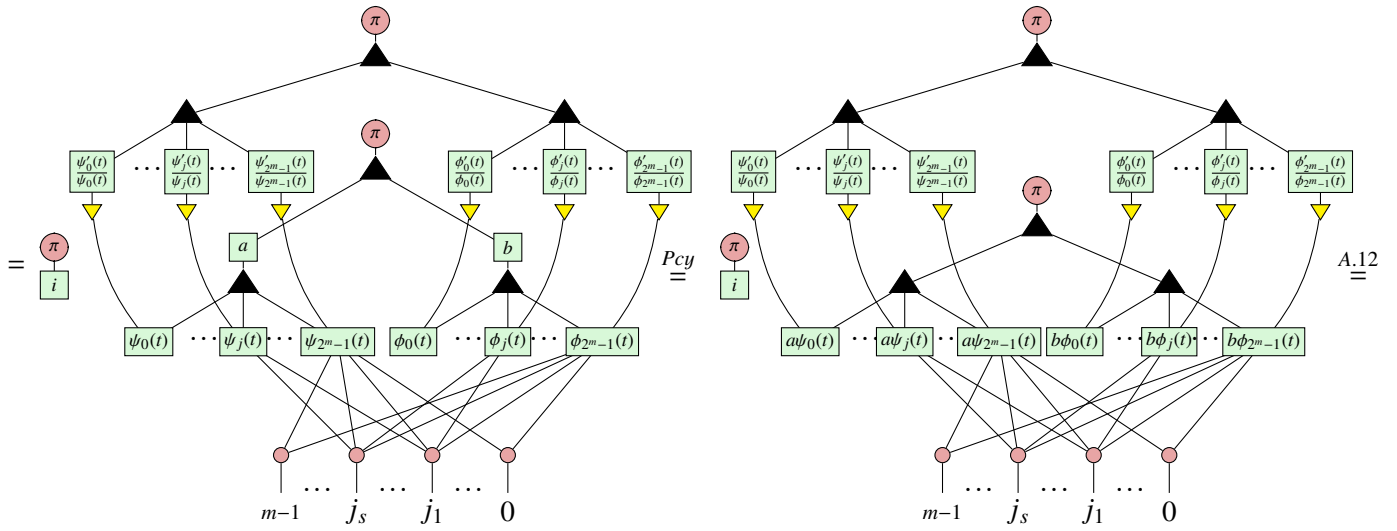


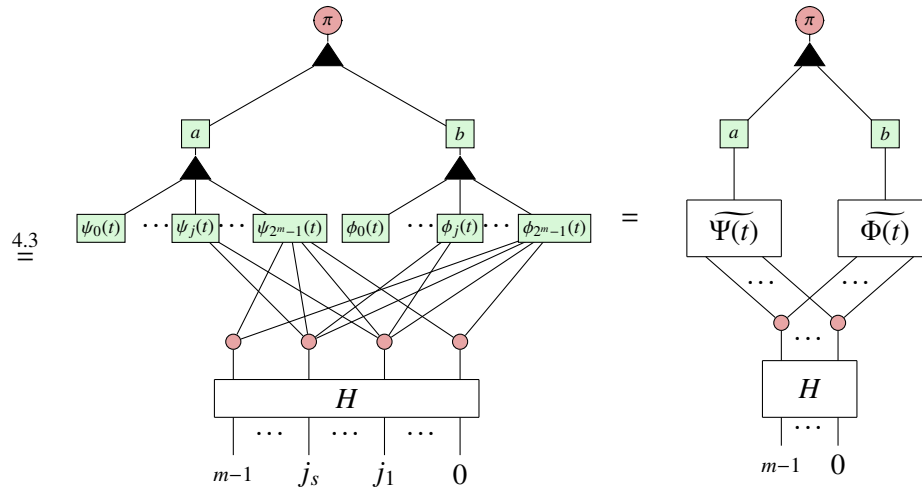
This shows that a Hadamard edge is added to the ‘body’ of the Pauli gadgets when their Hamiltonians anti-commute, the proof can be found in [38, Theorem 3].

A.2 Proofs

Proof of Proposition 5.1.

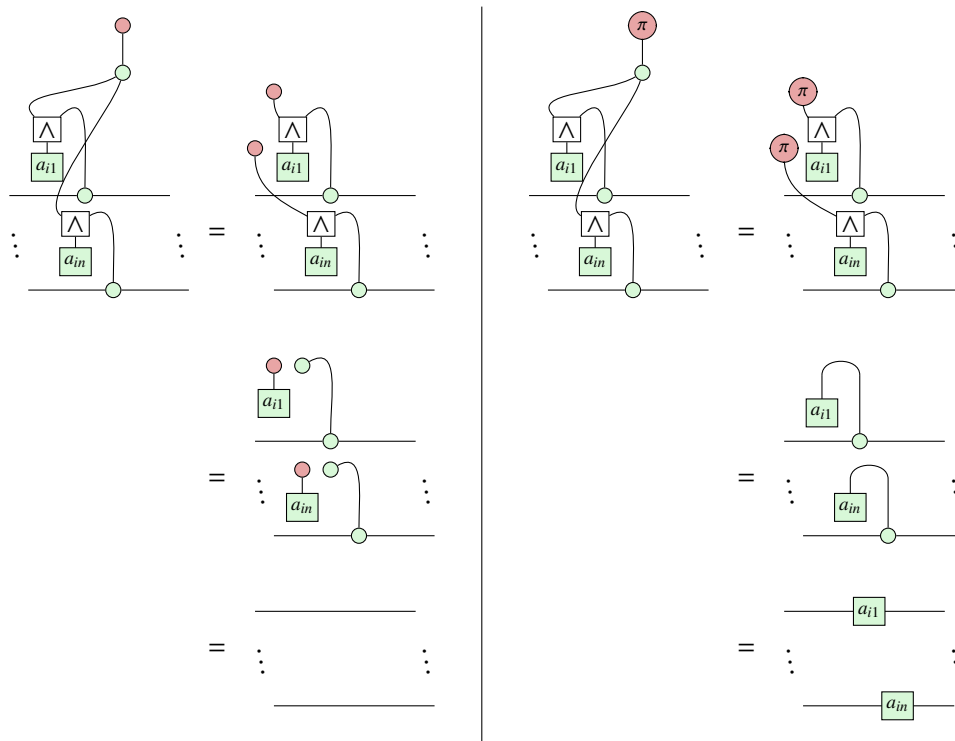






□

Proof of Lemma 6.1. We need check that the controlled matrix \tilde{M}_i represents the controlled matrix of $\bigotimes_{j=1}^m D(a_{ij})$. After that, the rest of the proof follows from Proposition 3.4.

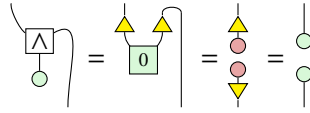


□

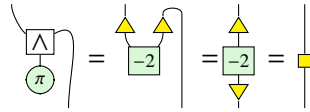
Proof of Theorem 6.2. In the Lemma 6.1, we set

$$a_{ij} = \begin{cases} e^{i0} & \text{if } P_{ij} = I \\ e^{i\pi} & \text{otherwise.} \end{cases}$$

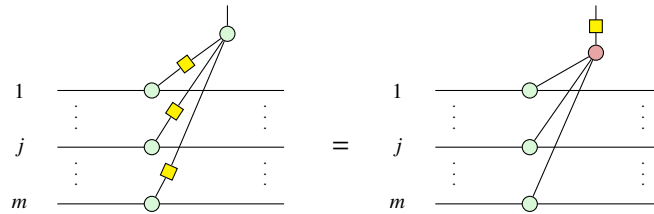
When $a_{ij} = e^{i0}$, the leg on the j -th qubit will be disconnected:



On the other hand, for $a_{ij} = e^{i\pi}$, we get

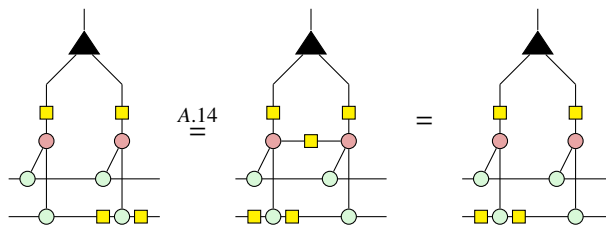


Substituting the above in the controlled matrix \tilde{M}_i , we get

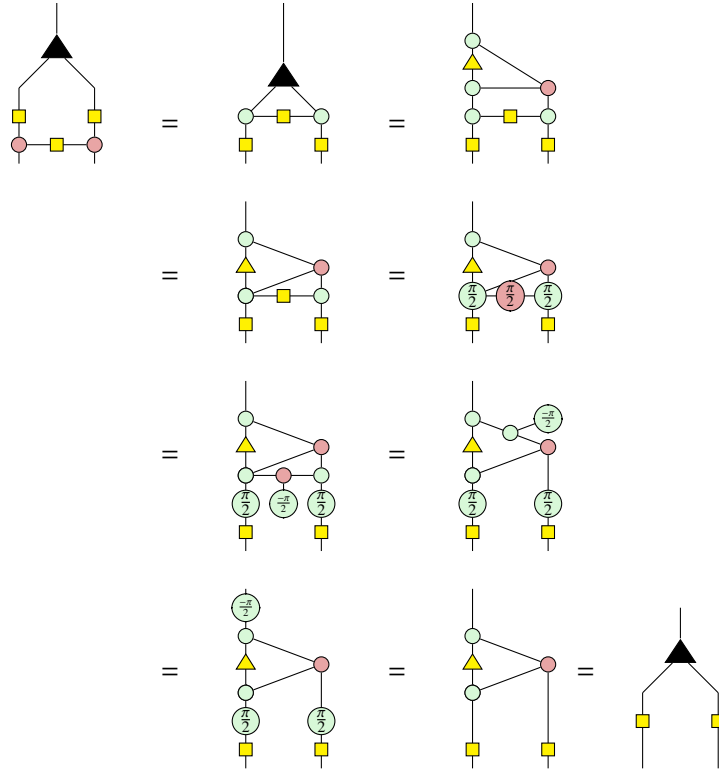


where there is a leg on the j -th qubit if $P_{ij} \neq I$. □

Proof of Proposition 6.4.



Here for the second equality we have

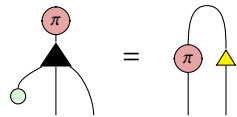


□

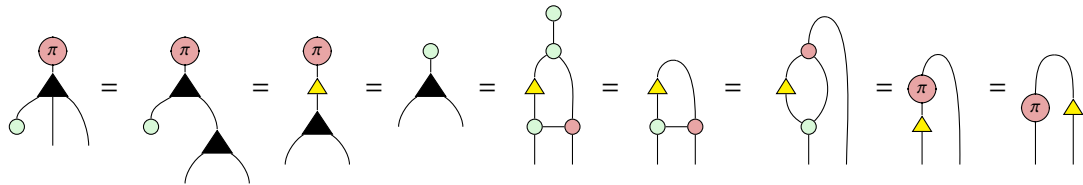
B Circuit extraction of the exponential from Equation 13

To simplify this diagram to a circuit, we will use the following two propositions.

Proposition B.1.

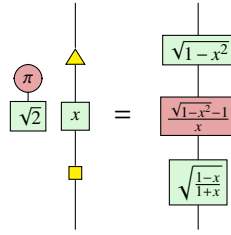


Proof of Proposition B.1.

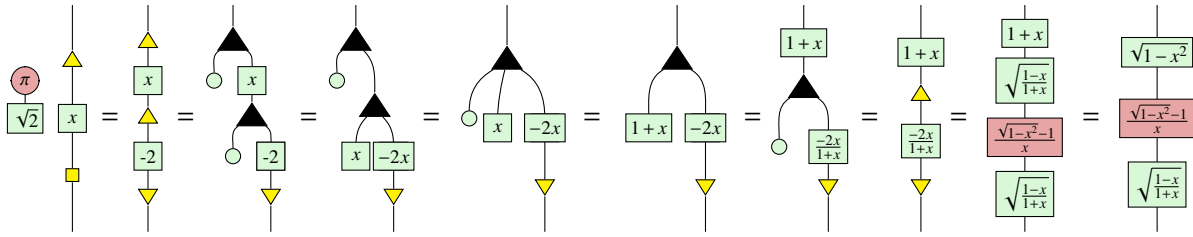


□

Proposition B.2.

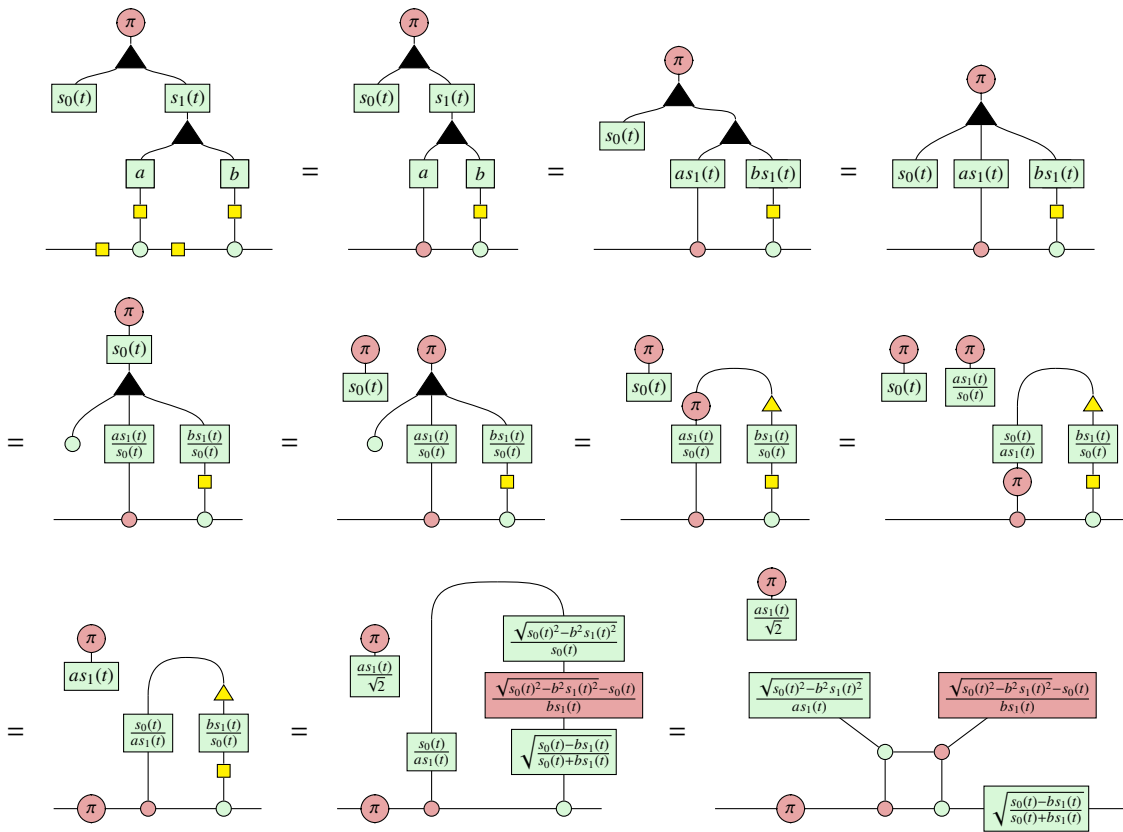


Proof of Proposition B.2.



□

Now, we begin simplifying (13).



$$\begin{aligned}
 &= \begin{array}{c} \pi \\ \boxed{\frac{as_1(t)}{\sqrt{2}}} \\ \hline \pi \end{array} \begin{array}{c} \boxed{\frac{\sqrt{s_0(t)^2 - b^2 s_1(t)^2 - s_0(t)}}{bs_1(t)}} \\ \hline \circ \end{array} \begin{array}{c} \boxed{\frac{\sqrt{s_0(t)^2 - b^2 s_1(t)^2}}{as_1(t)}} \\ \hline \bullet \end{array} \begin{array}{c} \boxed{\frac{\sqrt{s_0(t) - bs_1(t)}}{\sqrt{s_0(t) + bs_1(t)}}} \\ \hline \end{array} \\
 &= \begin{array}{c} \pi \\ \boxed{\frac{as_1(t)}{\sqrt{2}}} \\ \hline \pi \end{array} \begin{array}{c} \boxed{\frac{bs_1(t) + s_0(t) - \sqrt{s_0(t)^2 - b^2 s_1(t)^2}}{bs_1(t) - s_0(t) + \sqrt{s_0(t)^2 - b^2 s_1(t)^2}}} \\ \hline \circ \end{array} \begin{array}{c} \boxed{\frac{as_1(t) - \sqrt{s_0(t)^2 - b^2 s_1(t)^2}}{as_1(t) + \sqrt{s_0(t)^2 - b^2 s_1(t)^2}}} \\ \hline \bullet \end{array} \begin{array}{c} \boxed{\frac{\sqrt{s_0(t) - bs_1(t)}}{\sqrt{s_0(t) + bs_1(t)}}} \\ \hline \end{array}
 \end{aligned}$$

Diagrammatic Analysis for Parameterized Quantum Circuits

Tobias Stollenwerk

Institute for Quantum Computing Analytics (PGI-12), Jülich Research Centre, Wilhelm-Johnen-Straße, 52428 Jülich, Germany
German Aerospace Center (DLR), Linder Höhe, 51147 Cologne, Germany
`to.stollenwerk@fz-juelich.de`

Stuart Hadfield

Quantum Artificial Intelligence Lab (QuAIL), NASA Ames Research Center, Moffett Field, CA 94035, USA
USRA Research Institute for Advanced Computer Science (RIACS), Mountain View, CA 94043, USA
`stuart.hadfield@nasa.gov`

Diagrammatic representations of quantum algorithms and circuits offer novel approaches to their design and analysis. In this work, we describe extensions of the ZX-calculus especially suitable for parameterized quantum circuits, in particular for computing observable expectation values as functions of or for fixed parameters, which are important algorithmic quantities in a variety of applications ranging from combinatorial optimization to quantum chemistry. We provide several new ZX-diagram rewrite rules and generalizations for this setting. In particular, we give formal rules for dealing with linear combinations of ZX-diagrams, where the relative complex-valued scale factors of each diagram must be kept track of, in contrast to most previously studied single-diagram realizations where these coefficients can be effectively ignored. This allows us to directly import a number useful relations from the operator analysis to ZX-calculus setting, including causal cone and quantum gate commutation rules. We demonstrate that the diagrammatic approach offers useful insights into algorithm structure and performance by considering several ansätze from the literature including realizations of hardware-efficient ansätze and QAOA. We find that by employing a diagrammatic representation, calculations across different ansätze can become more intuitive and potentially easier to approach systematically than by alternative means. Finally, we outline how diagrammatic approaches may aid in the design and study of new and more effective quantum circuit ansätze.

1 Introduction

Diagrammatic approaches to quantum mechanics [9, 13, 12] have gained much attention in recent years as an advantageous alternative approach to analyzing and understanding quantum systems, providing simpler intuition and in some cases improved algorithmic approaches. These methods provide straightforward rules for representing, manipulating, and simplifying quantum objects, while at the same time are underpinned by sophisticated mathematical ideas (in particular, category theory [1, 55]). An important example is the ZX-calculus [9, 10, 55] and its closely related variants [43, 51, 31, 35, 32, 3, 20] which have seen a number of successful applications in quantum computing, ranging from circuit optimization [19, 37, 4, 25] and synthesis [15, 26], to algorithm analysis [7, 49], natural language processing [11] and machine learning [57, 48, 58], among others.

In this paper we show how the ZX-calculus is also useful for analyzing algorithms based on parameterized quantum circuits (PQCs), such as variational quantum algorithms, in particular for calculating important derived quantities such as expectation values of quantum observables, or their gradients. Such quantities may be computed as functions of the circuit parameters, in which case the parameters are symbolically carried through subsequent ZX-diagrams, or as numbers for the case of fixed parameters

of interest. To enable this, we present several new ZX-rules generalizing the standard ones appearing in the literature; in particular, we present rules and notation for explicitly handling linear combinations of ZX-diagrams which naturally arise, for example, when incorporating commutation rules for unitary operators which are used for instance in computing expectation values. For linear combination of diagrams, clearly, it is critical to keep track of the scalar multiplier of each diagram, whereas in previous single-diagram applications such global phases or normalization constants can typically be ignored. In our application these multipliers will typically be complex-valued functions of the quantum circuit parameters. Furthermore, our formalism then allows direct importation of a number of useful relations from the operator analysis to ZX calculus setting, such as causal cone and operator commutation rules, among others.

After stating the new rules we demonstrate their efficacy with several prototypical examples of parameterized quantum circuits in the context of combinatorial optimization, including straightforward derivation of some new and existing results concerning example circuits drawn from the literature. While for computing expectation values of relatively shallow circuits we are able to show most of the key diagram reduction steps explicitly, for deeper circuits our approach can be aided by integration with software implementations of the ZX-calculus (e.g., [38, 36]). Though we focus on the common task of analyzing quantum circuit expectation values, important in particular for assessing algorithm performance, our proposed rules are general and may find much broader application in future work. For instance, toward analyzing phenomena related to parameter setting, expectation value gradients may be obtained either by differentiating directly [48], or by reducing the calculation to that of computing further circuit expectation values as in parameter shift rules [17, 56]. We emphasize that our approach may be applied to a wide variety of application problems and related quantum circuits beyond those explicitly considered in our examples, and further ZX results and generalizations from the literature may be leveraged, including extensions to qudits [51] or fermions [32, 16], among others.

2 Preliminaries

2.1 ZX-Calculus

We refer the reader to [50, 55] and the references therein for comprehensive introductions, including complete sets of graphical rewrite rules as well as their mathematical details. A number of the most important ZX-diagram rewrite rules are displayed in Figure 1. We use the label attached to each equation to reference these rules when we apply them in the examples we consider below.

2.2 Parameterized Quantum Circuits

Parameterized quantum circuits (PQC) have gained much attention in recent years, in particular as heuristic approaches suitable for NISQ [42] era devices that are classically optimized (often variationally) as part of a hybrid protocol, though we emphasize they are by no means restricted to this setting; see [8, 5] for reviews of recent developments. Two particular approaches of interest are the QAOA (quantum alternating operator ansatz [29], which generalizes the quantum approximate optimization algorithm [22]) and VQE (variational quantum eigensolver [41, 40]) paradigms, as well as a number of more recent variants of these approaches. Here we briefly review the original QAOA paradigm and its application to combinatorial optimization, though our results to follow may be applied more generally to a variety of problems and algorithms. In QAOA we are given a cost function $c(x)$ and corresponding classical Hamiltonian C (i.e., diagonal in the computational basis, $C|x\rangle = c(x)|x\rangle$) we seek to optimize over bit

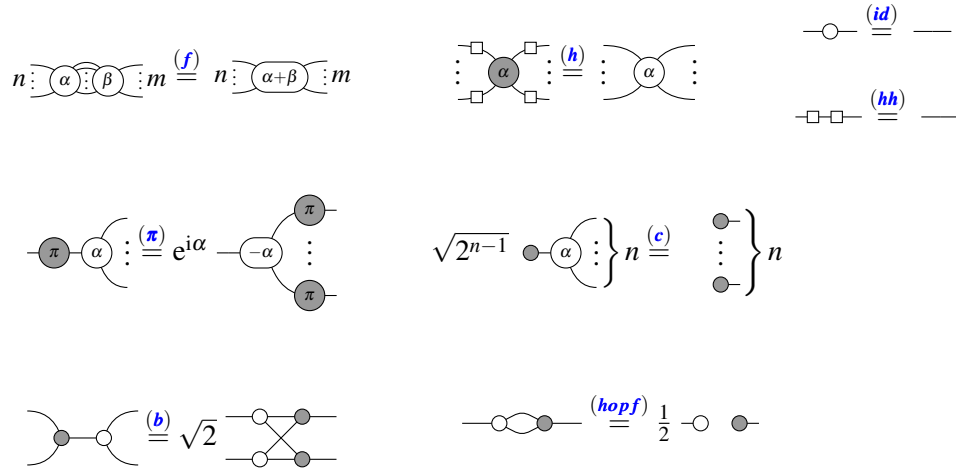


Figure 1: The ZX-diagram rewrite rules (cf. for example [55] or [58]). Note the explicit scalar factors.

strings $x \in \{0, 1\}^n$. A QAOA_p circuit consists of $2p$ alternating layers specified by $2p$ angles γ_i, β_i in some domain (e.g. $[-\pi, \pi]$) to create the state

$$|\boldsymbol{\gamma}\boldsymbol{\beta}\rangle = U_M(\beta_p)U_P(\gamma_p) \dots U_M(\beta_1)U_P(\gamma_1) |s\rangle,$$

for phase operator $U_P(\gamma) = \exp(-i\gamma C)$, (transverse-field) mixing operator $U_M(\beta) = \exp(-i\beta B)$ where $B = \sum_{i=1}^n X_i$, and standard initial product state $|s\rangle = |+\rangle^{\otimes n}$. The state is then measured in the computational basis which returns some $y \in \{0, 1\}^n$ achieving cost $c(y)$. Figure 2 shows a simple example of a QAOA circuit. Repeated state preparation and measurement gives further samples which may be used to

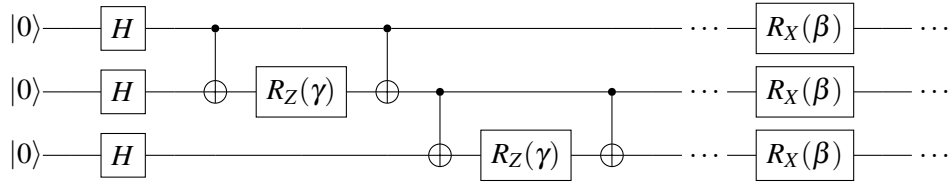


Figure 2: Example of a parameterized quantum circuit: QAOA on 3 qubits. Here the phase and mixing operators as well as initial state preparation have been compiled to basic quantum gates.

estimate the cost expectation $\langle C \rangle_p$ or other important quantities. These quantities may be used to update or search for better circuit parameters if desired; we emphasize that in different cases parameters may be found through analytic [53], numeric [22], or average-case [47] techniques, or, distinctly, searched for empirically (e.g., variationally). After a set number of runs overall, or when other suitable termination criteria has been reached, the best solution found is returned.

A fundamentally important quantity for QAOA as well as related approaches is the cost expectation value $\langle C \rangle$, which may be computed for a single instance or over a suitable class, and can be used to bound the expected approximation ratio achieved [22, 30, 29, 28] for the given problem. Importantly, we are often given a decomposition of the cost Hamiltonian such as $C = \sum_j C_j$ which we may exploit in computing $\langle C \rangle = \sum_j \langle C_j \rangle$ as a sum of terms (typically, a linear combination of Pauli Z operators [27]),

which directly motivates the rules we introduce for accommodating linear combinations of ZX-diagrams. For combinatorial optimization the C_j terms mutually commute which leads to further simplifications, whereas this may not be true for more general problems and applications such as quantum chemistry (though linearity of expectation still applies). In general, many quantities of interest for PQC's can be expressed as expectation values and are hence amenable to similar analysis via diagrammatic techniques as we explore below.

2.3 Related Work

Several recent papers provide related but distinct results towards applying the ZX calculus in the PQC setting. In particular, three papers [48, 52, 34] which appeared during preparation of this work that consider differentiation and addition of ZX-diagrams. These papers introduce diagrammatic extensions complementary to our results. However they do not deal with expectation values explicitly which is the focus of this work. In terms of previous applications to variational quantum algorithms, a recent paper [58] considers using the ZX-calculus for computing and analyzing expectation values of derivatives of the cost expectation for particular classes of random parameterized quantum circuits built from particular gate sets (see in particular [58, Assumption 1]), in the context of detecting possible barren plateaus [39]. Our work differs in that we consider expectation values of the cost function themselves, and make no similar assumption of randomly selected parameters. A particular similarity with [58] is both their application and ours require explicit accounting of scalar factors associated to ZX-diagrams (see Section 3). However, while it is observed in [58, Eq. 7] that quantum expectation values may be represented with the ZX-calculus in [58, Eq. 7], the authors do not apply the decomposition $C = \sum_j C_j$, which we exploit to derive novel ZX rules and analysis. Our approach and results are complementary to those of [58]. Another work [23] applied ZX-calculus in analysis of symmetries in the parameter landscape of the cost function expectation. We note that a different diagrammatic approach to constructing parameterized quantum circuits is considered in [33]. Concepts related to linear combinations of ZX-diagrams have been discussed in the framework of category theory for example in [14, 18].

3 ZX-Calculus for Parameterized Quantum Circuits

In this section we extend the ZX-calculus to accommodate linear combinations of (conventional) ZX-diagrams. Then, toward its application to parameterized quantum circuits we derive a collection of general rules and useful identities within the new framework. We will apply these rules to several concrete quantum circuit examples in Section 4 and the Appendices.

3.1 Diagrammatic Rules for Linear Combinations

Here we define linear combinations of diagrams, in which case diagram constants give the relative weights of the sum. For example, for computing the expectation value of an observable $H = \sum_{j=1}^m a_j H_j$ for some quantum circuit state $|\psi\rangle = U|\psi_0\rangle$ we have $\langle H \rangle_\psi = \sum_{j=1}^m a_j \langle H_j \rangle_\psi$, which hence corresponds to a single ZX-diagram or equivalently to a sum of m weighted diagrams. This idea generalizes in the natural way to sums of linear maps and more general ZX objects. We also show new ZX-diagram rules which relate single (sub)diagrams to sums or products of (sub)diagrams, such that the resulting diagram reductions involve differing numbers of diagrams.

As mentioned, we do not use the common convention of considering diagrams equivalent up to scalars or phases; hence we include complex scalar multipliers explicitly in our diagrams and rules to

follow, i.e.,

$$a \cdot m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n \neq b \cdot m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n \quad \text{unless } a = b,$$

where a, b are complex scalar multipliers and the diagram is a placeholder for an arbitrary ZX-diagram with m inputs and n outputs. In particular, care must be taken in applying the usual rules of ZX-calculus to account for any implicit constant factors. We note that scalar factors are also retained in the distinct application of [58]; see [58, Fig. 4] for an example list of some ZX-diagram rewrite rules with explicit scalars.

Definition 3.1 (Sum notation). We define novel diagram notation for describing arbitrary linear combinations. The linear combination of two ZX-diagrams with m inputs and n outputs is written

$$a \cdot m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n + b \cdot m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline B \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n =: m \begin{array}{|c|} \hline \Sigma \\ \hline \end{array} \begin{array}{|c|} \hline a \\ \hline \end{array} \begin{array}{|c|} \hline m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n \\ \hline \end{array} \begin{array}{|c|} \hline b \\ \hline \end{array} \begin{array}{|c|} \hline m \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline B \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n \\ \hline \end{array} \begin{array}{|c|} \hline \Xi \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} n \quad . \quad (1)$$

Each summand is written inside a *bubble* and the scalar factors are written on the line combining the new summation symbol and the bubbles. This definition naturally extends to an arbitrary number of summands. Summand diagrams are required have the same numbers of input (m) and output (n) lines as each other and as the those of the sum object. Note that m or n are zero for diagrams representing states ($m = 0$), effects ($n = 0$), or constants ($m = n = 0$). Sums of diagrams also arise in [57, 48, 58] in the context of differentiating diagram components (where sums arise, for example, from the product rule of calculus). Our work is complementary to these results in that we consider the generalization to complex linear combinations of diagrams; extensions to scalars beyond the complex numbers are also possible [48].

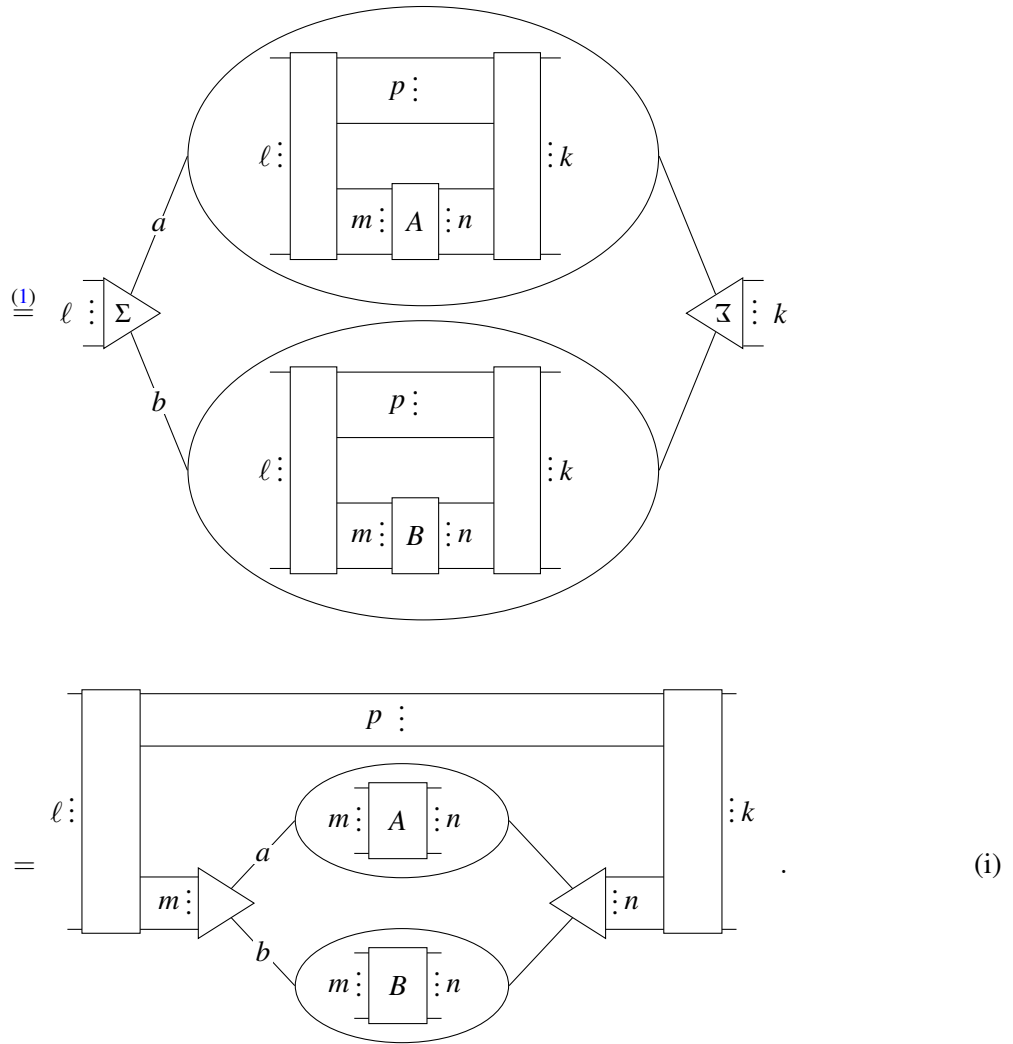
3.1.1 Rules

Now we state the two rules needed for the extension of ZX-calculus to linear combinations.

1. Diagram Pull Rule

The first rule applies if diagrams in a linear combination are equal up to a certain subdiagram (A and B below). Then we can write a single diagram containing the linear combination of the beforementioned subdiagrams

$$a \cdot \ell \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline p \vdots \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} k + b \cdot \ell \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline p \vdots \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} k$$

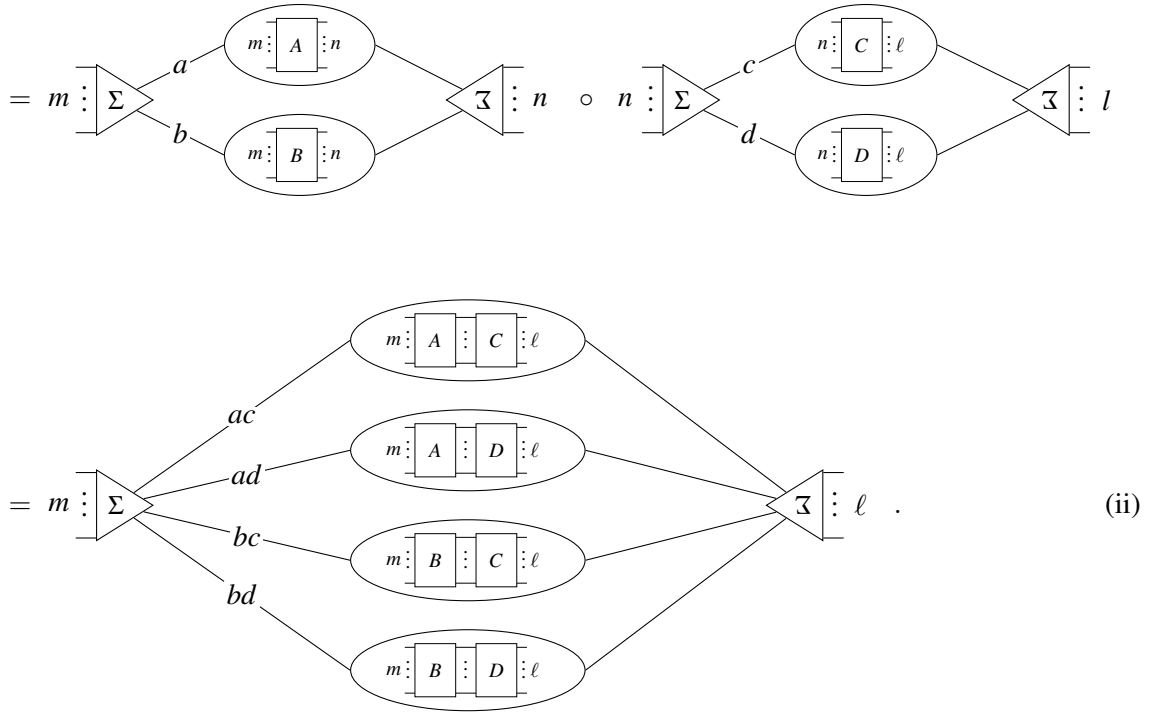


The last equality we call the *diagram pull rule*. This also holds if any of the ℓ, p, k, m, n vanish. Thus describing how to pull scalars, effects and states in and out of the bubbles. If $p = 0$, the rule describes how to pull in and out diagrams only from the left or only from the right. We will make heavy use of this in Section 4.

2. Product (Composition) Rule

The second rule describes how to combine products (i.e., compositions) of linear combinations of diagrams. We state the rule for a product of two linear combinations comprised of two summands each

$$\left(a \cdot \begin{array}{c} \text{---} \\ m \text{ : } \\ \boxed{A} \\ \text{ : } n \\ \text{---} \end{array} + b \cdot \begin{array}{c} \text{---} \\ m \text{ : } \\ \boxed{B} \\ \text{ : } n \\ \text{---} \end{array} \right) \circ \left(c \cdot \begin{array}{c} \text{---} \\ n \text{ : } \\ \boxed{C} \\ \text{ : } \ell \\ \text{---} \end{array} + d \cdot \begin{array}{c} \text{---} \\ n \text{ : } \\ \boxed{D} \\ \text{ : } \ell \\ \text{---} \end{array} \right)$$



The product rule extends in the obvious way to the case of more than two factors or summands. Several additional rules are given in Appendix A.

3.2 ZX-Calculus for Expectation Values of Quantum Circuits

In this section, we will present various identities within the extended ZX-calculus framework, that are useful for the analysis of parameterized quantum circuits. While we primarily consider Pauli operators here, similar results may be derived in different basis or gate sets. See [16] for some additional useful rules regarding Pauli operator exponentials.

3.2.1 Rotations

First, we can write rotation operators in terms of linear combinations of Clifford gates

$$e^{i\gamma Z} = e^{i\gamma} \text{---}(-2\gamma\text{---}) = \text{---} \Sigma \begin{matrix} c\gamma \\ i s\gamma \end{matrix} \begin{matrix} \text{---} \\ \text{---}(\pi\text{---}) \end{matrix} \text{---} Z \text{---} , \tag{5}$$

$$e^{i\beta X} = e^{i\beta} \text{---}(-2\beta\text{---}) = \text{---} \Sigma \begin{matrix} c\beta \\ i s\beta \end{matrix} \begin{matrix} \text{---} \\ \text{---}(\pi\text{---}) \end{matrix} \text{---} Z \text{---} . \tag{6}$$

In both cases the proof easily follows from the identity $e^{i\alpha A} = \cos \alpha I + i \sin \alpha A$ for operators satisfying $A^2 = I$. We use $c_\alpha := \cos(\alpha)$ and $s_\alpha := \sin(\alpha)$ throughout.

3.2.2 Phase-Gadgets

Important for parameterized quantum circuits are multi-qubit rotations, so-called *phase-gadgets* (cf. [16]), for example

$$e^{i\gamma Z_u Z_v} = \sqrt{2}e^{i\gamma} \begin{array}{c} \text{---} \\ \bullet \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \bullet \\ \text{---} \end{array} (-2\gamma) = \begin{array}{c} u \\ v \end{array} \Sigma \begin{array}{c} c\gamma \\ i s\gamma \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \pi \\ \pi \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \pi \\ \pi \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (7)$$

A proof of the first equality is given in [16, Corollary 3.4]. The second equality is derived similarly to (5), (6). In particular, for the analysis of QAOA expectation values, we will encounter conjugates of phase-gadgets in conjunction with π -X-spiders. We will make heavy use of the following identity which is proven in Appendix D.1.

$$\begin{array}{c} \text{---} \\ \text{---} \\ \gamma \\ \ell\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ b\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \gamma \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \end{array} = \begin{cases} \frac{e^{i\gamma}}{\sqrt{2}} & \begin{array}{c} \text{---} \\ \text{---} \\ (t+r)\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ -2\gamma \\ \text{---} \\ \text{---} \\ \text{---} \\ r\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ (b+1)\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \end{array} & \text{if } (t+l+b+r) \text{ odd} \\ \frac{1}{2} & \begin{array}{c} \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \\ \text{---} \\ \text{---} \\ \text{---} \\ b\pi \\ \text{---} \\ \text{---} \\ \text{---} \\ \pi \end{array} & \text{if } (t+l+b+r) \text{ even} \end{cases} \quad (8)$$

Phase-gadgets can be combined to implement so-called *phase polynomials*, i.e., parameterized exponentials of diagonal Hamiltonians such as utilized in QAOA circuits [26, 16, 29, 27].

3.2.3 Lightcones

For quantum circuits of limited depth or connectivity, it is often the case when computing a particular quantity that a significant fraction of the gates and qubits can be ignored or discarded due to having no effect, in analogy with spacelike-separated events in relativity. Naturally, the same principle may be fruitfully applied to diagrammatic analysis.

Given an observable $C = \sum_j C_j$, typically each C_j acts nontrivially on a subset of $\ell < n$ qubits. Hence, depending on the structure of the problem and given quantum circuit ansatz $U |\psi_0\rangle$, the n -qubit expectation values $\langle C_j \rangle$ may be equivalently reduced to ones over L qubits, $\ell \leq L \leq n$, by in each case restricting the quantum circuit in the natural way. This phenomena is generally known as the *lightcone* or *causal cone* rule [21, 22, 47, 28], and is clearly exhibited with the ZX-calculus. For example, if $|\psi_0\rangle$ is a product state and U consists of only 1-local gates, then $L = \ell$ independently of the circuit depth (cf. the example of Section 4.1). For QAOA applied to MaxCut, $\ell = 2$ and it is easily shown that the lightcone after each q th QAOA layer consists of the restriction to the subgraph within distance q of the given edge [22, 28], i.e., its size L depends on the vertex degrees in the graph neighborhood. Hence, importantly, for QAOA or similar layered ansatz we may apply the lightcone rule layer-by-layer. Applying this restriction, the

inner operator for a MaxCut QAOA expectation value reads

$$\mathcal{O}_{uv}^p := \prod_{\ell=1}^p e^{i\gamma_\ell C} e^{i\tilde{\beta}_\ell B} Z_u Z_v \prod_{k=p}^1 e^{-i\tilde{\beta}_k B} e^{-i\gamma_k C}$$

$$\mathcal{M}_{uv}^p \quad \vdots$$

$$= \mathcal{N}_{uv}^{p-1} \quad \vdots$$

$$\mathcal{N}_{uv}^1 \quad \vdots$$

$$, \quad (9)$$

where we used placeholder diagrams for the *reduced* phase-separation layer

$$, \quad (10)$$

and the *reduced* mixing layer

$$, \quad (11)$$

where $\beta := -2\tilde{\beta}$ for convenience. For the reduced phase-separation layer we have used the MaxCut cost function Hamiltonian $C = \frac{1}{2} \sum_{(u,v) \in E} (1 - Z_u Z_v)$ and

$$e^{i\gamma C} = \prod_{(u,v) \in E} e^{\frac{i\gamma}{2}} e^{\frac{-i\gamma}{2} Z_u Z_v} \stackrel{(7)}{=} \sqrt{2}^{|E|} \prod_{(u,v) \in E} \begin{array}{c} \text{---} u \\ \text{---} \circ \text{---} \gamma \text{---} \circ \text{---} \\ \text{---} v \end{array}.$$

This leads to the factor $\sqrt{2}^{n_L}$ in (10), where n_L is the number of phase-gadgets in the right diagram of (10). Also, we implicitly used the neighborhood of a set of nodes L , $N_L := \bigcup_{\ell \in L} \text{nbhd}(\ell)$, the exclusive p -th neighborhood of $\{u, v\}$, recursively defined by

$$\mathcal{N}_{uv}^p := \bigcup_{i,j \in \mathcal{N}_{uv}^{p-1} \times \mathcal{N}_{uv}^{p-1} \cap E} N_{\{i,j\}} \setminus \bigcup_{k=0}^{p-1} \mathcal{N}_{uv}^k,$$

where $\mathcal{N}_{uv}^0 := \{u, v\}$, as well as the complement $\mathcal{M}_{uv}^p = \mathcal{N}_{uv}^p \setminus E$.

While here we have considered QAOA circuits as a demonstrative example, the same principle may be applied to or formalized for more general ansätze and observables.

4 Application to Combinatorial Optimization

Expectation values of quantum circuit observables – i.e., constants – may be represented with ZX-diagrams, as has been previously observed in [58, Eq. 7]. In doing so, in some cases the structure of the original problem may be directly reflected in the structure of the corresponding ZX-diagrams. This is demonstrated by two examples in this section, in which apply our ZX-calculus extension to calculate cost expectation values for a particular ansatz for combinatorial optimization. The purpose of this section is twofold. First, we want to demonstrate that calculations with parameterized quantum circuits, like the finding an analytical expression for expectation values, can sometimes become more intuitive and simplified by using ZX-calculus in conjunction with our extension to linear combinations. Second, we show that our extension is indeed necessary to achieve the aforementioned task diagrammatically by, for instance, providing means to “commute” X- and Z-spiders (cf. (12)), while explicitly keeping track of all resulting terms.

We show how the cost function expectation value $\langle C \rangle$ may be computed and analyzed using our extended ZX-calculus. Recall that given a decomposition of the cost Hamiltonian $C = \sum C_\ell$ it suffices to compute the $\langle C_\ell \rangle$ values independently, which typically correspond to similar diagrams. In particular (sub)graph symmetry can be exploited to reduced the number of unique diagrams required [22, 44, 45]. Generally the quantity $\langle C \rangle$ is important in parameter setting, as well as bounding algorithm performance such as the approximation ratio achieved [30].

4.1 Independent Single-Qubit Rotations Ansatz

We begin with a simple but important example. Consider an arbitrary cost function and corresponding cost (diagonal) Hamiltonian C on n qubits we seek to extremize, together with the simple depth-1 ansatz consisting of a free single-qubit Pauli- Y rotation on each qubit, applied to the initial state $|00\dots 0\rangle = |0\rangle^{\otimes n}$,

$$\begin{array}{c} |0\rangle \text{ --- } \boxed{R_Y(\alpha_1)} \text{ ---} \\ |0\rangle \text{ --- } \boxed{R_Y(\alpha_2)} \text{ ---} \\ \vdots \\ |0\rangle \text{ --- } \boxed{R_Y(\alpha_n)} \text{ ---} \end{array} = \frac{1}{\sqrt{2^n}} \begin{array}{c} \bullet \text{---} \left(-\frac{\pi}{2}\right) \text{---} \alpha \text{---} \left(\frac{\pi}{2}\right) \text{---} \\ \bullet \text{---} \left(-\frac{\pi}{2}\right) \text{---} \alpha \text{---} \left(\frac{\pi}{2}\right) \text{---} \\ \vdots \\ \bullet \text{---} \left(-\frac{\pi}{2}\right) \text{---} \alpha \text{---} \left(\frac{\pi}{2}\right) \text{---} \end{array}.$$

For example, consider an arbitrary instance of MaxCut, a prototypical NP-hard optimization problem, though the same argument we show here applies similarly to many other problems. For a graph with edge

set E the cost Hamiltonian is $C = \frac{|E|}{2} - \frac{1}{2} \sum_{(uv) \in E} Z_u Z_v$. As demonstrated in Equation (12) the derivation of each $\langle Z_u Z_v \rangle$ becomes very simple with ZX-calculus.

$$\begin{aligned}
 \langle Z_u Z_v \rangle &= \frac{1}{2^n} \text{Tr} \left(\underbrace{\begin{array}{c} \circ \text{---} \circ \\ \vdots \\ \circ \text{---} \circ \\ \circ \text{---} \left(-\frac{\pi}{2} \right) \text{---} \alpha_v \text{---} \pi \text{---} -\alpha_v \text{---} \left(\frac{\pi}{2} \right) \text{---} \circ \\ \circ \text{---} \left(-\frac{\pi}{2} \right) \text{---} \alpha_u \text{---} \pi \text{---} -\alpha_u \text{---} \left(\frac{\pi}{2} \right) \text{---} \circ \end{array}}_{(f) (\pi)} \right) = c_{\alpha_u} c_{\alpha_v} \\
 &= e^{i\alpha_u} \text{Tr} \left(\begin{array}{c} \circ \text{---} \left(\frac{\pi}{2} \right) \text{---} -2\alpha_u \text{---} \left(\frac{\pi}{2} \right) \text{---} \circ \end{array} \right) \\
 &\stackrel{(6)}{=} \text{Tr} \left(\begin{array}{c} \circ \text{---} \left(\frac{\pi}{2} \right) \text{---} \Sigma \begin{array}{l} c_{\alpha_u} \text{---} \text{---} \\ i s_{\alpha_u} \text{---} \pi \text{---} \end{array} \text{---} \Xi \text{---} \left(\frac{\pi}{2} \right) \text{---} \circ \end{array} \right) \\
 &= \text{Tr} \left(\begin{array}{c} \Sigma \begin{array}{l} c_{\alpha_u} \text{---} \text{---} \\ i s_{\alpha_u} \text{---} \text{---} \end{array} \text{---} \Xi \text{---} \underbrace{\begin{array}{c} \text{---} \text{---} \\ \left(\frac{\pi}{2} \right) \text{---} \pi \text{---} \left(\frac{\pi}{2} \right) \end{array}}_{=0} \end{array} \right) = 2c_{\alpha_u} \quad . \quad (12)
 \end{aligned}$$

Here, again, $s_\alpha = \sin(\alpha)$ and $c_\alpha = \cos(\alpha)$, and each underbrace used refers only to the subdiagram directly above. Note that after the second step the usage of linear combinations to handle the X-spider with phase $(-2\alpha_u)$ provides a way to continue the calculation, which would not be possible within the conventional ZX-framework. From the permutation symmetry of the ansatz, the expectation value $\langle Z_i Z_j \rangle$ of each edge is of the same form [44]. Hence we have

$$\langle C \rangle = \frac{|E|}{2} - \frac{1}{2} \sum_{(u,v) \in E} \cos(\alpha_u) \cos(\alpha_v), \tag{13}$$

which implies

$$\max_{\alpha} \langle C \rangle = \max_{\alpha \in \{0, \pi\}^n} \langle C \rangle = \max_x c(x) = c(y^*),$$

where we have used the observation that angles $\alpha^* \in \{0, \pi\}^n$ encode a bit string y^* via $y_i^* = \frac{1}{2} - \frac{1}{2} \cos(\alpha_i^*)$. Hence, as any globally optimal angles must directly encode an optimal solution to the MaxCut instance, the expectation value $\langle C \rangle$ is NP-hard to optimize. Indeed, for MaxCut, (13) reproduces the quantity of Equation 1 of [6] (up to an affine shift). This result is used throughout [6] via further reductions to show that optimizing a number of other classes of PQC's is NP-hard in general. We have similarly demonstrated that the single-qubit rotations ansatz is NP-hard to optimize for problems such as MaxCut, but via a compact derivation using ZX-diagrams.

4.2 QAOA₁ for MaxCut on a Simple Graph

Next we turn to QAOA [22, 29], for which we continue our use of MaxCut as a running example. For simplicity we consider QAOA₁, the lowest depth realization, which is indicative of the $p > 1$ case due

to the alternating structure of the ansatz. Recall that for a QAOA state the MaxCut expectation value reads $\langle C \rangle = \frac{|E|}{2} - \frac{1}{2} \sum_{i,j \in E} \langle Z_i Z_j \rangle$. We begin with the specific graph G of Figure 3, before we consider ring graphs in Section B.2, and arbitrary graphs in Appendix B.3. Observe how the structure of the graph

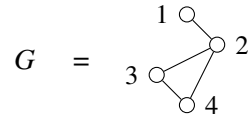
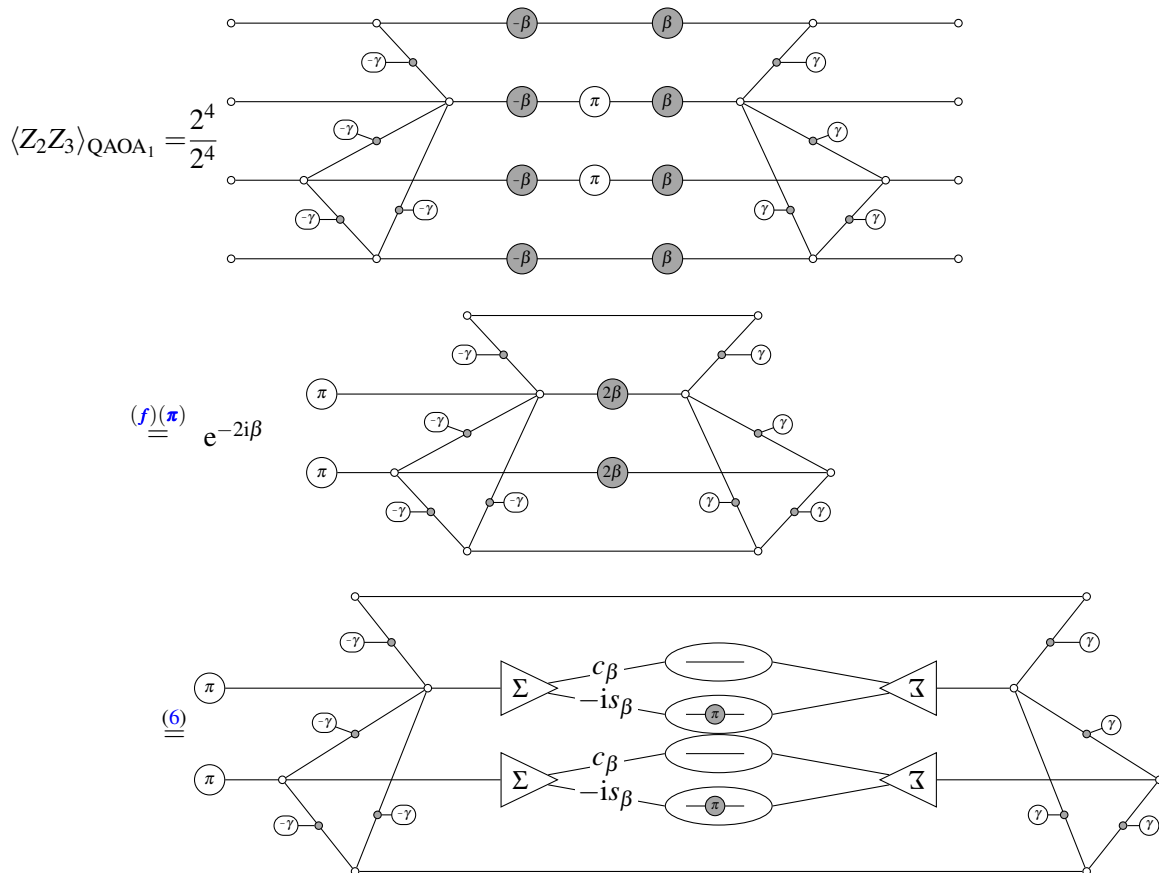
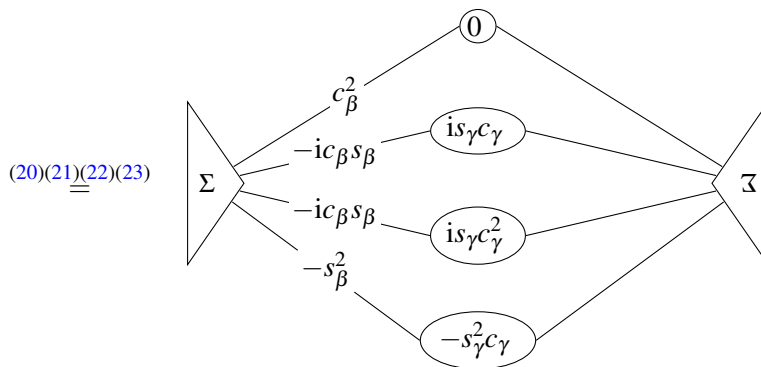
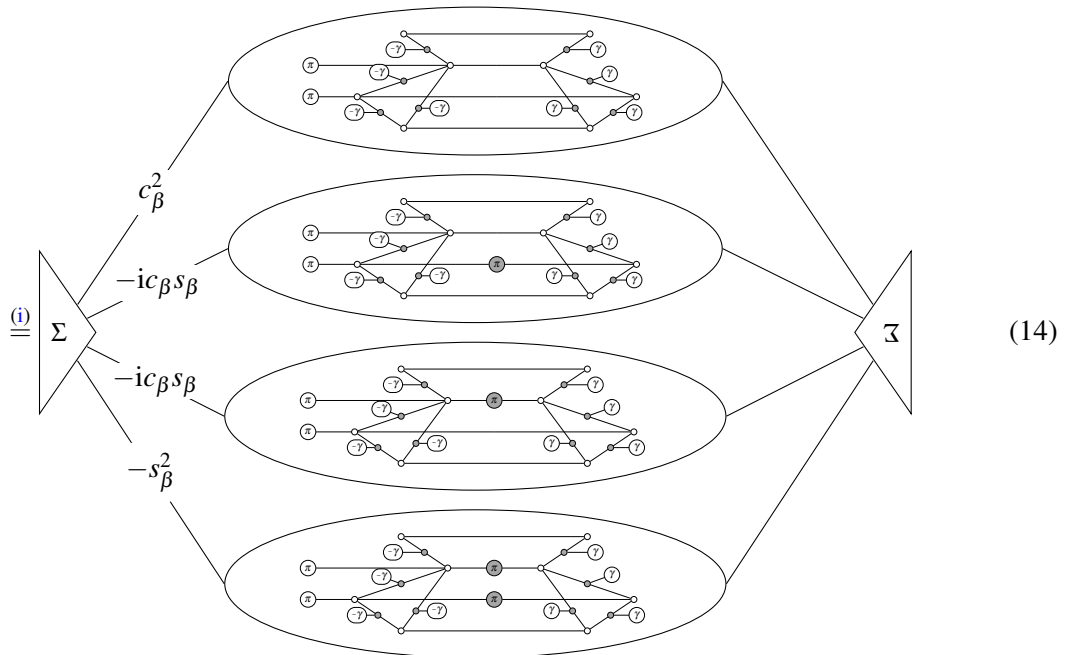
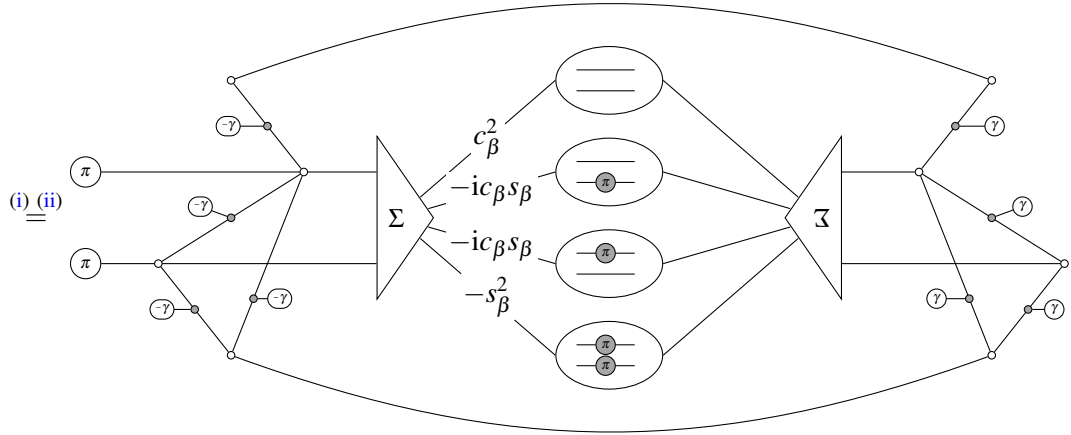


Figure 3: Simple example graph to consider for MaxCut with QAOA

directly reappears in the diagrams below, which reflects the fact that the QAOA phase operator is derived from the cost Hamiltonian. For deeper QAOA circuits, the graph structure will again appear at each layer in the diagrammatic representation. Hence ZX-calculus provides a toolkit toward directly incorporating or better understanding the relationship between the cost function and a given parameterized quantum algorithm.

Here we demonstrate the edge expectation value calculation for QAOA_1 ,





$$= c_\beta s_\beta s_\gamma c_\gamma + c_\beta s_\beta s_\gamma c_\gamma^2 + s_\beta^2 s_\gamma^2 c_\gamma \quad .$$

The remaining expectation values can be similarly computed for each of the edges in E to give $\langle C \rangle$. In the third step above, we could not have easily continued within the conventional ZX-calculus framework. Whenever one needs to pull parameterized X-spiders through parameterized Z-spiders or vice versa, our extension is utilized. The detailed calculation of the four contributions used in the last step is given in Appendix C. Note that calculation of the general n -qubit case (cf. Appendix B.3) is surprisingly concise compared to the special case of 4-qubits considered here.

We consider a hardware-efficient ansatz and two more general QAOA examples in Appendix B.

5 Outlook

We introduced an extension of the ZX-calculus to conveniently incorporate linear combinations of ZX-diagrams. Moreover we demonstrated how this generalized diagrammatic framework can be applied to the analysis of parameterized quantum circuits, in particular to the calculation of observable expectation values. Further quantities of interest such as gradients may be similarly derived, as well as more complicated PQC phenomenon such as barren plateaus studied, by combining our framework with several distinct but complementary recent ZX-calculus advances [58, 34, 52]. Software implementation of these results may facilitate novel approaches for automatic contraction of diagrams related to PQCs, including but not limited to expectation values. A concrete next step is to rigorously derive such algorithms and carefully analyze problems and PQC classes where they may yield advantages.

Future research could further formalize our approach as well as integrate it with other variants of ZX-calculus, like ZH-calculus [3] or the ZX-framework for qudits [43, 51]. In particular the latter could facilitate novel insights into performance analysis of quantum alternating operator ansätze [29] for problems like graph-coloring [54] and beyond [46]. Similarly, our approach could be likewise applied to applications beyond combinatorial optimization, like variational quantum eigensolvers for quantum chemistry applications [16]. Generally, it is of interest to explore to what extent diagrammatic approaches may ultimately aid in the design and analysis of better performing parameterized quantum circuit ansätze, as well as help with important related challenges such as alleviating the cost of parameter setting, avoiding undesirable features such as barren plateaus, or tailoring ansatz design to a given set of hardware constraints.

Acknowledgments

SH is grateful for support from the NASA Ames Research Center, from NASA Academic Mission Services (NAMS) under Contract No. NNA16BD14C, and from the DARPA ONISQ program under inter-agency agreement IAA 8839, Annex 114.

References

- [1] Samson Abramsky & Bob Coecke (2004): *A categorical semantics of quantum protocols*. In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, IEEE, pp. 415–425, doi:10.1109/LICS.2004.1319636.
- [2] Gadi Aleksandrowicz et al. (2019): *Qiskit: An Open-source Framework for Quantum Computing*, doi:10.5281/zenodo.2562111. Available at <https://qiskit.org/documentation/stubs/qiskit.circuit.library.EfficientSU2.html>.

- [3] Miriam Backens & Aleks Kissinger (2019): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*. In Peter Selinger & Giulio Chiribella, editors: Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018, *Electronic Proceedings in Theoretical Computer Science* 287, Open Publishing Association, pp. 23–42, doi:10.4204/EPTCS.287.2. Available at <https://arxiv.org/abs/1805.02175>.
- [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 <https://quantum-journal.org/papers/q-2021-03-25-421/>.
- [5] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke et al. (2022): *Noisy intermediate-scale quantum algorithms*. *Reviews of Modern Physics* 94(1), p. 015004, doi:10.1103/RevModPhys.94.015004. Available at <https://link.aps.org/doi/10.1103/RevModPhys.94.015004>.
- [6] Lennart Bittel & Martin Kliesch (2021): *Training variational quantum algorithms is NP-hard*. *Physical Review Letters* 127(12), p. 120502, doi:10.1103/PhysRevLett.127.120502. Available at <https://link.aps.org/doi/10.1103/PhysRevLett.127.120502>.
- [7] Titouan Carette, Yohann D’Anello & Simon Perdrix (2021): *Quantum Algorithms and Oracles with the Scalable ZX-calculus*. In Chris Heunen & Miriam Backens, editors: Proceedings 18th International Conference on Quantum Physics and Logic, Gdansk, Poland, and online, 7-11 June 2021, *Electronic Proceedings in Theoretical Computer Science* 343, Open Publishing Association, pp. 193–209, doi:10.4204/EPTCS.343.10. Available at <https://arxiv.org/abs/2104.01043>.
- [8] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio et al. (2021): *Variational quantum algorithms*. *Nature Reviews Physics* 3(9), pp. 625–644, doi:10.1038/s42254-021-00348-9.
- [9] Bob Coecke & Ross Duncan (2008): *Interacting quantum observables*. In: *International Colloquium on Automata, Languages, and Programming*, Springer, pp. 298–310, doi:10.1088/1367-2630/13/4/043016. Available at <https://arxiv.org/abs/0906.4725>.
- [10] 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. Available at <https://iopscience.iop.org/article/10.1088/1367-2630/13/4/043016>.
- [11] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis & Alexis Toumi (2020): *Foundations for Near-Term Quantum Natural Language Processing*. *arXiv preprint arXiv:2012.03755*, doi:10.48550/arXiv.2012.03755. Available at <https://arxiv.org/abs/2012.03755>.
- [12] Bob Coecke, Dominic Horsman, Aleks Kissinger & Quanlong Wang (2021): *Kindergarden quantum mechanics graduates (... or how I learned to stop gluing LEGO together and love the ZX-calculus)*. *arXiv preprint arXiv:2102.10984*, doi:10.48550/arXiv.2102.10984. Available at <https://arxiv.org/abs/2102.10984>.
- [13] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.
- [14] Cole Comfort, Antonin Delpuech & Jules Hedges (2020): *Sheet diagrams for bimonoidal categories*. *arXiv preprint arXiv:2010.13361*, doi:10.48550/arXiv.2010.13361. Available at <https://arxiv.org/abs/2010.13361>.
- [15] 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. Available at <https://arxiv.org/abs/1906.01734>.

- [16] Alexander Cowtan, Will Simmons & Ross Duncan (2020): *A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz*. arXiv preprint arXiv:2007.10515, doi:10.48550/arXiv.2007.10515. Available at <https://arxiv.org/abs/2007.10515>.
- [17] Gavin E Crooks (2019): *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition*. arXiv preprint arXiv:1905.13311, doi:10.48550/arXiv.1905.13311. Available at <https://arxiv.org/abs/1905.13311>.
- [18] Ross Duncan (2009): *Generalized Proof-Nets for Compact Categories with Biproducts*, p. 70–134. Cambridge University Press, doi:10.1017/CBO9781139193313.004.
- [19] 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/>.
- [20] Richard DP East, John van de Wetering, Nicholas Chancellor & Adolfo G Grushin (2022): *AKLT-states as ZX-diagrams: Diagrammatic reasoning for quantum states*. PRX Quantum 3(1), p. 010302, doi:10.1103/PRXQuantum.3.010302. Available at <https://link.aps.org/doi/10.1103/PRXQuantum.3.010302>.
- [21] Glen Evenbly & Guifré Vidal (2009): *Algorithms for entanglement renormalization*. Physical Review B 79(14), p. 144108, doi:10.1103/PhysRevB.79.144108. Available at <https://link.aps.org/doi/10.1103/PhysRevB.79.144108>.
- [22] Edward Farhi, Jeffrey Goldstone & Sam Gutmann (2014): *A quantum approximate optimization algorithm*. arXiv preprint arXiv:1411.4028, doi:10.48550/arXiv.1411.4028. Available at <https://arxiv.org/abs/1411.4028>.
- [23] Enrico Fontana, M Cerezo, Andrew Arrasmith, Ivan Rungger & Patrick J Coles (2020): *Optimizing parametrized quantum circuits via noise-induced breaking of symmetries*. arXiv preprint arXiv:2011.08763, doi:10.48550/arXiv.2011.08763. Available at <https://arxiv.org/abs/2011.08763v2>.
- [24] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn & Paolo Stornati (2021): *Dimensional Expressivity Analysis of Parametric Quantum Circuits*. Quantum 5, p. 422, doi:10.22331/q-2021-03-29-422. Available at <https://quantum-journal.org/papers/q-2021-03-29-422/>.
- [25] Jonathan Gorard, Manojna Namuduri & Xerxes D Arsiwalla (2021): *ZX-Calculus and Extended Wolfram Model Systems II: Fast Diagrammatic Reasoning with an Application to Quantum Circuit Simplification*. arXiv preprint arXiv:2103.15820, doi:10.48550/arXiv.2103.15820. Available at <https://arxiv.org/abs/2103.15820>.
- [26] Arianne Meijer-van de Griend & Ross Duncan (2020): *Architecture-aware synthesis of phase polynomials for NISQ devices*. arXiv preprint arXiv:2004.06052, doi:10.48550/arXiv.2004.06052. Available at <https://arxiv.org/abs/2004.06052>.
- [27] Stuart Hadfield (2021): *On the representation of Boolean and real functions as Hamiltonians for quantum computing*. ACM Transactions on Quantum Computing 2(4), pp. 1–21, doi:10.1145/3478519.
- [28] Stuart Hadfield, Tad Hogg & Eleanor G Rieffel (2022): *Analytical framework for quantum alternating operator ansätze*. Quantum Science and Technology 8(1), p. 015017, doi:10.1088/2058-9565/aca3ce. Available at <https://dx.doi.org/10.1088/2058-9565/aca3ce>.
- [29] Stuart Hadfield, Zhihui Wang, Bryan O’Gorman, Eleanor G. Rieffel, Davide Venturelli & Rupak Biswas (2019): *From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz*. Algorithms 12(2), doi:10.3390/a12020034. Available at <https://www.mdpi.com/1999-4893/12/2/34>.
- [30] Stuart Andrew Hadfield (2018): *Quantum algorithms for scientific computing and approximate optimization*. Columbia University, doi:10.48550/arXiv.1805.03265. Available at <https://arxiv.org/abs/1805.03265>.
- [31] Amar Hadzihasanovic (2015): *A Diagrammatic Axiomatisation for Qubit Entanglement*. In: *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, LICS ’15, IEEE Com-

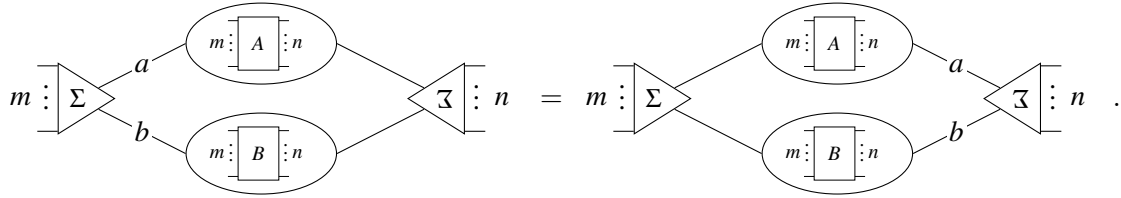
- puter Society, USA, p. 573–584, doi:10.1109/LICS.2015.59. Available at <https://ieeexplore.ieee.org/document/7174913/>.
- [32] Amar Hadzihasanovic, Giovanni de Felice & Kang Feng Ng (2018): *A Diagrammatic Axiomatisation of Fermionic Quantum Circuits*. In H el ene Kirchner, editor: *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, Leibniz International Proceedings in Informatics (LIPIcs) 108, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 17:1–17:20, doi:10.4230/LIPIcs.FSCD.2018.17. Available at <http://drops.dagstuhl.de/opus/volltexte/2018/9187>.
- [33] Y Herasymenko & TE O’Brien (2021): *A diagrammatic approach to variational quantum ansatz construction*. *Quantum* 5, p. 596, doi:10.22331/q-2021-12-02-596.
- [34] Emmanuel Jeandel, Simon Perdrix & Margarita Veshchezerova (2022): *Addition and Differentiation of ZX-diagrams*. *arXiv preprint arXiv:2202.11386*, doi:10.48550/arXiv.2202.11386. Available at <https://arxiv.org/abs/2202.11386>.
- [35] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *Y-Calculus: A Language for Real Matrices Derived from the ZX-Calculus*. In Bob Coecke & Aleks Kissinger, editors: *Proceedings 14th International Conference on Quantum Physics and Logic*, Nijmegen, The Netherlands, 3-7 July 2017, *Electronic Proceedings in Theoretical Computer Science* 266, Open Publishing Association, pp. 23–57, doi:10.4204/EPTCS.266.2. Available at <https://arxiv.org/abs/1702.00934>.
- [36] 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. Available at <https://arxiv.org/abs/1904.04735>.
- [37] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Phys. Rev. A* 102, p. 022406, doi:10.1103/PhysRevA.102.022406. Available at <https://link.aps.org/doi/10.1103/PhysRevA.102.022406>.
- [38] Aleks Kissinger & Vladimir Zamdzhiev (2015): *Quantomatic: A proof assistant for diagrammatic reasoning*. In: *International Conference on Automated Deduction*, Springer, pp. 326–336, doi:10.1007/978-3-319-21401-6. Available at <http://quantomatic.github.io/>.
- [39] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush & Hartmut Neven (2018): *B barren plateaus in quantum neural network training landscapes*. *Nature communications* 9(1), pp. 1–6, doi:10.1038/s41467-018-07090-4.
- [40] Jarrod R McClean, Jonathan Romero, Ryan Babbush & Al an Aspuru-Guzik (2016): *The theory of variational hybrid quantum-classical algorithms*. *New Journal of Physics* 18(2), p. 023023, doi:10.1088/1367-2630/18/2/023023.
- [41] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Al an Aspuru-Guzik & Jeremy L O’Brien (2014): *A variational eigenvalue solver on a photonic quantum processor*. *Nature communications* 5(1), pp. 1–7, doi:10.1038/ncomms5213.
- [42] John Preskill (2018): *Quantum computing in the NISQ era and beyond*. *Quantum* 2, p. 79, doi:10.22331/q-2018-08-06-79.
- [43] Andr e Ranchin (2014): *Depicting qudit quantum mechanics and mutually unbiased qudit theories*. In Bob Coecke, Ichiro Hasuo & Prakash Panangaden, editors: *Proceedings of the 11th workshop on Quantum Physics and Logic*, Kyoto, Japan, 4-6th June 2014, *Electronic Proceedings in Theoretical Computer Science* 172, Open Publishing Association, pp. 68–91, doi:10.4204/EPTCS.172.6. Available at <https://arxiv.org/abs/1404.1288>.
- [44] Ruslan Shaydulin, Stuart Hadfield, Tad Hogg & Ilya Safro (2021): *Classical symmetries and the Quantum Approximate Optimization Algorithm*. *Quantum Information Processing* 20(11), pp. 1–28, doi:10.1007/s11128-021-03298-4.

- [45] Ruslan Shaydulin & Stefan M Wild (2021): *Exploiting symmetry reduces the cost of training QAOA*. *IEEE Transactions on Quantum Engineering* 2, pp. 1–9, doi:[10.1109/TQE.2021.3066275](https://doi.org/10.1109/TQE.2021.3066275).
- [46] Tobias Stollenwerk, Stuart Hadfield & Zhihui Wang (2020): *Toward Quantum Gate-Model Heuristics for Real-World Planning Problems*. *IEEE Transactions on Quantum Engineering* 1, pp. 1–16, doi:[10.1109/TQE.2020.3030609](https://doi.org/10.1109/TQE.2020.3030609).
- [47] Michael Streif & Martin Leib (2020): *Training the quantum approximate optimization algorithm without access to a quantum processing unit*. *Quantum Science and Technology* 5(3), p. 034008, doi:[10.1088/2058-9565/ab8c2b](https://doi.org/10.1088/2058-9565/ab8c2b).
- [48] Alexis Toumi, Richie Yeung & Giovanni de Felice (2021): *Diagrammatic Differentiation for Quantum Machine Learning*. In Chris Heunen & Miriam Backens, editors: Proceedings 18th International Conference on Quantum Physics and Logic, Gdansk, Poland, and online, 7-11 June 2021, *Electronic Proceedings in Theoretical Computer Science* 343, Open Publishing Association, pp. 132–144, doi:[10.4204/EPTCS.343.7](https://doi.org/10.4204/EPTCS.343.7). Available at <https://arxiv.org/abs/2103.07960>.
- [49] Alex Townsend-Teague & Konstantinos Meichanetzidis (2021): *Classifying Complexity with the ZX-Calculus: Jones Polynomials and Potts Partition Functions*. *arXiv preprint arXiv:2103.06914*, doi:[10.48550/arXiv.2103.06914](https://doi.org/10.48550/arXiv.2103.06914). Available at <https://arxiv.org/abs/2103.06914>.
- [50] 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), IEEE, pp. 1–10, doi:[10.1109/LICS.2019.8785765](https://doi.org/10.1109/LICS.2019.8785765).
- [51] Quanlong Wang & Xiaoning Bian (2014): *Qutrit Dichromatic Calculus and Its Universality*. In Bob Coecke, Ichiro Hasuo & Prakash Panangaden, editors: Proceedings of the 11th workshop on Quantum Physics and Logic, Kyoto, Japan, 4-6th June 2014, *Electronic Proceedings in Theoretical Computer Science* 172, Open Publishing Association, pp. 92–101, doi:[10.4204/EPTCS.172.7](https://doi.org/10.4204/EPTCS.172.7). Available at <https://arxiv.org/abs/1406.3056>.
- [52] Quanlong Wang & Richie Yeung (2022): *Differentiating and Integrating ZX Diagrams*. *arXiv preprint arXiv:2201.13250*, doi:[10.48550/arXiv.2201.13250](https://doi.org/10.48550/arXiv.2201.13250). Available at <https://arxiv.org/abs/2201.13250>.
- [53] Zhihui Wang, Stuart Hadfield, Zhang Jiang & Eleanor G Rieffel (2018): *Quantum approximate optimization algorithm for MaxCut: A fermionic view*. *Physical Review A* 97(2), p. 022304, doi:[10.1103/PhysRevA.97.022304](https://doi.org/10.1103/PhysRevA.97.022304).
- [54] Zhihui Wang, Nicholas C. Rubin, Jason M. Dominy & Eleanor G. Rieffel (2020): *XY mixers: Analytical and numerical results for the quantum alternating operator ansatz*. *Phys. Rev. A* 101, p. 012320, doi:[10.1103/PhysRevA.101.012320](https://doi.org/10.1103/PhysRevA.101.012320). Available at <https://link.aps.org/doi/10.1103/PhysRevA.101.012320>.
- [55] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. *arXiv preprint arXiv:2012.13966*, doi:[10.48550/arXiv.2012.13966](https://doi.org/10.48550/arXiv.2012.13966). Available at <https://arxiv.org/abs/2012.13966>.
- [56] David Wierichs, Josh Izaac, Cody Wang & Cedric Yen-Yu Lin (2022): *General parameter-shift rules for quantum gradients*. *Quantum* 6, p. 677, doi:[10.22331/q-2022-03-30-677](https://doi.org/10.22331/q-2022-03-30-677). Available at <https://quantum-journal.org/papers/q-2022-03-30-677/>.
- [57] Richie Yeung (2020): *Diagrammatic Design and Study of Ansätze for Quantum Machine Learning*. *arXiv preprint arXiv:2011.11073*, doi:[10.48550/arXiv.2011.11073](https://doi.org/10.48550/arXiv.2011.11073). Available at <https://arxiv.org/abs/2011.11073>.
- [58] Chen Zhao & Xiao-Shan Gao (2021): *Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus*. *Quantum* 5, p. 466, doi:[10.22331/q-2021-06-04-466](https://doi.org/10.22331/q-2021-06-04-466). Available at <https://quantum-journal.org/papers/q-2021-06-04-466/>.

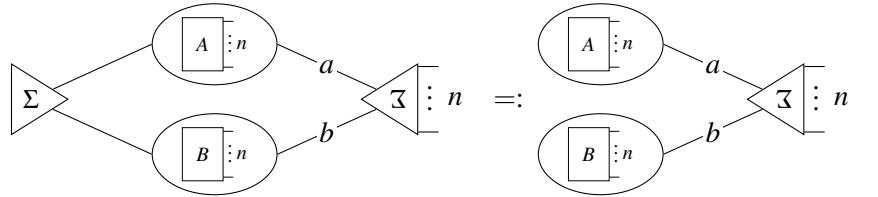
A Additional Rules for Linear Combinations of ZX-Diagrams

We introduce several additional rules which are useful for the calculation of expectation values for PQC which we utilize in the derivations to follow.

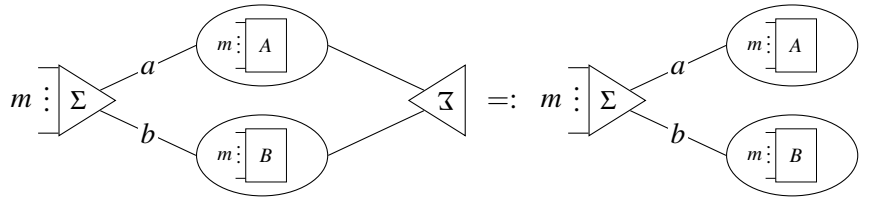
Scalar-pull rule First, scalars can be pulled through the bubble. I.e. it does not matter if we write them to the left or right of the bubbles.



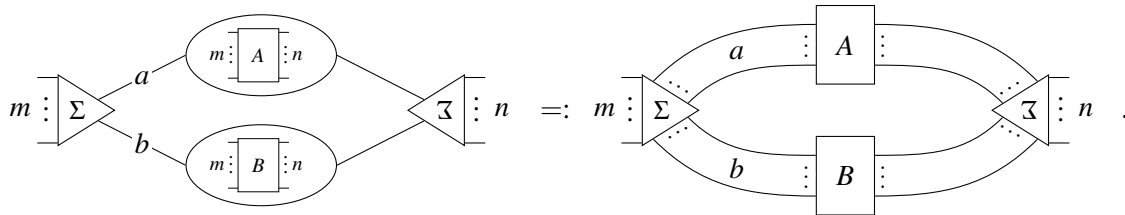
Linear combinations for states and effects Since we can put the scalar factor left or right of the bubbles, we can simplify linear combinations in the case of states or effects. For states (no inputs), we can cut the left half of the diagram



For effects (no outputs), we can cut the right half of the diagram



Direct connection of diagrams (no bubbles) We can also completely drop the bubbles and continue the input and output wires through the sum symbols



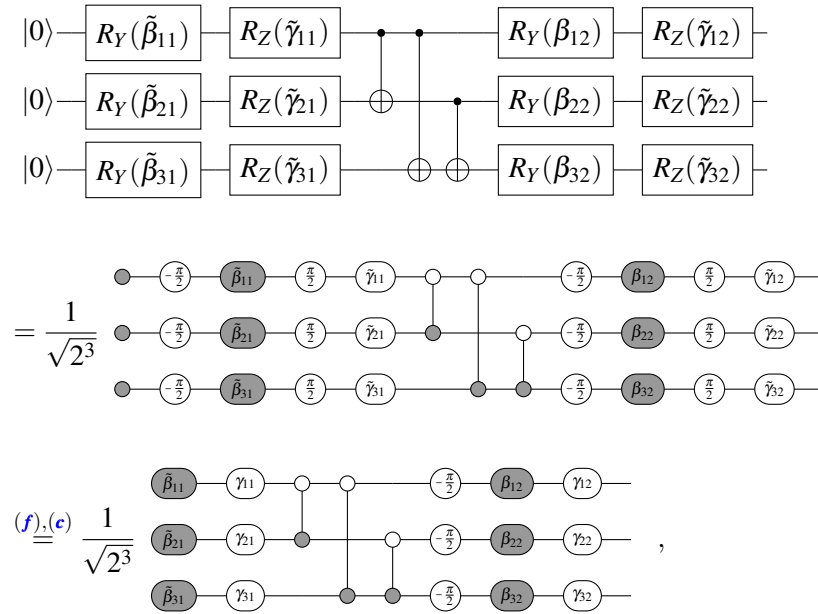
However, we will not require this notation in the examples considered in this paper.

B Additional Examples

Here we continue our examples from Section 4 and diagrammatically derive MaxCut expectation values for a hardware efficient ansatz as well as for QAOA₁ on rings and general graphs.

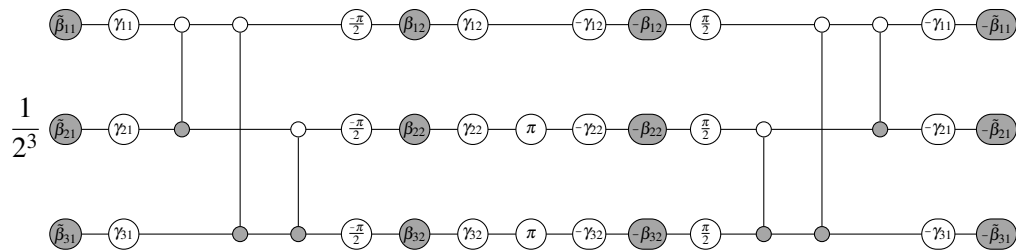
B.1 Hardware Efficient Ansatz

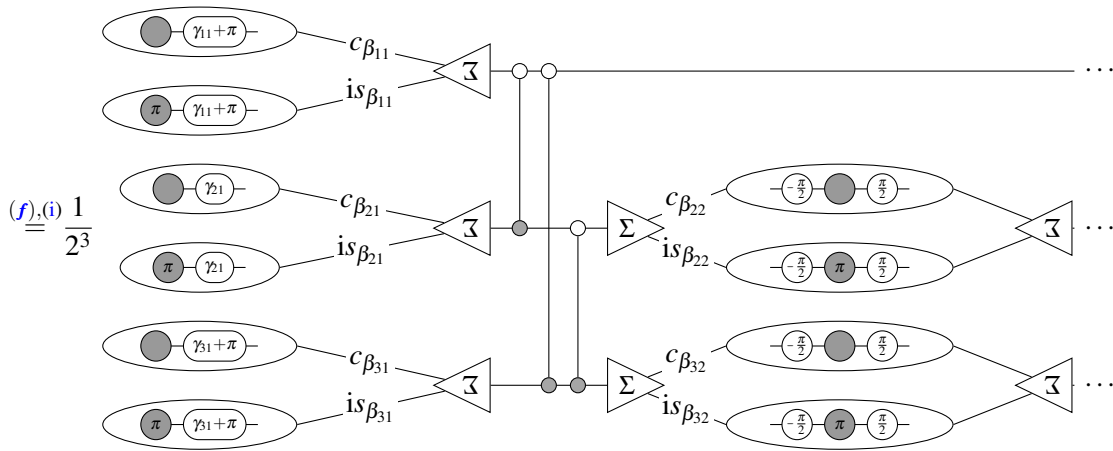
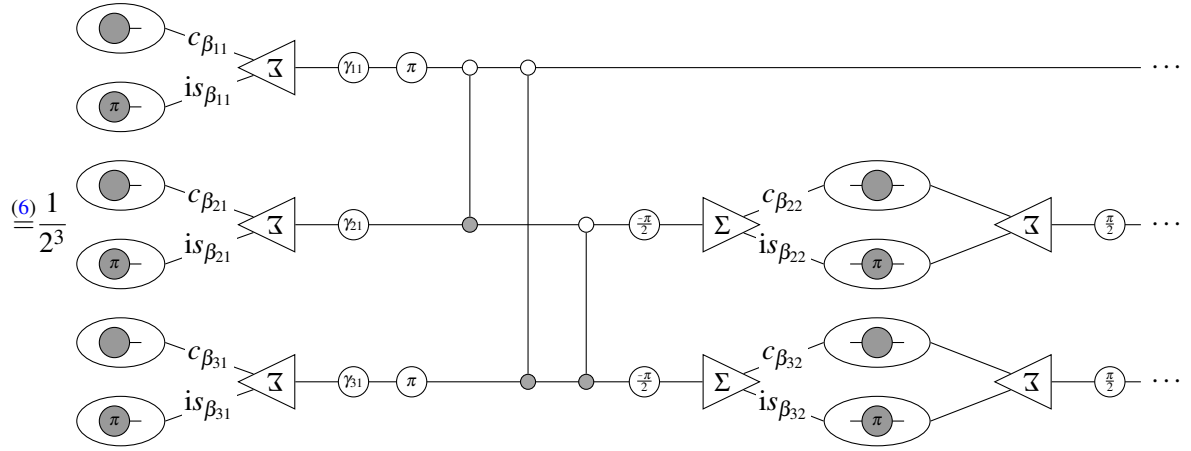
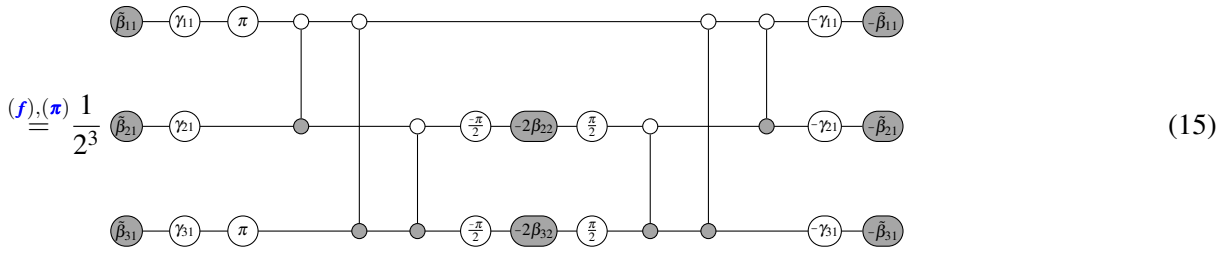
We consider a variant of a hardware efficient SU-2 2-local ansatz from Qiskit [2]. This ansatz was also studied in [24]. For simplicity here we consider a 3-qubit realization,

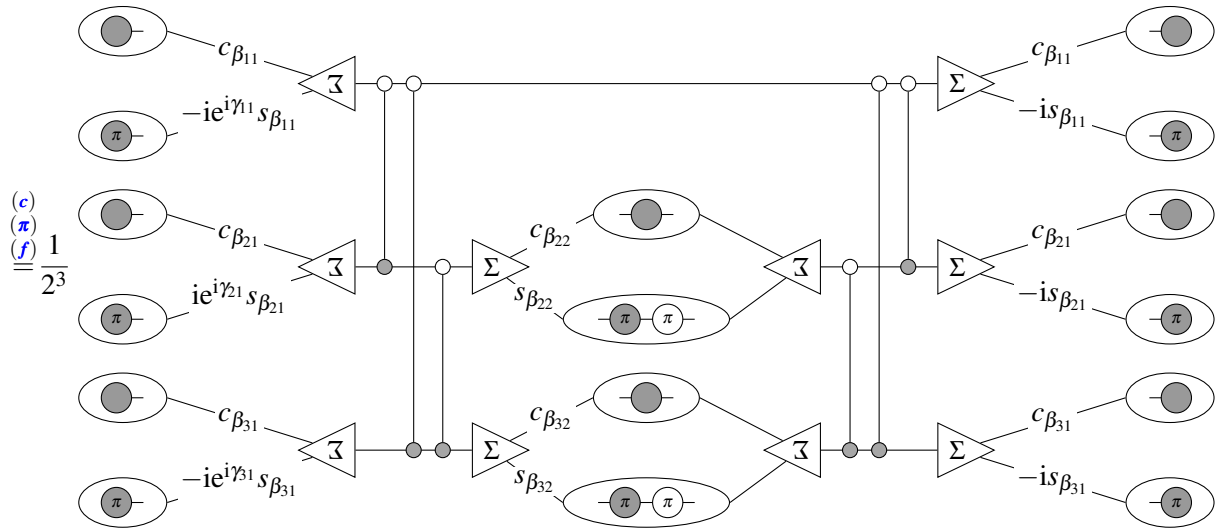


where we conveniently set $\gamma_{ij} := \tilde{\gamma}_{ij} + \frac{\pi}{2}$ and $\beta_{ij} := -\frac{\tilde{\beta}_{ij}}{2}$. To compute the expectation value of a given cost Hamiltonian, we again require expectation values of products of Pauli-Z operators. We demonstrate how such calculations can be performed diagrammatically by considering again MaxCut as an example. For the expectation value corresponding to a given edge (2,3) we have

$$\langle Z_2 Z_3 \rangle =$$







$$\begin{aligned}
 & \stackrel{(17)}{=} c_{\beta_{11}}^2 c_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}} c_{\beta_{32}} - ic_{\beta_{11}}^2 c_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}} s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{31}} \\
 & + ic_{\beta_{11}}^2 c_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}} s_{\beta_{31}} s_{\beta_{32}} - c_{\beta_{11}}^2 c_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{32}} s_{\beta_{31}}^2 e^{i\gamma_{31}} \\
 & + ic_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{32}} e^{i\gamma_{21}} - ic_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{32}} \\
 & + c_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} e^{i\gamma_{21}} e^{i\gamma_{31}} + c_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} e^{i\gamma_{21}} \\
 & + c_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} e^{i\gamma_{31}} + c_{\beta_{11}}^2 c_{\beta_{21}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} \\
 & + ic_{\beta_{11}}^2 c_{\beta_{21}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}}^2 s_{\beta_{32}} e^{i\gamma_{21}} e^{i\gamma_{31}} - ic_{\beta_{11}}^2 c_{\beta_{21}} s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}}^2 s_{\beta_{32}} e^{i\gamma_{31}} \\
 & + c_{\beta_{11}}^2 c_{\beta_{22}} c_{\beta_{31}}^2 c_{\beta_{32}} s_{\beta_{21}}^2 e^{i\gamma_{21}} + ic_{\beta_{11}}^2 c_{\beta_{22}} c_{\beta_{31}} s_{\beta_{21}}^2 s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{21}} e^{i\gamma_{31}} \\
 & - ic_{\beta_{11}}^2 c_{\beta_{22}} c_{\beta_{31}} s_{\beta_{21}}^2 s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{21}} - c_{\beta_{11}}^2 c_{\beta_{22}} c_{\beta_{32}} s_{\beta_{21}}^2 s_{\beta_{31}}^2 e^{i\gamma_{21}} e^{i\gamma_{31}} \\
 & - c_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}}^2 s_{\beta_{11}}^2 s_{\beta_{32}} e^{i\gamma_{11}} + ic_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{31}} e^{i\gamma_{11}} e^{i\gamma_{31}} \\
 & + ic_{\beta_{21}}^2 c_{\beta_{22}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{31}} e^{i\gamma_{11}} - c_{\beta_{21}}^2 c_{\beta_{22}} s_{\beta_{11}}^2 s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{31}} \\
 & + ic_{\beta_{21}} c_{\beta_{31}}^2 c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} e^{i\gamma_{11}} e^{i\gamma_{21}} + ic_{\beta_{21}} c_{\beta_{31}}^2 c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} e^{i\gamma_{11}} \\
 & + c_{\beta_{21}} c_{\beta_{31}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{21}} e^{i\gamma_{31}} - c_{\beta_{21}} c_{\beta_{31}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{21}} \\
 & - c_{\beta_{21}} c_{\beta_{31}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{31}} + c_{\beta_{21}} c_{\beta_{31}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} \\
 & - ic_{\beta_{21}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}}^2 e^{i\gamma_{11}} e^{i\gamma_{21}} e^{i\gamma_{31}} - ic_{\beta_{21}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{22}} s_{\beta_{31}}^2 e^{i\gamma_{11}} e^{i\gamma_{31}} \\
 & + c_{\beta_{22}} c_{\beta_{31}}^2 s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{21}} + ic_{\beta_{22}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{31}} e^{i\gamma_{11}} e^{i\gamma_{21}} e^{i\gamma_{31}} \\
 & + ic_{\beta_{22}} c_{\beta_{31}} c_{\beta_{32}} s_{\beta_{11}}^2 s_{\beta_{21}} s_{\beta_{31}} e^{i\gamma_{11}} e^{i\gamma_{21}} + c_{\beta_{22}} s_{\beta_{11}}^2 s_{\beta_{21}}^2 s_{\beta_{31}} s_{\beta_{32}} e^{i\gamma_{11}} e^{i\gamma_{21}} e^{i\gamma_{31}}, \tag{16}
 \end{aligned}$$

where in the last step we have used the identity

$$\frac{(l_1+m_2+l_3+m_3+r_3)\pi}{2^3} \quad (r_1+m_2+l_3+m_3+r_3)\pi \quad (l_2+m_2+r_2)\pi$$

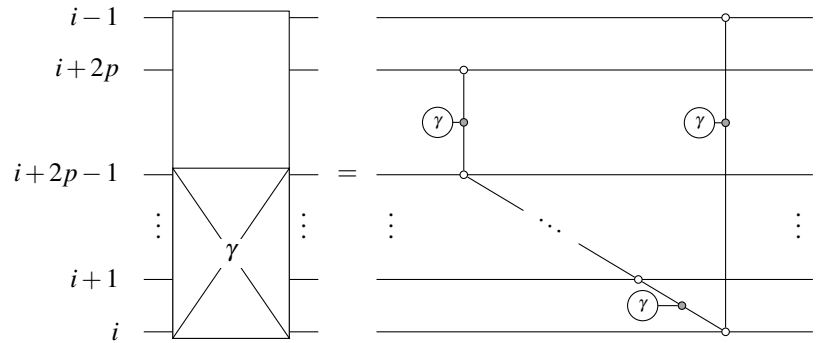
$$= \begin{cases} 0 & \text{if } \begin{array}{l} \ell_1 + m_2 + \ell_3 + m_3 + r_3 \text{ odd} \\ \vee r_1 + m_2 + \ell_3 + m_3 + r_3 \text{ odd} \\ \vee \ell_2 + m_2 + r_2 \text{ odd} \end{array} \\ f_{m_2 m_3}^{r_1 r_2 r_3} & \text{else} \end{cases}, \quad (17)$$

with $\ell_1, \ell_2, \ell_3, m_2, m_3, r_1, r_2, r_3 \in \{0, 1\}^{\times 8}$ and $f_{m_2 m_3}^{r_1 r_2 r_3} := (-1)^{m_2 r_1 + (m_2 \oplus m_3) r_2 + m_3 r_3}$, which is proven in Appendix D.2. Observe that in the second step above any dependency on the parameters γ_{12}, γ_{22} , and γ_{32} was immediately shown to cancel out (due to commuting with the diagonal cost Hamiltonian), and likewise for β_{12} (due to the locality of $Z_2 Z_3$). Similar simplifications are often easily obtained from the diagrammatic perspective.

The formula (16) exemplifies the significant difficulty faced in obtaining analytical results for PQC, even for relatively small ansätze. Nevertheless, in our analysis the complexity remained manageable with the diagrammatic approach up until the very last step, where we applied a simple numerical procedure to collect all the surviving terms (according to (17)) of the contraction. Different hardware-efficient ansätze may be similarly considered, including ones tailored to specific hardware topology. As mentioned, for deeper or more complicated ansätze, analysis may be aided or automated through implementation in software. Here (15) demonstrates how diagrammatic approaches can yield more compact representations of expectation values (as compared to (16)).

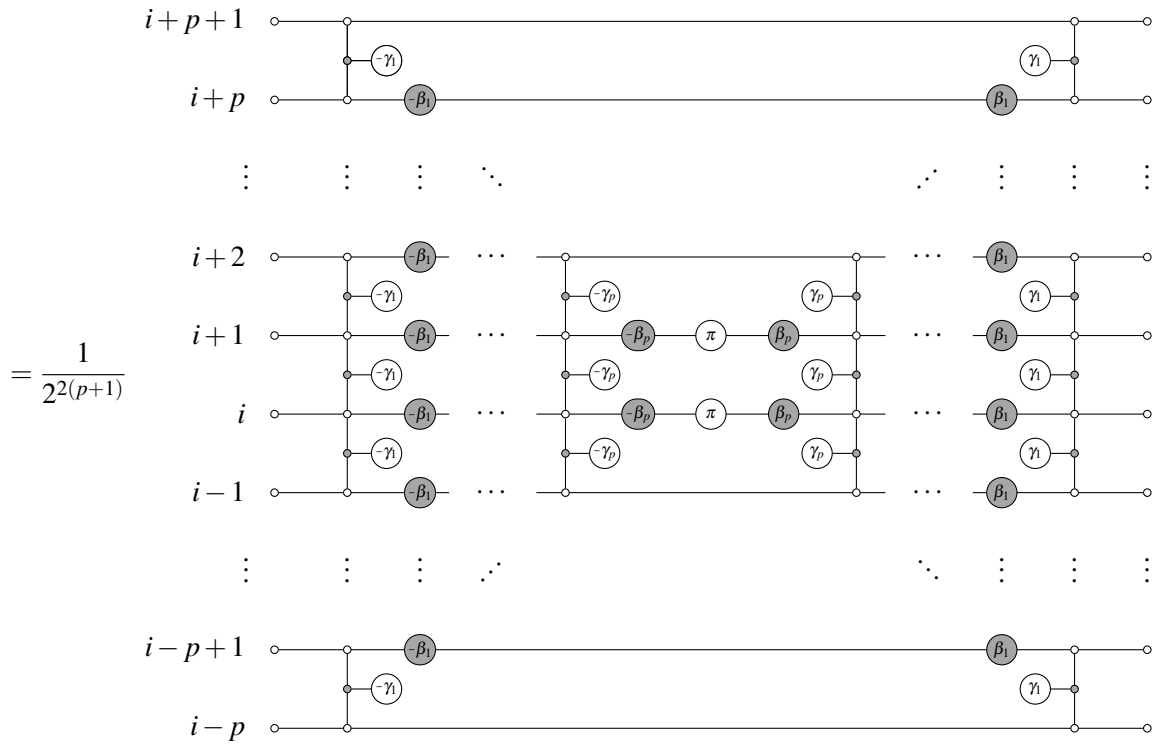
B.2 QAOA for MaxCut on Ring graphs

We consider the simple example of the one-dimensional “ring-of-disagrees”, i.e., 2-regular connected graphs, and rederive the QAOA₁ expectation value as previously shown in [22, 53]. First consider the case of QAOA with arbitrary number of layers p , with $n \gg p$. From the problem symmetry, it suffices to consider the expectation value of a single edge term $\langle Z_i Z_{i+1} \rangle_{\text{QAOA}_p}$. Applying the lightcone rule (9), the outermost reduced phase-separation layer (10) reads

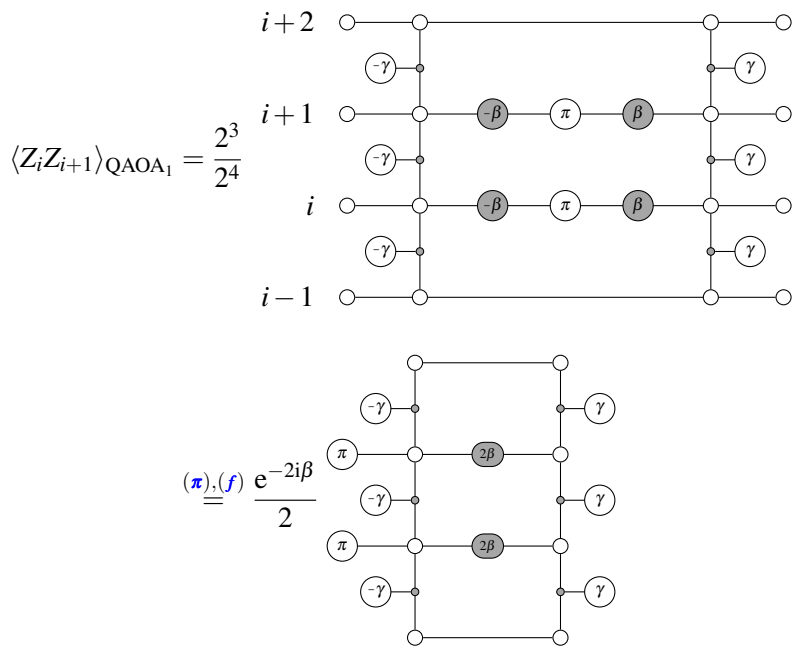


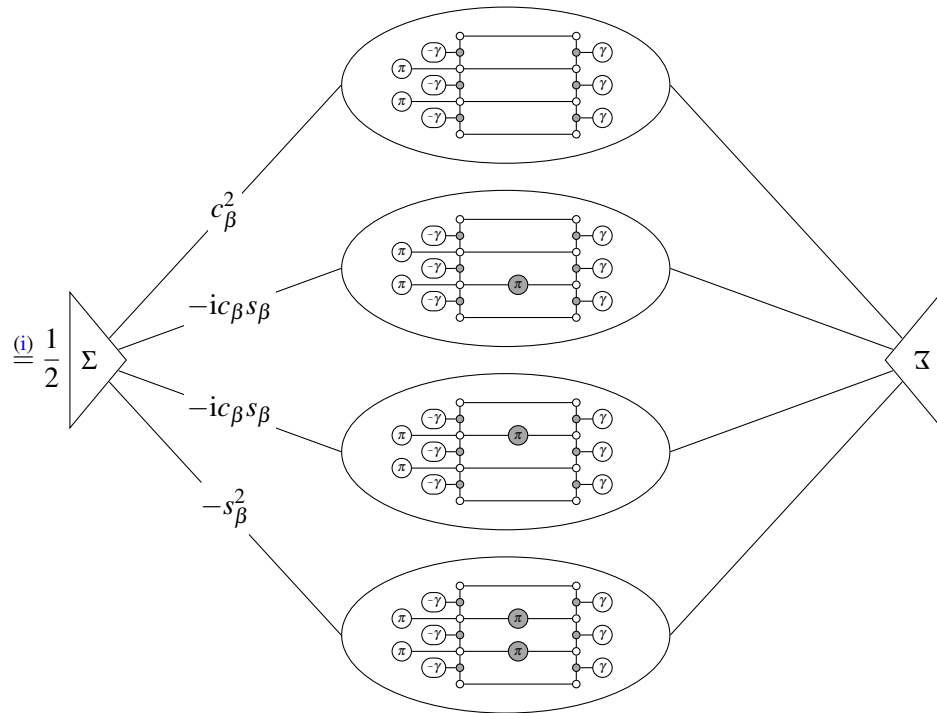
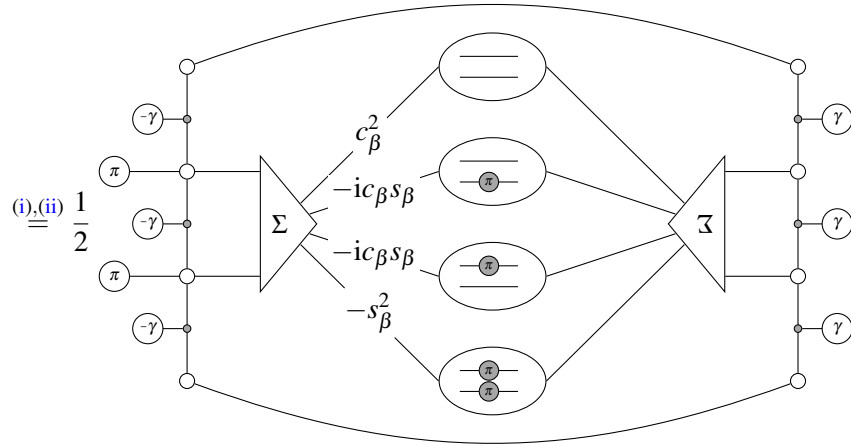
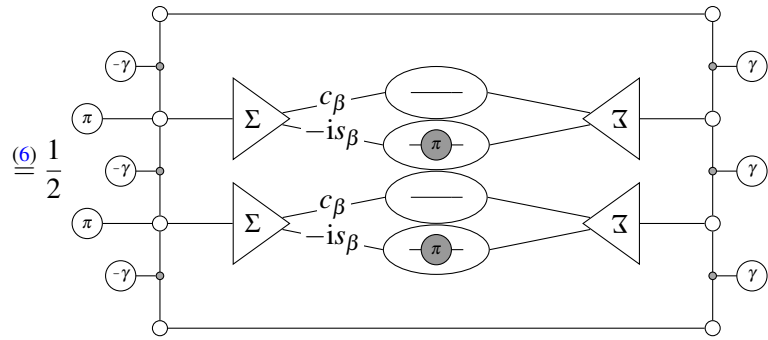
Hence for the QAOA_p expectation value we obtain

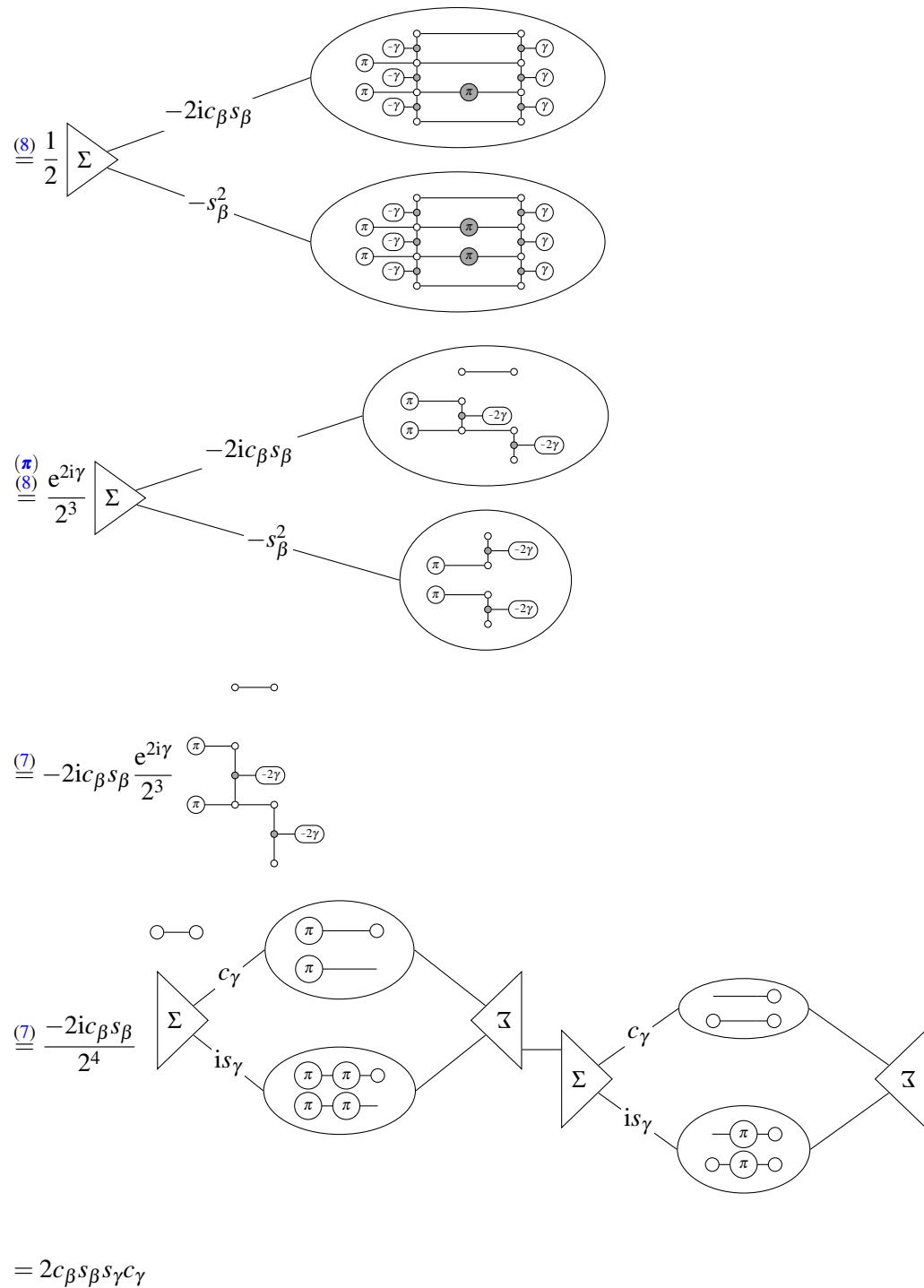
$$\langle Z_i Z_{i+1} \rangle_{\text{QAOA}_p}$$



Observe how the problem and structure again appears in the above diagram (i.e., p -neighborhoods of the edge $(i, i+1)$ are line graphs). Furthermore, the utility of the lightcone rule is clearly demonstrated here. Continuing for the $p = 1$ case, we get







The result is consistent with that of [53, Thm. 1]. The expression obtained for $\langle C \rangle$ is easily optimized to reproduce the performance result obtained numerically for the ring of disagrees in [22]. Similar to the previous examples, here we saw the necessity of our extension for handling X-Z commutations in the third and seventh steps of the derivation above.

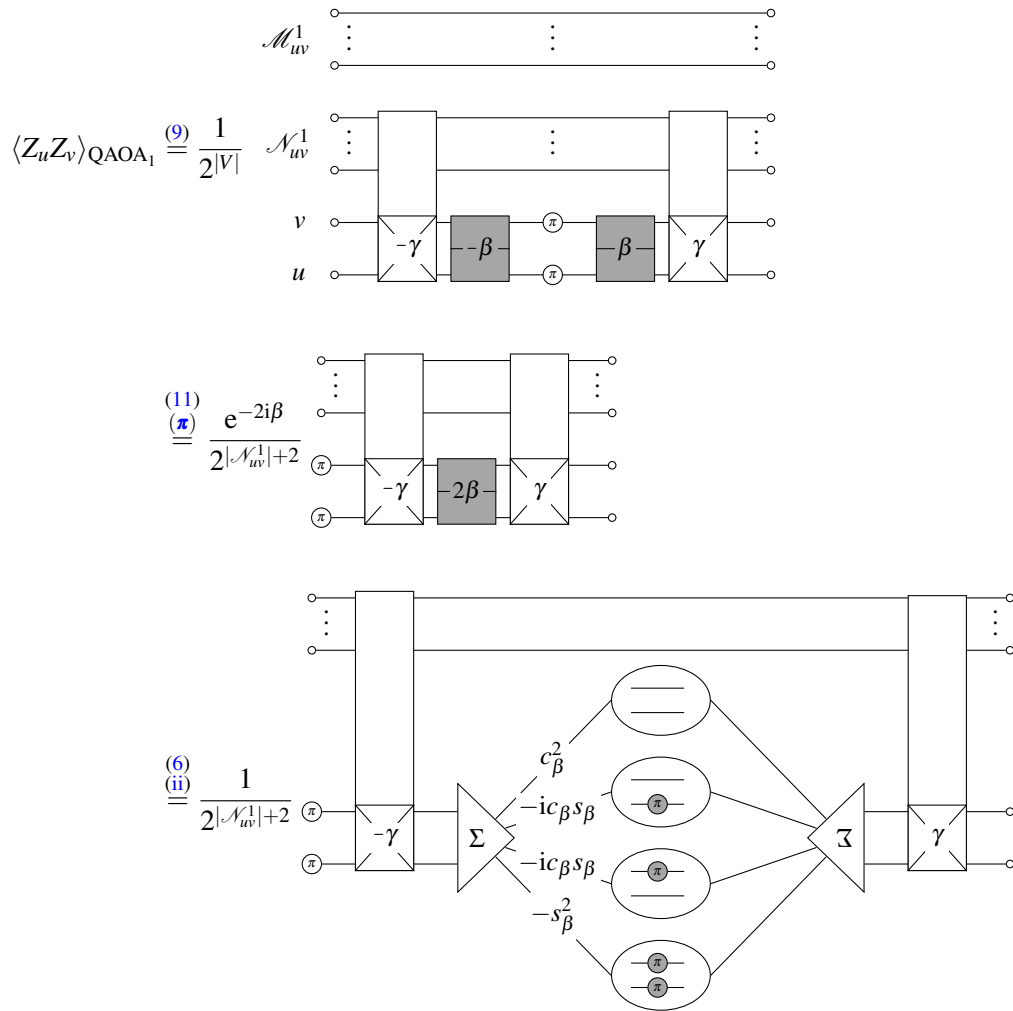
In Appendix B.3 we show the same calculation for MaxCut on general graphs, as obtained for QAOA₁ in [53, Thm. 1]. Similar techniques may be applied and results obtained for a wide variety

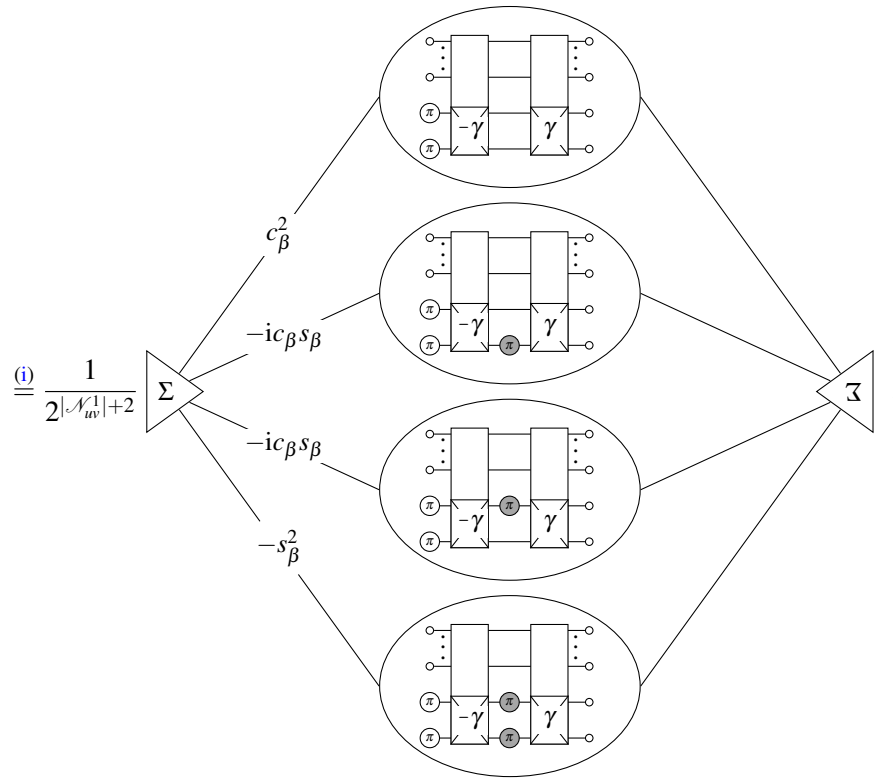
of important problems, for instance quadratic binary optimization problems of which MaxCut is a special case.

B.3 QAOA₁ for MaxCut on General Graphs

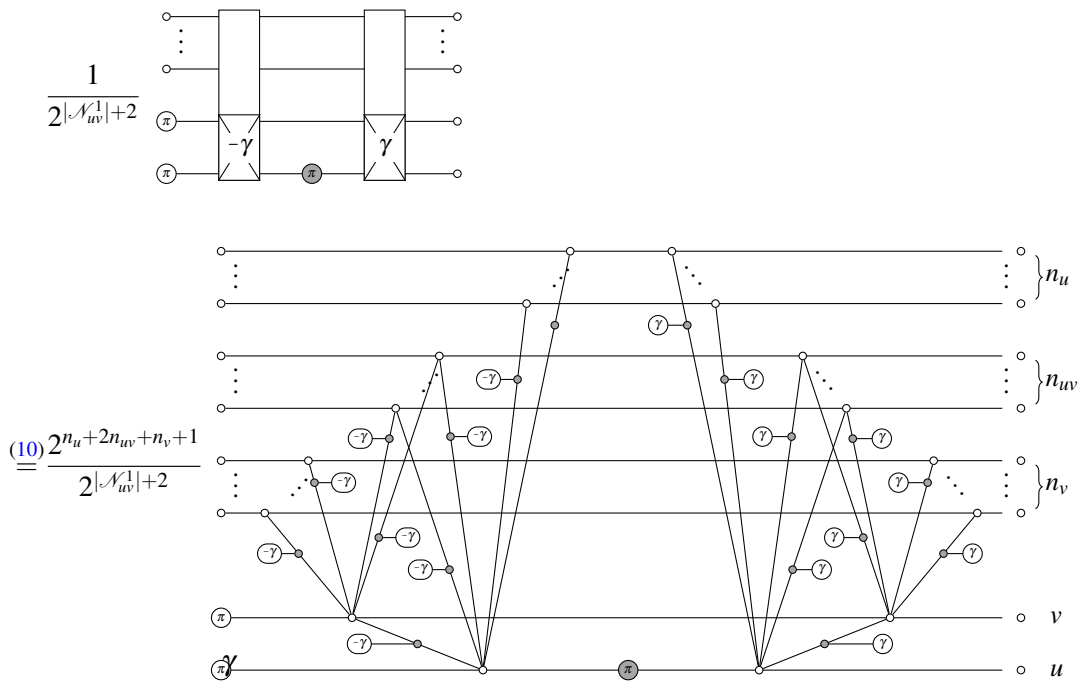
For the QAOA expectation value for MaxCut $\langle C \rangle = \frac{|E|}{2} - \frac{1}{2} \sum_{u,v \in E} \langle Z_u Z_v \rangle$ on general graphs we need to calculate the contributions $\langle Z_u Z_v \rangle$. In this section, we perform the calculation for general graphs in the QAOA, $p = 1$ case, reproducing results obtained in [53].

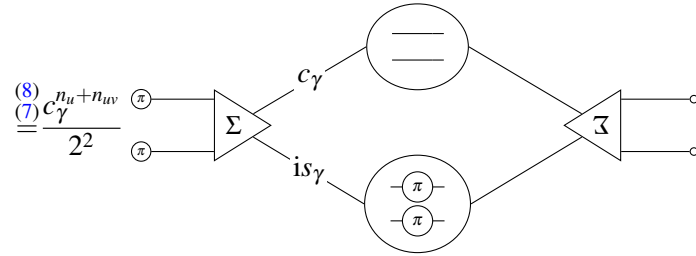
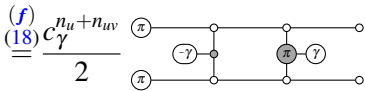
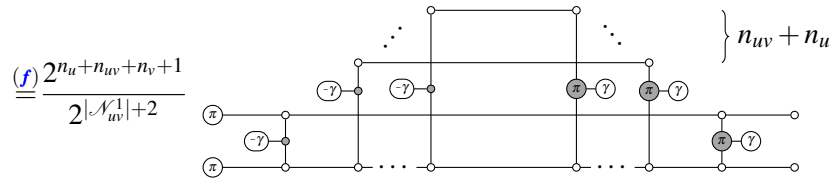
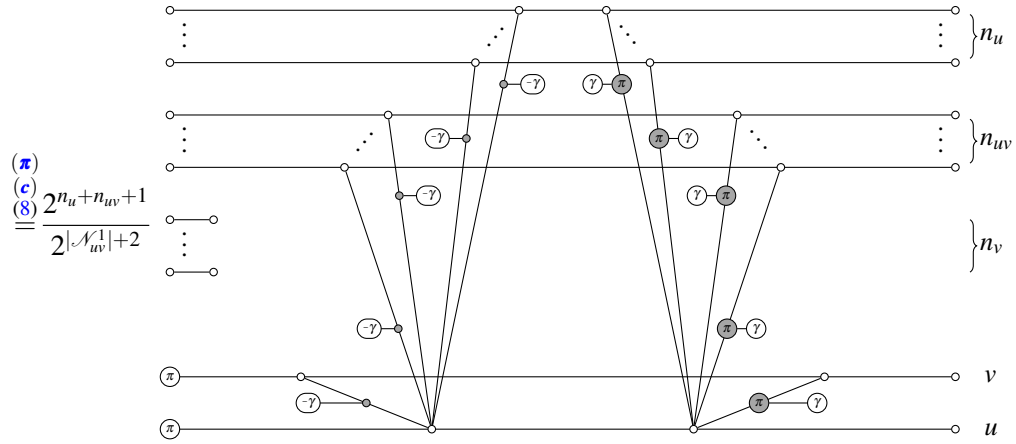
Following the lightcone rule from Equation (9) we obtain for Z-Z terms in the MaxCut QAOA₁ expectation value on a general graph $G = (V, E)$





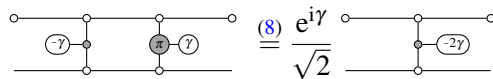
The first summand vanishes and the second and third are linked by symmetry. We continue with the second summand (the I - X -term)

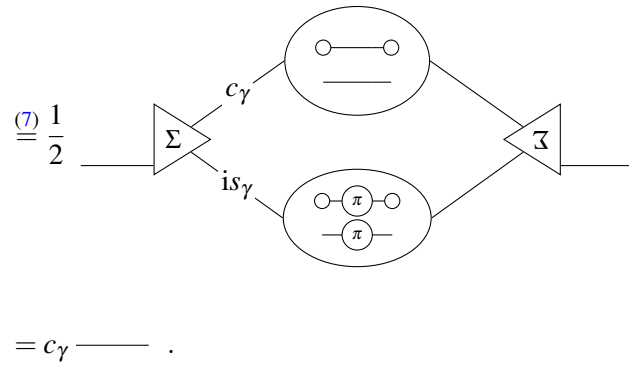




$= i s \gamma C_{\gamma}^{n_u+n_{uv}},$

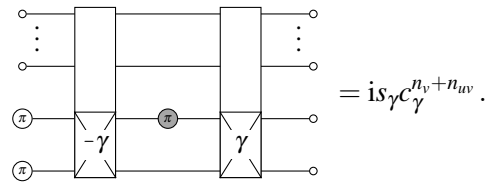
where we have used the size of the exclusive neighborhoods $n_u := |N_u \setminus \{N_v \cup u\}|$, $n_v := |N_v \setminus \{N_u \cup v\}|$, and the joined neighborhood $n_{uv} := |N_u \cap N_v|$, the relation $|N_{uv}^1| = n_u + n_{uv} + n_v$, as well as



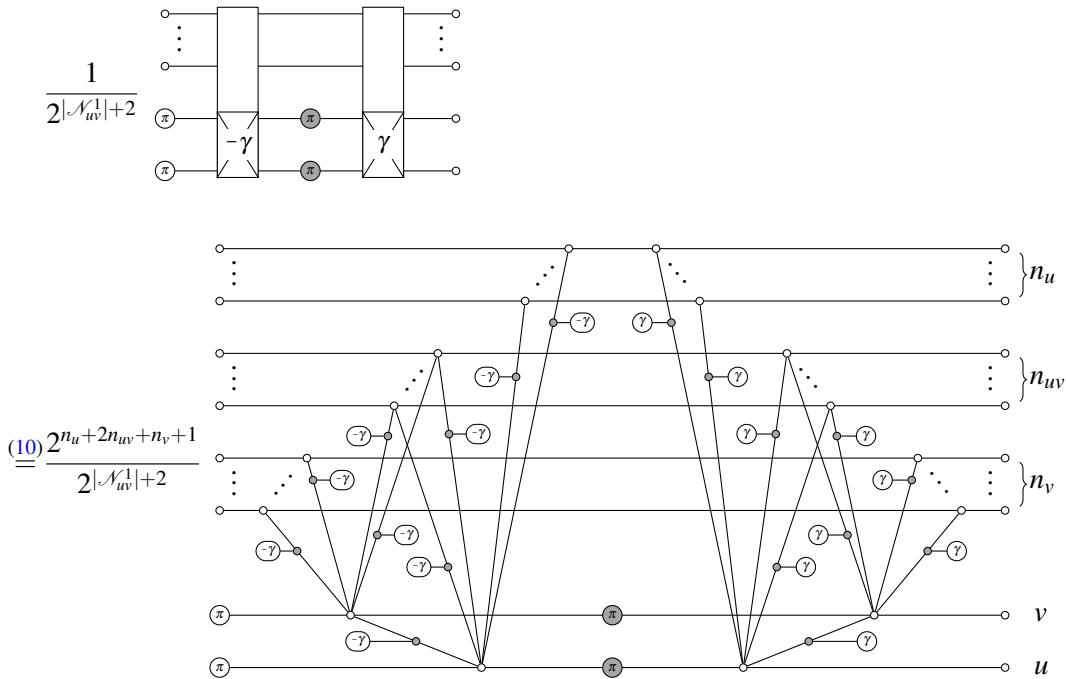


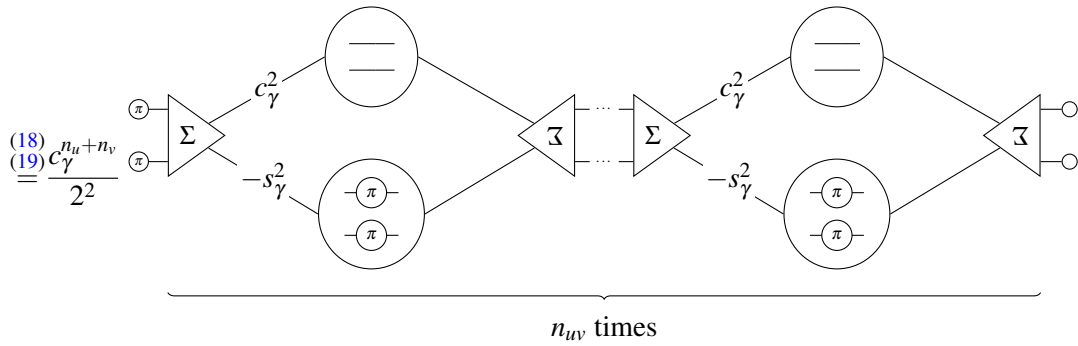
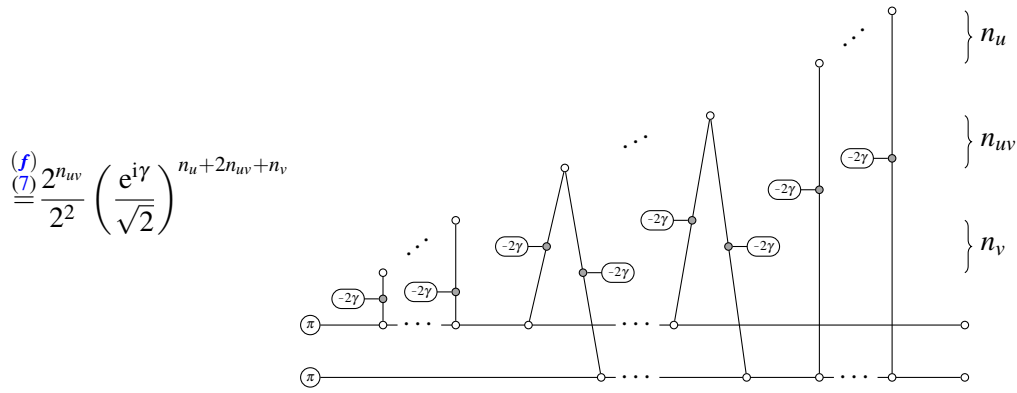
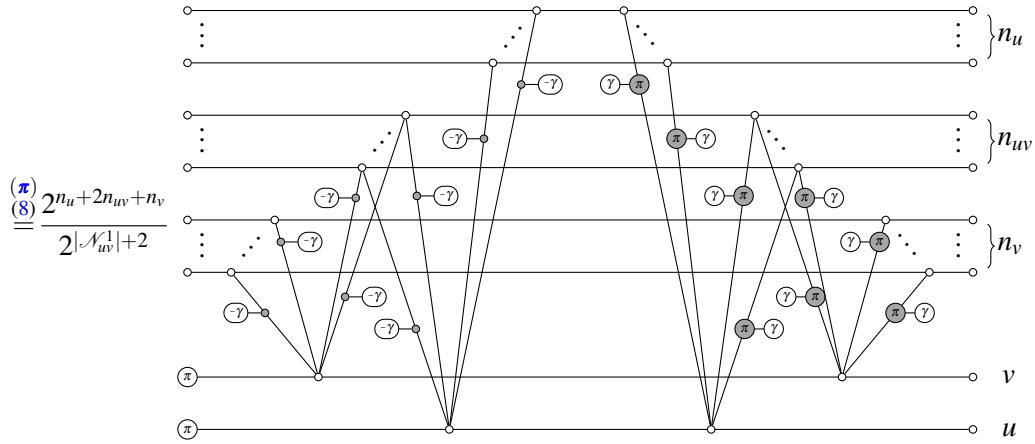
(18)

Analogously the third summand (the X-I-term) can be obtained as



The fourth summand (the X-X-term) reads



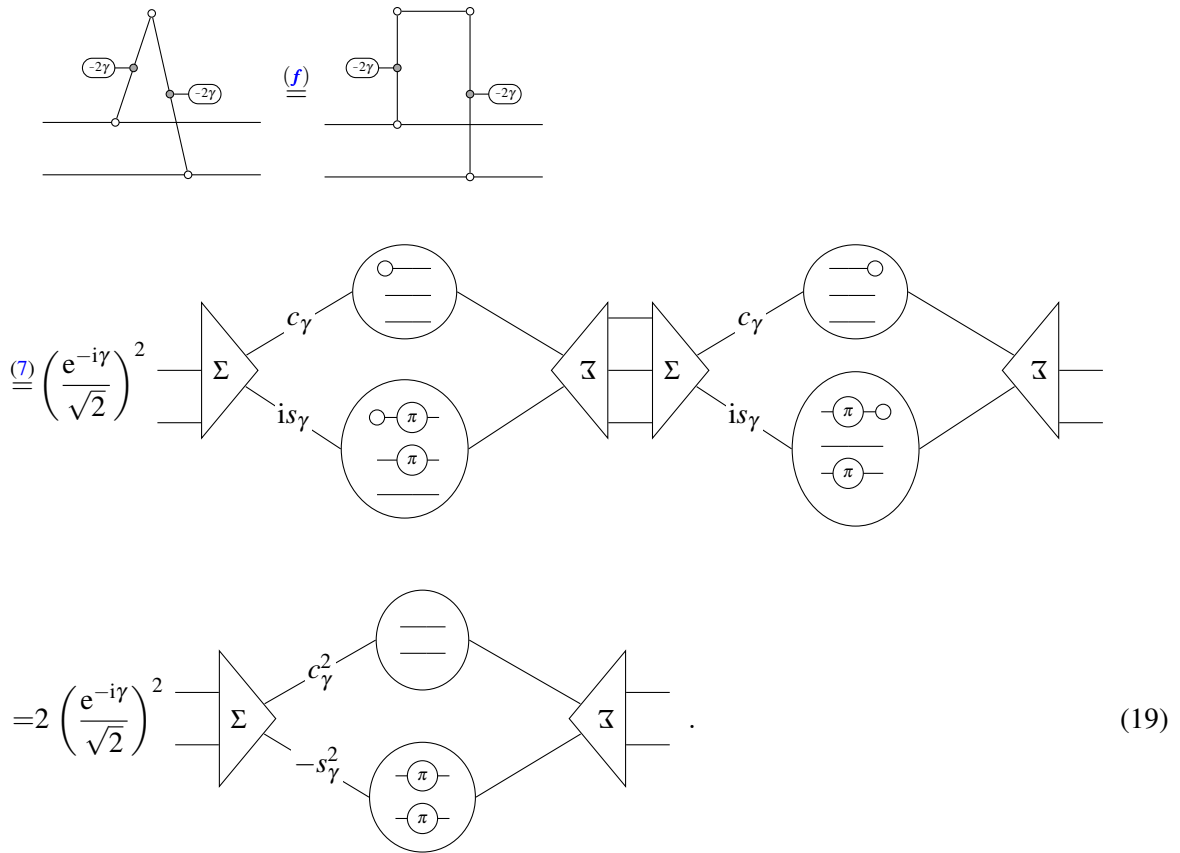


(ii) $\frac{-c_\gamma^{n_u+n_v}}{2^2} \left\{ \begin{pmatrix} n_{uv} \\ 1 \end{pmatrix} s_\gamma^2 c_\gamma^{n_{uv}-2} \begin{array}{c} \pi \quad \pi \\ \pi \quad \pi \end{array} \right.$

$+ \begin{pmatrix} n_{uv} \\ 3 \end{pmatrix} s_\gamma^6 c_\gamma^{n_{uv}-6} \begin{array}{c} \pi \quad \pi \quad \pi \quad \pi \\ \pi \quad \pi \quad \pi \quad \pi \end{array}$

$$\begin{aligned}
 & + \binom{n_{uv}}{5} s_\gamma^{10} c_\gamma^{n_{uv}-10} \begin{array}{c} \circ-\pi-\pi-\pi-\pi-\pi-\pi-\circ \\ \circ-\pi-\pi-\pi-\pi-\pi-\pi-\circ \end{array} \\
 & + \dots \quad \left. \vphantom{\binom{n_{uv}}{5}} \right\} \\
 & = -c_\gamma^{n_u+n_v} \sum_{i=1,3,\dots} \binom{n_{uv}}{i} (s_\gamma^2)^i (c_\gamma^2)^{n_{uv}-i},
 \end{aligned}$$

where we have used



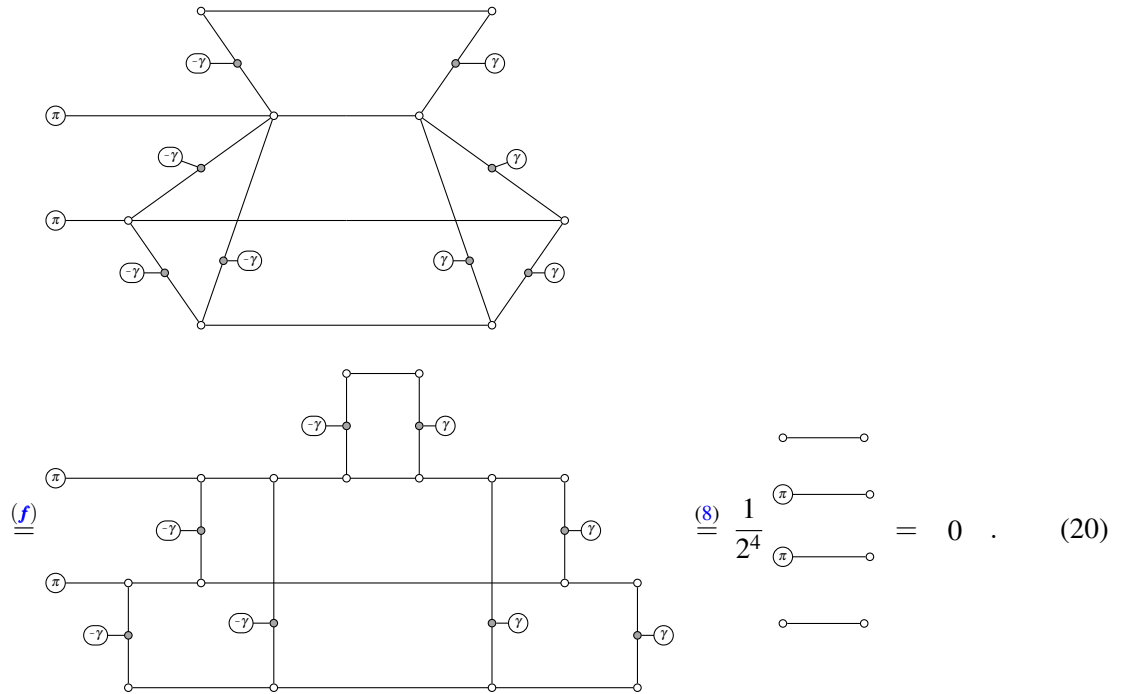
Hence, the total Z-Z-expectation value reads

$$\langle Z_u Z_v \rangle_{\text{QAOA}_1} = c_\beta s_\beta s_\gamma (c_\gamma^{n_u+n_{uv}} + c_\gamma^{n_v+n_{uv}}) + c_\gamma^{n_u+n_v} s_\beta^2 \sum_{i=1,3,\dots} \binom{n_{uv}}{i} (s_\gamma^2)^i (c_\gamma^2)^{n_{uv}-i}.$$

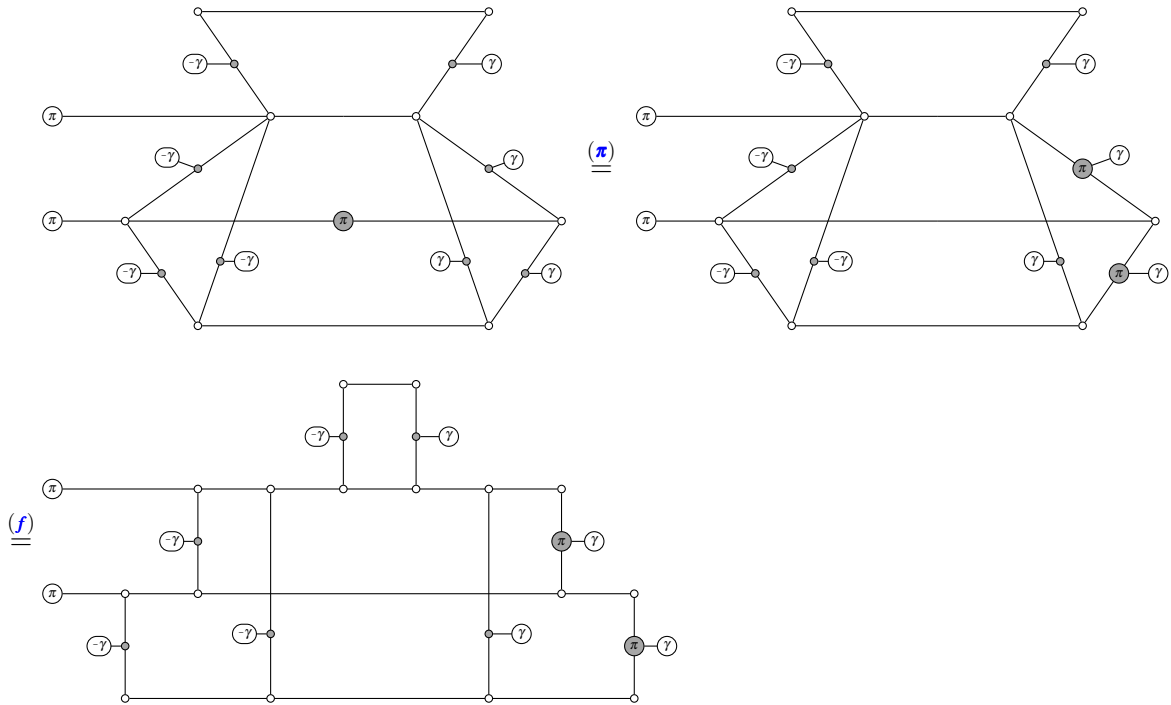
This result is consistent with the corresponding QAOA₁ performance analysis of [53]; applying the binomial theorem to write the sum above in closed form then leads directly to the result of [53, Thm. 1].

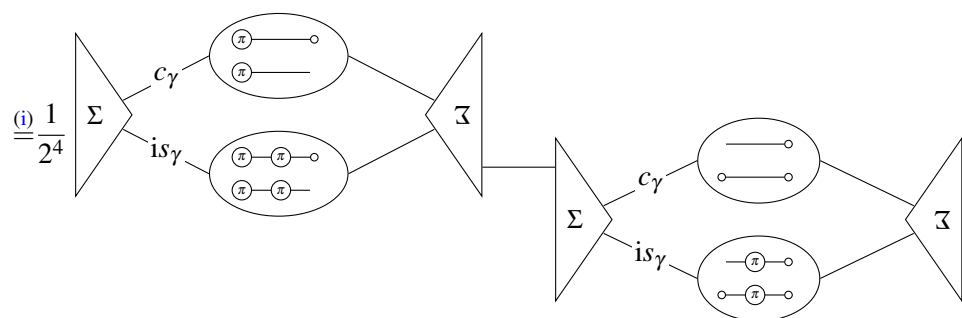
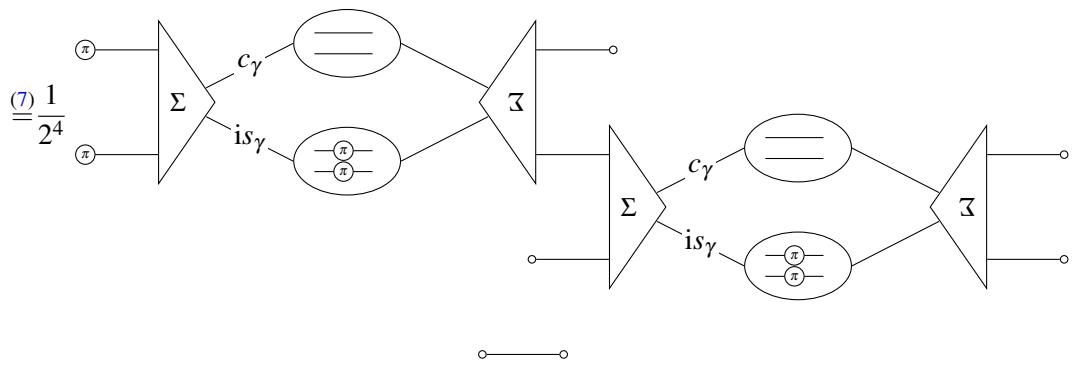
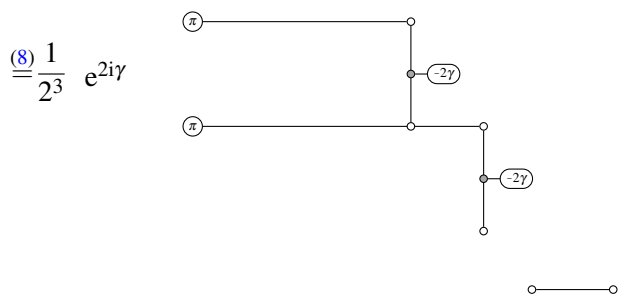
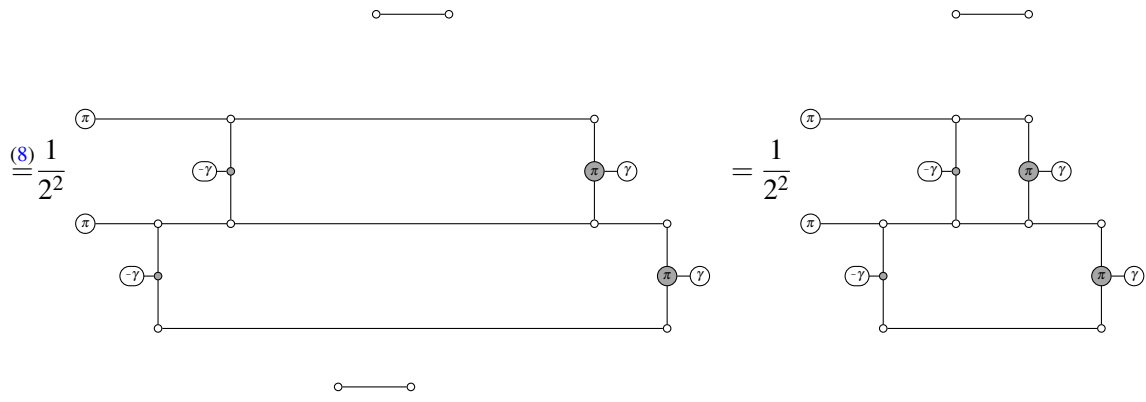
C Details on QAOA₁ for MaxCut on Simple Graph

We calculate each of the four summands in (14). The first summand (the I - I -term) reads



The second summand (the I - X -term) reads

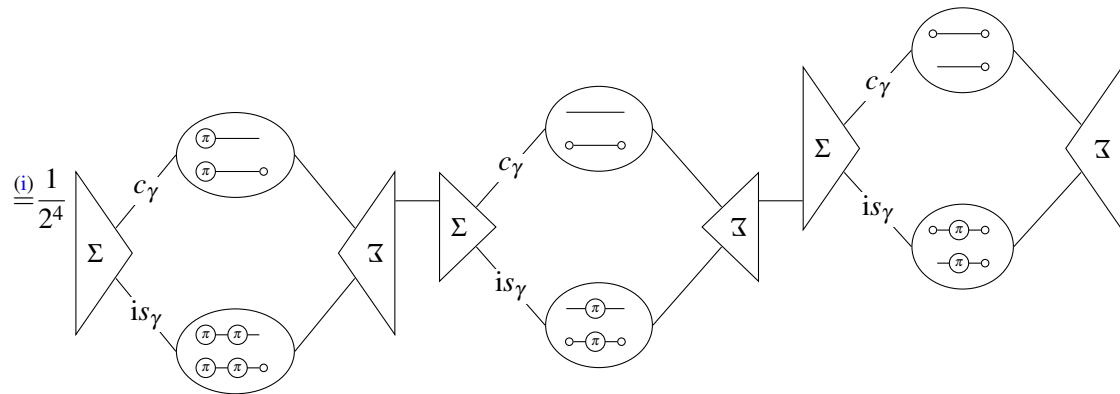
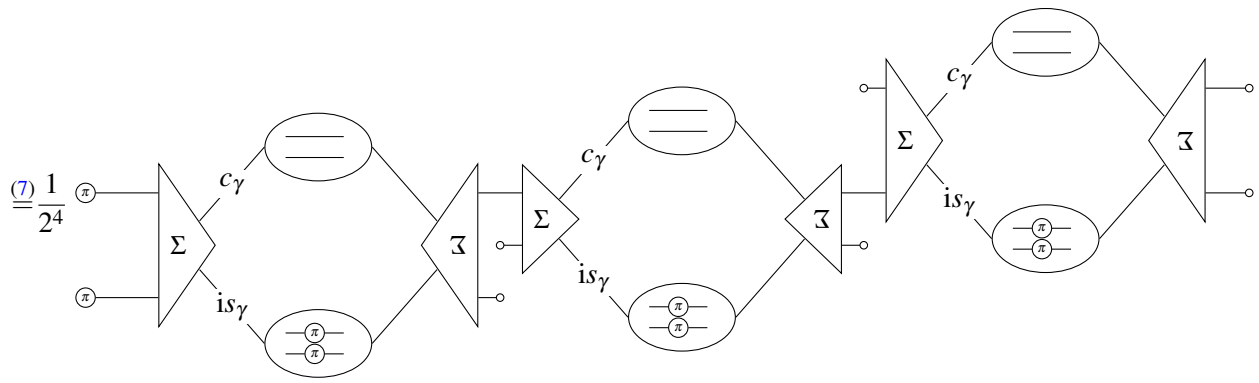




$$\begin{array}{c}
 \textcircled{\text{ii}} \frac{1}{24} i s_{\gamma} c_{\gamma} \\
 \begin{array}{c}
 \textcircled{\pi} \textcircled{\pi} \\
 \textcircled{\pi} \textcircled{\pi} \\
 \textcircled{\pi} \textcircled{\pi}
 \end{array} \\
 \textcircled{\pi} \textcircled{\pi}
 \end{array} = i s_{\gamma} c_{\gamma} \quad (21)$$

The third summand (the X - I -term) reads

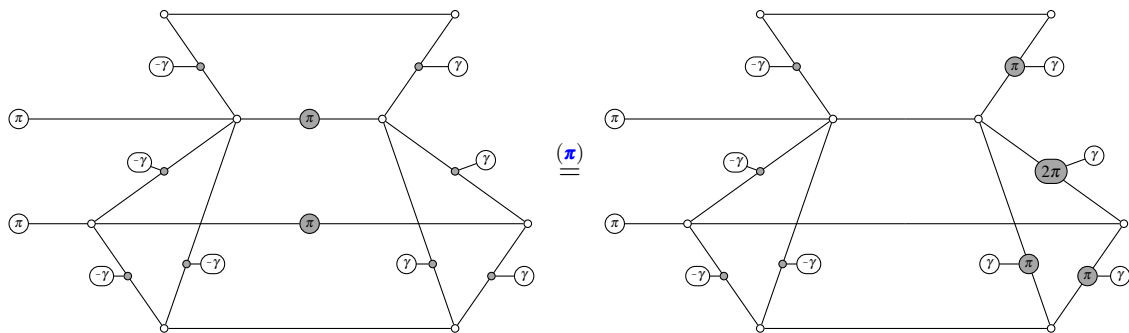
$$\begin{array}{c}
 \textcircled{f} \\
 \textcircled{8} \frac{1}{2} \\
 \textcircled{8} \frac{1}{2} e^{3i\gamma}
 \end{array}$$

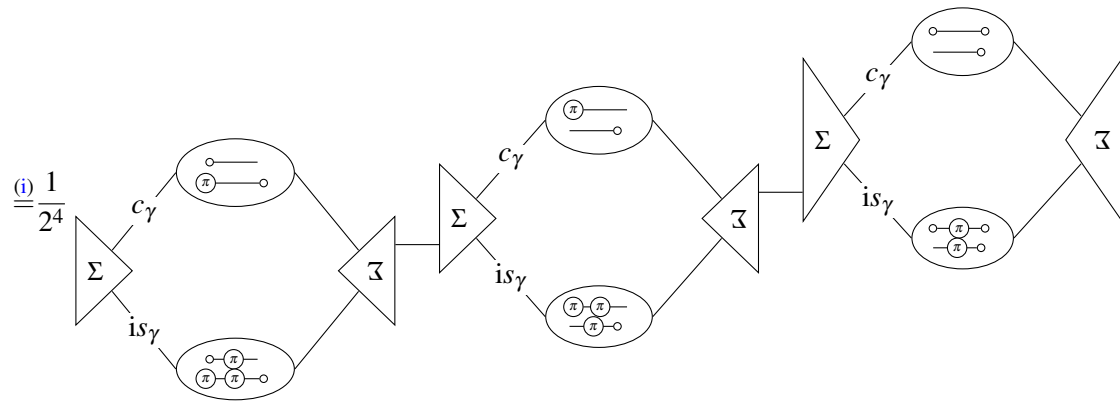
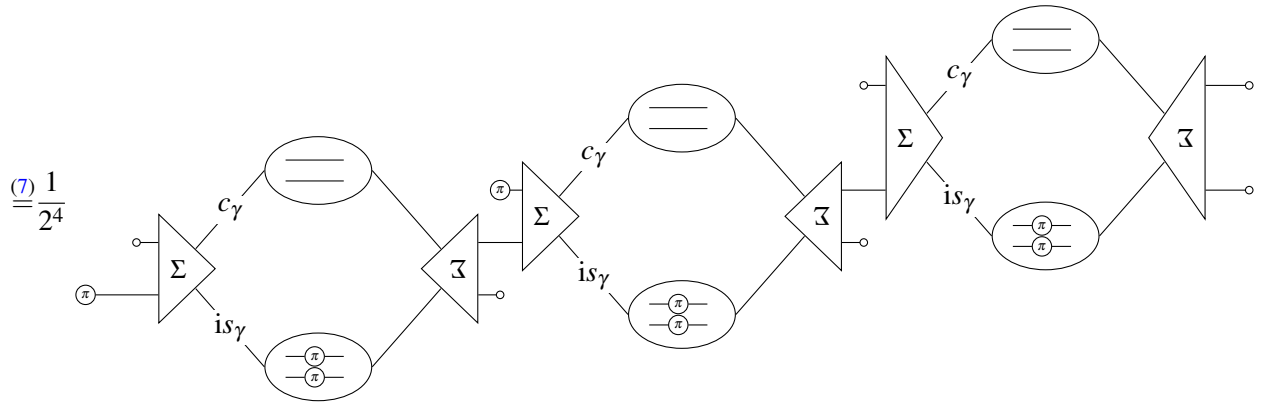
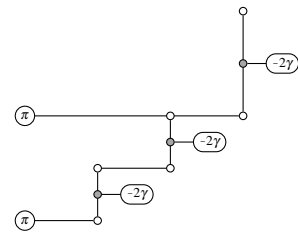
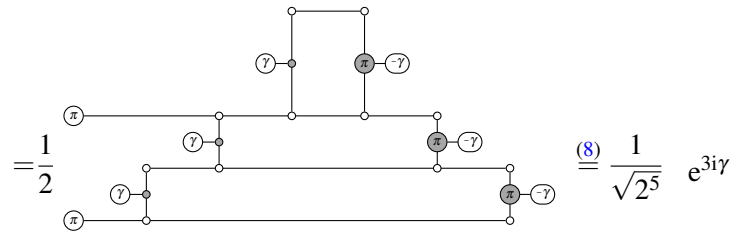
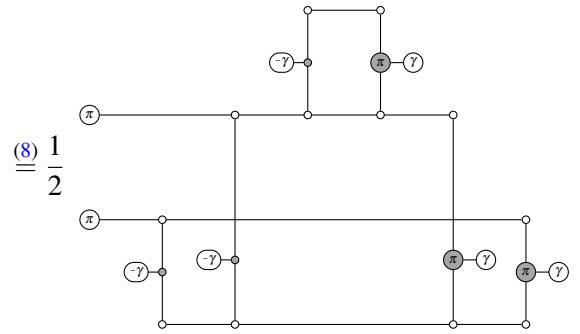
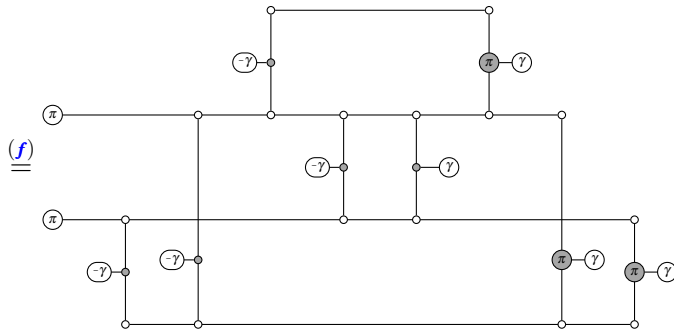


(ii) $\frac{1}{2^4} is\gamma c\gamma^2$

(22)

The fourth summand (the X-X-term) reads



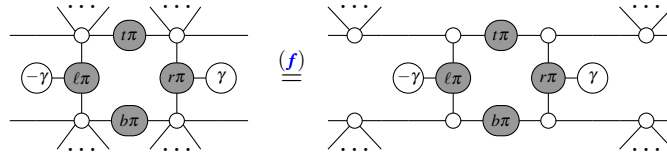


$$\begin{aligned}
 & \text{---} \circ \\
 & \begin{matrix} \circ \text{---} \pi \text{---} \circ \\ \circ \text{---} \pi \text{---} \pi \text{---} \circ \\ \circ \text{---} \pi \text{---} \circ \end{matrix} \\
 & \begin{matrix} \circ \text{---} \pi \text{---} \circ \\ \circ \text{---} \pi \text{---} \circ \end{matrix}
 \end{aligned}
 = -s_\gamma^2 c_\gamma \quad (23)$$

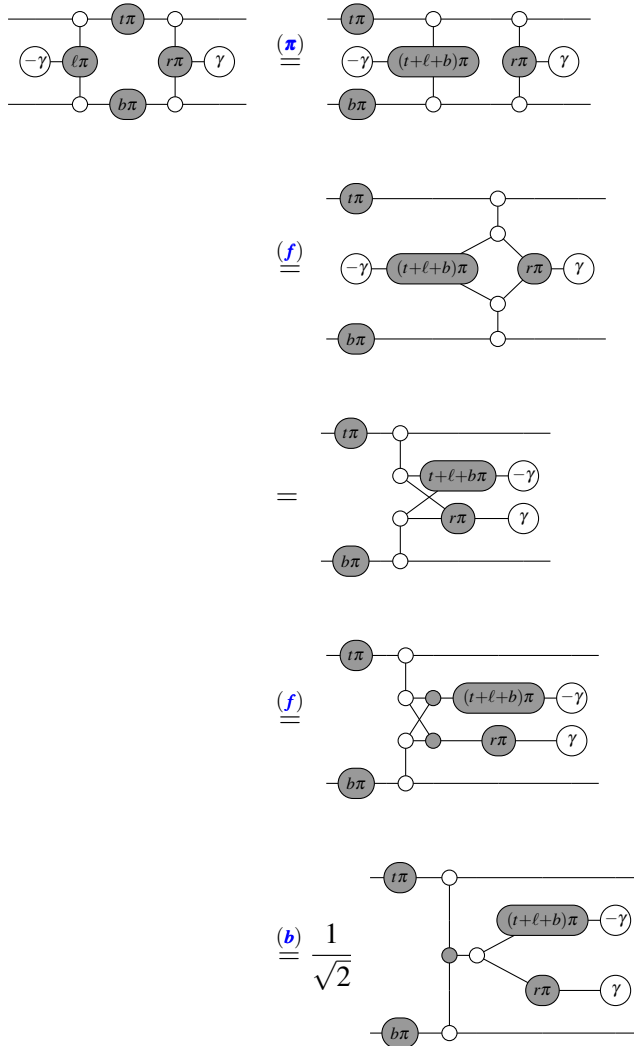
D Proofs of useful ZX-diagram Identities

D.1 Phase-gadget identity

Proof of (8). First, we can use the spider fusion rule to write



Then, we just consider the inner part



$$(\boldsymbol{\pi})(f) \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} \gamma \text{---} (t+l+b+r)\pi \text{---} -\gamma \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} b\pi \text{---} \circ \text{---} \end{array} .$$

If $t + l + b + r$ even, we have

$$\begin{aligned} \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} \gamma \text{---} \circ \text{---} -\gamma \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} b\pi \text{---} \circ \text{---} \end{array} &\stackrel{(id),(f)}{=} \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} \circ \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} b\pi \text{---} \circ \text{---} \end{array} \\ &\stackrel{(c)}{=} \frac{1}{2} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} \circ \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} b\pi \text{---} \circ \text{---} \end{array} \\ &\stackrel{(id),(f)}{=} \frac{1}{2} \begin{array}{c} \text{---} t\pi \text{---} \\ \text{---} b\pi \text{---} \end{array} , \end{aligned}$$

else, if $t + l + b + r$ odd, we have

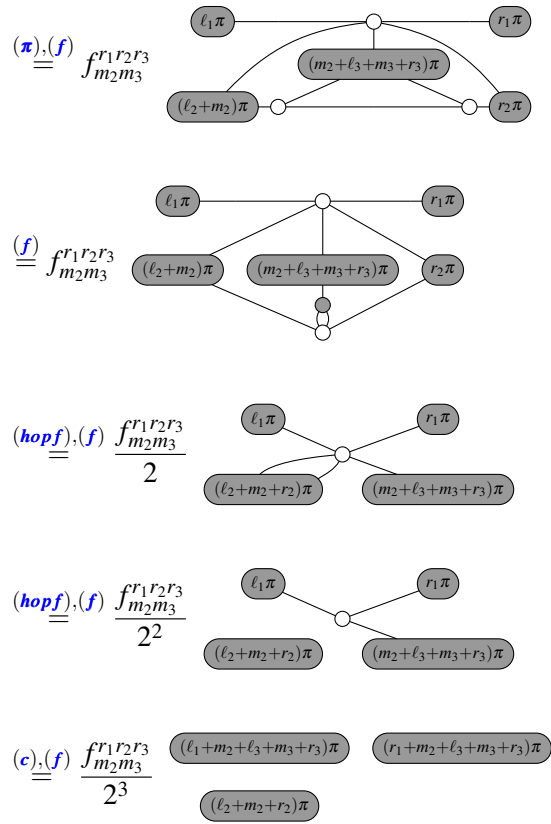
$$\frac{1}{\sqrt{2}} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} \gamma \text{---} \pi \text{---} -\gamma \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} b\pi \text{---} \circ \text{---} \end{array} \stackrel{(f),(\boldsymbol{\pi})}{=} \frac{1}{\sqrt{2}} e^{i\gamma} \begin{array}{c} \text{---} (t+r)\pi \text{---} \circ \text{---} r\pi \text{---} \\ | \quad \quad \quad | \\ \circ \text{---} -2\gamma \text{---} \circ \\ | \quad \quad \quad | \\ \text{---} (b+1)\pi \text{---} \circ \text{---} \pi \text{---} \end{array} ,$$

which proves (8). □

D.2 Hardware Efficient Ansatz

Proof of (17).

$$\begin{aligned} &\begin{array}{c} \ell_1\pi \text{---} \circ \text{---} r_1\pi \\ | \quad \quad \quad | \\ \ell_2\pi \text{---} \circ \text{---} m_2\pi \text{---} m_2\pi \text{---} \circ \text{---} r_2\pi \\ | \quad \quad \quad | \\ \ell_3\pi \text{---} \circ \text{---} m_3\pi \text{---} m_3\pi \text{---} \circ \text{---} r_3\pi \end{array} \stackrel{(\boldsymbol{\pi})(f)}{=} \begin{array}{c} \ell_1\pi \text{---} \circ \text{---} r_1\pi \\ | \quad \quad \quad | \\ \ell_2\pi \text{---} \circ \text{---} m_2\pi \text{---} \circ \text{---} (m_2+m_3)\pi \text{---} r_2\pi \\ | \quad \quad \quad | \\ \ell_3\pi \text{---} \circ \text{---} m_3\pi \text{---} \circ \text{---} m_3\pi \text{---} r_3\pi \end{array} \\ &\stackrel{(c),(\boldsymbol{\pi}), (f)}{=} \underbrace{(-1)^{m_2 r_1 + (m_2 \oplus m_3) r_2 + m_3 r_3}}_{=: f_{m_2 m_3}^{r_1 r_2 r_3}} \begin{array}{c} \ell_1\pi \text{---} \circ \text{---} r_1\pi \\ | \quad \quad \quad | \\ \ell_2\pi \text{---} \circ \text{---} m_2\pi \text{---} \circ \text{---} r_2\pi \\ | \quad \quad \quad | \\ \ell_3\pi \text{---} \circ \text{---} m_3\pi \text{---} \circ \text{---} r_3\pi \end{array} \\ &\stackrel{(f)}{=} f_{m_2 m_3}^{r_1 r_2 r_3} \begin{array}{c} \ell_1\pi \text{---} \circ \text{---} r_1\pi \\ | \quad \quad \quad | \\ \ell_2\pi \text{---} \circ \text{---} m_2\pi \text{---} \circ \text{---} r_2\pi \\ | \quad \quad \quad | \\ \ell_3\pi \text{---} \circ \text{---} m_3\pi \text{---} \circ \text{---} r_3\pi \end{array} \stackrel{(f)}{=} f_{m_2 m_3}^{r_1 r_2 r_3} \begin{array}{c} \ell_1\pi \text{---} \circ \text{---} r_1\pi \\ | \quad \quad \quad | \\ \ell_2\pi \text{---} \circ \text{---} (l_3+m_3+r_3)\pi \text{---} r_2\pi \\ | \quad \quad \quad | \\ \ell_3\pi \text{---} \circ \text{---} m_2\pi \text{---} \circ \text{---} r_2\pi \end{array} \end{aligned}$$



□

A Biset-Enriched Categorical Model for Proto-Quipper with Dynamic Lifting

Peng Fu

Dalhousie University

Kohei Kishida

University of Illinois at Urbana-Champaign

Neil J. Ross

Dalhousie University

Peter Selinger

Dalhousie University

Quipper and Proto-Quipper are a family of quantum programming languages that, by their nature as circuit description languages, involve two runtimes: one at which the program generates a circuit and one at which the circuit is executed, normally with probabilistic results due to measurements. Accordingly, the language distinguishes two kinds of data: parameters, which are known at circuit generation time, and states, which are known at circuit execution time. Sometimes, it is desirable for the results of measurements to control the generation of the next part of the circuit. Therefore, the language needs to turn states, such as measurement outcomes, into parameters, an operation we call *dynamic lifting*. The goal of this paper is to model this interaction between the runtimes by providing a general categorical structure enriched in what we call “bisets”. We demonstrate that the biset-enriched structure achieves a proper semantics of the two runtimes and their interaction, by showing that it models a variant of Proto-Quipper with dynamic lifting. The present paper deals with the concrete categorical semantics of this language, whereas a companion paper [7] deals with the syntax, type system, operational semantics, and abstract categorical semantics.

1 Introduction

Quipper [9, 10] is a functional programming language for designing quantum circuits. It shares many properties with hardware description languages. For example, Quipper distinguishes two kinds of runtime: (i) Circuit generation time. This is when a quantum circuit is generated on a classical computer. (ii) Circuit execution time. This is when a quantum circuit is run on a quantum computer or simulator. As a result of these two runtimes, Quipper makes a distinction between (i) *parameters* and (ii) *states*. A parameter is a value known at circuit generation time, such as a boolean for an if-then-else expression. A state is a value only known at circuit execution time, such as the state of a qubit or a bit in a circuit.

The distinction between parameters and states reflects the assumption that classical computers and quantum devices may reside in different physical locations and that they cooperate to perform computations. This is also an assumption shared by the quantum computing model QRAM [12]. In practice, the computation in a quantum device can interleave with the computation in a classical computer. This means that there should be a mechanism to turn the results of measurements, which are states, into parameters. Dynamic lifting is a construct that makes this possible in the programming language. It lifts the result of a measurement from a quantum computer to a boolean in the programming language, where it can then be used as a parameter in the construction of the rest of the circuit. This enables more general post-processing for quantum computation than the simpler model where all measurements are done at the end. Some quantum algorithms, such as those involving magic state distillation, require dynamic lifting, while many others do not.

Since Quipper is implemented as an embedded language in the host language Haskell, it does not have a formal semantics. Proto-Quipper [6, 8, 17, 18] is a family of quantum programming languages that

are intended to provide Quipper with a formal foundation such as operational and categorical semantics. Like Quipper, Proto-Quipper has the two runtimes and distinguishes between parameters and states.

The semantics of the two runtimes depends on the meaning of “circuit” and “quantum operation”. Rather than fixing one specific kind of circuit or quantum operation, the programming language is parametric on two small categories \mathbf{M} and \mathbf{Q} , which are assumed to be given but otherwise arbitrary, subject to some conditions. The first of these is a symmetric monoidal category \mathbf{M} , whose morphisms represent quantum circuits. The second is a symmetric monoidal category \mathbf{Q} , whose morphisms represent quantum operations. We note that there is an important conceptual difference between these categories. The morphisms of \mathbf{M} represent circuits as *syntactic* entities. For example, Quipper allows circuits to be *boxed*, which turns them into a data structure that can be inspected and operated on. A boxed circuit may then be reversed, printed, iterated over, etc. Thus, \mathbf{M} is typically a free category generated by some collection of (quantum and classical) gates. Measurement can be supported in the category \mathbf{M} , but it will merely be a gate in a circuit, turning a qubit into a classical bit of the circuit. On the other hand, the category \mathbf{Q} represents quantum operations, which are *physical* entities. Typically, \mathbf{Q} is a category of superoperators (which include unitary operations and measurements). We assume that \mathbf{M} and \mathbf{Q} have the same objects, and that there is a symmetric monoidal *interpretation functor* $J : \mathbf{M} \rightarrow \mathbf{Q}$, which interprets circuits by the quantum operations they embody.

We emphasize that measurement and dynamic lifting are two different concepts that should not be confused. Measurement is merely a gate in a quantum circuit, which turns a qubit (a state) into a classical bit (also a state). On the other hand, dynamic lifting is an operation of the programming language, which turns a classical bit (a state) into a boolean of the programming language (a parameter). In the categorical semantics, measurement is a morphism $\mathbf{Qubit} \rightarrow \mathbf{Bit}$ in the categories \mathbf{M} and \mathbf{Q} . On the other hand, dynamic lifting is not a morphism in \mathbf{M} or in \mathbf{Q} ; rather, it is a morphism in a certain Kleisli category.

Specifically, in our recent work [7], we proposed a type system, an operational semantics and an abstract categorical semantics for a version of Proto-Quipper with dynamic lifting, which is called Proto-Quipper-Dyn. Dynamic lifting is modeled as a map $\mathbf{Bit} \rightarrow T\mathbf{Bool}$, where T is a commutative strong monad, such that the following diagram commutes.

$$\begin{array}{ccc}
 & & \mathbf{Bit} \\
 & \nearrow \text{init} & \\
 \mathbf{Bool} & \xrightarrow{\eta} & T\mathbf{Bool} \\
 & & \downarrow \text{dynlift}
 \end{array}$$

We have shown in [7] that our categorical model is sound with respect to the type system and operational semantics of the language. However, the categorical semantics in [7] is purely abstract, simply listing the properties that such a categorical model must have, without showing that such a category actually exists or giving an example of one.

In this paper, we construct a concrete model for the general categorical semantics of [7]. Constructing such a model is challenging because it requires a novel combination of quantum circuits (morphisms in \mathbf{M}) and quantum operations (morphisms in \mathbf{Q}): The categorical model must be able to account for both quantum circuits and quantum operations, as well as operations such as boxing, dynamic lifting, and of course higher-order functions.

Our technical innovation to make all of this work is *biset enrichment*. A *biset* is an object in the category $\mathbf{Set}^{2^{\text{op}}}$, or, more concretely, it is a triple (X_0, X_1, f) of sets X_0, X_1 and a function $f : X_1 \rightarrow X_0$. A morphism of bisets is an obvious commutative square. We will consider categories enriched in bisets. Concretely, such a category has one kind of objects, but two kinds of morphisms, which we use to model quantum circuits and quantum operations, respectively. Our construction is based on a biset-enriched category \mathbf{C} constructed from \mathbf{M} and \mathbf{Q} . Its objects are the same as those of \mathbf{M} and \mathbf{Q} , and its hom-bisets

are $(\mathbf{Q}(A, B), \mathbf{M}(A, B), J_{A, B})$, where the function $J_{A, B} : \mathbf{M}(A, B) \rightarrow \mathbf{Q}(A, B)$ is given by the interpretation functor J . A global element f of $\mathbf{C}(A, B)$ consists of a pair of functions f_0, f_1 that makes the following diagram commute.

$$\begin{array}{ccc} 1 & \xrightarrow{f_1} & \mathbf{M}(A, B) \\ & \searrow^{f_0} & \downarrow J_{A, B} \\ & & \mathbf{Q}(A, B) \end{array}$$

Thus, f_1 is a quantum circuit, which can be used as a quantum operation f_0 by composing with $J_{A, B}$. The biset-enriched category \mathbf{C} therefore maintains a distinction between \mathbf{M} and \mathbf{Q} while taking the interpretation functor J into account. To model the higher-order features of the programming language, we embed \mathbf{C} in a monoidal closed biset-enriched category $\tilde{\mathbf{C}}$, which we construct as a certain subcategory of the biset-enriched category of presheaves over \mathbf{C} . We show that $\tilde{\mathbf{C}}$ satisfies the axiomatization specified in [7]. Therefore it is a concrete model for Proto-Quipper with dynamic lifting.

Our approach to modeling dynamic lifting differs from recent work by Lee et al. [15], where the category of *quantum channels*, which generalize quantum circuits with a notion of branching for measurement results, is used to model a single runtime. Because our model accounts separately for circuit generation time (category \mathbf{M}) and circuit execution time (category \mathbf{Q}), we are able to support a type system that distinguishes quantum circuits from quantum computations that use dynamic lifting [7]. This prevents a class of runtime errors in Quipper caused by boxing a computation that uses dynamic lifting.

The rest of the paper is structured as follows. In Section 2, we first review some basic concepts from enriched category theory, and then recall from [7] the axiomatization of an enriched categorical semantics for dynamic lifting. In Section 3, we define the biset-enriched category \mathbf{C} . We show its presheaf category $\overline{\mathbf{C}}$ admits a commutative strong monad and a linear-non-linear adjunction. In Section 4, we construct a reflective subcategory $\tilde{\mathbf{C}}$ of $\overline{\mathbf{C}}$ and show that it is an enriched categorical model for dynamic lifting.

2 An enriched categorical semantics for dynamic lifting

Enriched categories are a generalization of categories where, instead of hom-sets, one works with hom-objects, which are objects in a monoidal category.

Definition 2.1. Let \mathcal{V} be a monoidal category. A \mathcal{V} -enriched category \mathbf{A} is given by the following:

- A class of objects, also denoted \mathbf{A} .
- For any $A, B \in \mathbf{A}$, an object $\mathbf{A}(A, B)$ in \mathcal{V} .
- For any $A \in \mathbf{A}$, a morphism $u_A : I \rightarrow \mathbf{A}(A, A)$ in \mathcal{V} , called the *identity* on A .
- For any $A, B, C \in \mathbf{A}$, a morphism $c_{A, B, C} : \mathbf{A}(A, B) \otimes \mathbf{A}(B, C) \rightarrow \mathbf{A}(A, C)$ in \mathcal{V} , called *composition*.
- The composition and identity morphisms must satisfy suitable diagrams in \mathcal{V} (see [2, 11]).

Remarks

- Many concepts from the theory of non-enriched categories can be generalized to the enriched setting. For example, \mathcal{V} -functors, \mathcal{V} -natural transformations, \mathcal{V} -adjunctions, and the \mathcal{V} -Yoneda embedding are all straightforward generalizations of their non-enriched counterparts. We refer the reader to [2, 11] for comprehensive introductions. Symmetric monoidal categories can also be generalized to the enriched setting (see Appendix A for a definition).

- In the rest of this paper, when we speak of a map $f : A \rightarrow B$ in a \mathcal{V} -enriched category \mathbf{A} , we mean a morphism of the form $f : I \rightarrow \mathbf{A}(A, B)$ in \mathcal{V} . Furthermore, when $g : B \rightarrow C$ is also a map in \mathbf{A} , we write $g \circ f : A \rightarrow C$ as a shorthand for

$$I \xrightarrow{f \otimes g} \mathbf{A}(A, B) \otimes \mathbf{A}(B, C) \xrightarrow{c} \mathbf{A}(A, C).$$

- A \mathcal{V} -enriched category \mathbf{A} gives rise to an ordinary (non-enriched) category $V(\mathbf{A})$, called the *underlying category* of \mathbf{A} .¹ The objects of $V(\mathbf{A})$ are the objects of \mathbf{A} and the hom-sets of $V(\mathbf{A})$ are defined as $V(\mathbf{A})(A, B) = \mathcal{V}(I, \mathbf{A}(A, B))$, for any $A, B \in V(\mathbf{A})$. Similarly, a \mathcal{V} -functor $F : \mathbf{A} \rightarrow \mathbf{B}$ gives rise to a functor $VF : V(\mathbf{A}) \rightarrow V(\mathbf{B})$ and a \mathcal{V} -natural transformation $\alpha : F \rightarrow G$ gives rise to a natural transformation $V\alpha : VF \rightarrow VG$.

The construction in this paper is parameterized by two small symmetric monoidal categories, denoted by \mathbf{M} and \mathbf{Q} . We fix \mathbf{M} and \mathbf{Q} once and for all and require the following:

- (1) \mathbf{M} and \mathbf{Q} have the same objects, including a distinguished object called **Bit**. The category \mathbf{M} has distinguished morphisms $\text{zero}, \text{one} : I \rightarrow \mathbf{Bit}$.
- (2) \mathbf{Q} has a coproduct $\mathbf{Bit} = I + I$, and the tensor product in \mathbf{Q} distributes over this coproduct.
- (3) There is a strict symmetric monoidal functor $J : \mathbf{M} \rightarrow \mathbf{Q}$ that is the identity on objects and $J(\text{zero}) = \text{inj}_1 : I \rightarrow I + I, J(\text{one}) = \text{inj}_2 : I \rightarrow I + I$. We call J the *interpretation functor*.
- (4) The category \mathbf{Q} is enriched in *convex spaces*. That is, for any real numbers $p_1, p_2 \in [0, 1]$ such that $p_1 + p_2 = 1$, and any maps $f, g \in \mathbf{Q}(A, B)$, there is a *convex sum* $p_1 f + p_2 g \in \mathbf{Q}(A, B)$, and the convex sum satisfies certain standard conditions which are detailed in Appendix C. Moreover, composition is *bilinear* with respect to convex sum, i.e., $(p_1 f_1 + p_2 f_2) \circ g = p_1 (f_1 \circ g) + p_2 (f_2 \circ g)$ and $h \circ (p_1 f_1 + p_2 f_2) = p_1 (h \circ f_1) + p_2 (h \circ f_2)$.
- (5) For any $A \in \mathbf{Q}$, and $f : I \rightarrow \mathbf{Bit} \otimes A \in \mathbf{Q}$, we have $f = p_1 (\text{inj}_1 \otimes f_1) + p_2 (\text{inj}_2 \otimes f_2)$, where $\text{inj}_1, \text{inj}_2 : I \rightarrow I + I$ and $p_1, p_2 \in [0, 1]$ are uniquely determined real numbers such that $p_1 + p_2 = 1$. When $p_i \neq 0$, the map $f_i : I \rightarrow A$ is also unique.

Perhaps it is useful to explain more specifically what we mean when we say that \mathbf{M} and \mathbf{Q} are fixed “once and for all”. The point is that these categories are not only used in the categorical semantics, but also in the operational semantics of Proto-Quipper-Dyn (i.e., to run the program, we must know what a circuit is and what a quantum operation is). Therefore, these categories should be regarded as given as part of the language specification, rather than as a degree of freedom in the semantics. On the other hand, nothing in the operational or denotational semantics depends on particular properties of \mathbf{M} and \mathbf{Q} other than properties (1)–(5) above. Therefore, Proto-Quipper-Dyn can handle a wide variety of possible circuit models and physical execution models.

In practice, the category \mathbf{M} will be a category of quantum circuits and the category \mathbf{Q} will be a category of quantum operations. These categories will typically have additional objects, such as **Qubit** and perhaps **Qutrit**, and additional morphisms, such as $H : \mathbf{Qubit} \rightarrow \mathbf{Qubit}$ and $\text{Meas} : \mathbf{Qubit} \rightarrow \mathbf{Bit}$. Requirement (5) is only needed in the operational semantics of Proto-Quipper-Dyn; it is not needed for the denotational semantics.

We now recall the enriched categorical semantics for dynamic lifting specified in [7].

¹We use $V(\mathbf{A})$ to denote the underlying category, rather than the usual $U(\mathbf{A})$, because the letter U will serve another purpose in this paper.

Definition 2.2. Let \mathcal{V} be a cartesian closed category with coproducts. A \mathcal{V} -category \mathbf{A} is a *model for Proto-Quipper with dynamic lifting* if it satisfies the following properties.

- a** \mathbf{A} is symmetric monoidal closed, i.e., it is symmetric monoidal and there is a \mathcal{V} -adjunction $- \otimes A \dashv A \multimap -$ for any $A \in \mathbf{A}$.
- b** \mathbf{A} has coproducts. Note that the tensor products distribute over coproducts, because $- \otimes A$ is a left adjoint functor for any $A \in \mathbf{A}$, which preserves coproducts.
- c** \mathbf{A} is equipped with a \mathcal{V} -adjunction $p : \mathcal{V} \rightarrow \mathbf{A} \dashv \flat : \mathbf{A} \rightarrow \mathcal{V}$ such that p is a strong monoidal \mathcal{V} -functor. This implies that $p(1) \cong I$ and $p(X \times Y) \cong pX \otimes pY$.
- d** \mathbf{A} is equipped with a commutative strong \mathcal{V} -monad T . For any $A, B \in \mathbf{A}$, we write $t_{A,B} : A \otimes TB \rightarrow T(A \otimes B)$ for the strength and $s_{A,B} : TA \otimes B \rightarrow T(A \otimes B)$ for the costrength.
- e** Let $V(\mathbf{A})$ be the underlying category of \mathbf{A} , VT be the underlying monad of T , and $Kl_{VT}(V(\mathbf{A}))$ be the Kleisli category of VT . The Kleisli category $Kl_{VT}(V(\mathbf{A}))$ is enriched in convex spaces. In other words, for any $A, B, C \in \mathbf{A}$, if $f, g : A \rightarrow TB$ and $p, q \in [0, 1], p + q = 1$, then there exists a convex sum $pf + qg : A \rightarrow TB$. Moreover, for any $h : C \rightarrow TA, e : B \rightarrow TC$, we have the following:

$$\mu \circ T(pf + qg) \circ h = p(\mu \circ Tf \circ h) + q(\mu \circ Tg \circ h),$$

$$\mu \circ Te \circ (pf + qg) = p(\mu \circ Te \circ f) + q(\mu \circ Te \circ g).$$

- f** There are fully faithful embeddings $\mathbf{M} \xrightarrow{\psi} V(\mathbf{A})$ and $\mathbf{Q} \xrightarrow{\phi} Kl_{VT}(V(\mathbf{A}))$. These embedding functors are strong monoidal, and ϕ preserves the convex sum. Moreover, the following diagram commutes for any $S, U \in \mathbf{M}$.

$$\begin{array}{ccc} \mathbf{M}(S, U) & \xrightarrow{\psi_{S,U}} & V(\mathbf{A})(S, U) \\ \downarrow J_{S,U} & & \downarrow E_{S,U} \\ \mathbf{Q}(S, U) & \xrightarrow{\phi_{S,U}} & Kl_{VT}(V(\mathbf{A}))(S, U) \end{array}$$

Here, $E : V(\mathbf{A}) \rightarrow Kl_{VT}(V(\mathbf{A}))$ is the functor such that $E(A) = A$ and $E(f) = \eta \circ f$.

- g** Let \mathcal{S} denote the set of objects in the image of ψ . For any $S, U \in \mathcal{S}$, there is an isomorphism

$$\flat(S \multimap U) \xrightarrow{e} \mathbf{A}(S, U).$$

- h** There are maps $\text{dynlift} : \mathbf{Bit} \rightarrow T\mathbf{Bool}$ and $\text{init} : \mathbf{Bool} \rightarrow \mathbf{Bit}$ in \mathbf{A} such that the following diagram commutes.

$$\begin{array}{ccc} & & \mathbf{Bit} \\ & \nearrow \text{init} & \downarrow \text{dynlift} \\ \mathbf{Bool} & \xrightarrow{\eta} & T\mathbf{Bool} \end{array}$$

Remarks

- Condition **c** gives rise to a comonoid structure $\text{dup}_X : pX \rightarrow pX \otimes pX$ and $\text{discard}_X : pX \rightarrow I$ for any $X \in \mathcal{V}$. Moreover, for any map $f : X \rightarrow Y$ in \mathcal{V} , we have the following in \mathbf{A} .

$$\text{dup}_Y \circ pf = (pf \otimes pf) \circ \text{dup}_X.$$

- Objects in the image of the functor p are called *parameter objects* in \mathbf{A} . Such objects are equipped with maps $\text{dup} : A \rightarrow A \otimes A$ and $\text{discard} : A \rightarrow I$. In particular, $\mathbf{Bool} := I + I = p(1) + p(1) = p(2)$ is a parameter object.
- Using condition **g**, we define $\text{box} = p(e)$ and $\text{unbox} = p(e^{-1})$, and we have

$$pb(S \multimap U) \stackrel{\text{box/unbox}}{\cong} p\mathbf{A}(S, U).$$

- Note that

$$Kl_{VT}(V(\mathbf{A}))(A, B) = V(\mathbf{A})(A, VTB) = \mathcal{V}(1, \mathbf{A}(A, TB)) = \mathcal{V}(1, Kl_T(\mathbf{A})(A, B)) = V(Kl_T(\mathbf{A}))(A, B).$$

- The Kleisli category $Kl_{VT}(V(\mathbf{A}))$ is monoidal because VT is a commutative strong monad and $V(\mathbf{A})$ is monoidal. For any $f : A_1 \rightarrow VTB_1$ and $g : A_2 \rightarrow VTB_2$ in $Kl_{VT}(V(\mathbf{A}))$, we define $f \otimes g \in Kl_{VT}(V(\mathbf{A}))(A_1 \otimes A_2, B_1 \otimes B_2)$ by

$$A_1 \otimes A_2 \xrightarrow{f \otimes g} VTB_1 \otimes VTB_2 \xrightarrow{s} VT(B_1 \otimes VTB_2) \xrightarrow{Tt} VTVT(B_1 \otimes B_2) \xrightarrow{\mu} VT(B_1 \otimes B_2).$$

- Since $\psi(S) = \phi(S)$ for any $S \in \mathbf{M}, \mathbf{Q}$, we define $\mathbf{Bit} = \psi(\mathbf{Bit}) = \phi(\mathbf{Bit}) \in \mathbf{A}$.
- Condition **f** expresses the requirement that the enriched category \mathbf{A} must combine both categories \mathbf{M} and \mathbf{Q} , i.e., they are subcategories of $V(\mathbf{A})$ and its Kleisli category, respectively. Thus \mathbf{A} has both quantum circuits and quantum operations. The commutative diagram implies that a circuit in \mathbf{A} can be used as a quantum operation.
- In [7], we have shown that conditions **a-h** are sufficient to give a model of Proto-Quipper-Dyn that is sound with respect to its type system and an operational semantics.

3 A biset-enriched category \mathbf{C} and its category of presheaves $\overline{\mathbf{C}}$

3.1 Biset enrichment

We now begin our construction of a concrete model satisfying Definition 2.2. Let $\mathbf{2}$ be the category with two objects $0, 1$ and one nontrivial arrow $0 \rightarrow 1$. Let $\mathcal{V} = \mathbf{Set}^{2^{\text{op}}}$ be the category of functors from 2^{op} to \mathbf{Set} . Concretely, the objects of \mathcal{V} are triples (A_0, A_1, f) , where A_0, A_1 are sets and f is a function $A_1 \rightarrow A_0$. We call such a triple a *biset*. A morphism in \mathcal{V} from (A_0, A_1, f) to (B_0, B_1, g) is a pair (h_0, h_1) , where $h_0 : A_0 \rightarrow B_0$ and $h_1 : A_1 \rightarrow B_1$ are functions such that the following diagram commutes.

$$\begin{array}{ccc} A_1 & \xrightarrow{h_1} & B_1 \\ \downarrow f & & \downarrow g \\ A_0 & \xrightarrow{h_0} & B_0 \end{array}$$

Because it is a presheaf category, the category of bisets $\mathcal{V} = \mathbf{Set}^{2^{\text{op}}}$ is complete, cocomplete, and cartesian closed. We write $A \Rightarrow B$ to denote an exponential object in \mathcal{V} .

The category \mathcal{V} is itself a \mathcal{V} -category where the hom-object $\mathcal{V}(A, B)$ is given by the exponential object $A \Rightarrow B$. We write $\text{Hom}_{\mathcal{V}}(A, B)$ to denote a hom-set when viewing \mathcal{V} as an ordinary category. Any set X can be viewed as a trivial biset (X, X, Id) . Therefore, any ordinary category can be viewed as a trivial biset-enriched category. For example, \mathbf{Set} can be viewed as a \mathcal{V} -category, where the hom-objects are given by $\mathbf{Set}(A, B) = (\mathbf{Set}(A, B), \mathbf{Set}(A, B), \text{Id})$ for any $A, B \in \mathbf{Set}$.

Definition 3.1. We define \mathcal{V} -functors $U_0(A_0, A_1, a) = A_0 : \mathcal{V} \rightarrow \mathbf{Set}$, and $\Delta(X) = (X, X, \text{Id}) : \mathbf{Set} \rightarrow \mathcal{V}$.

The \mathcal{V} -functor Δ is fully faithful and U_0 is strong monoidal. Note that there is also another functor $U_1(A_0, A_1, a) = A_1 : \mathcal{V} \rightarrow \mathbf{Set}$, but it is only an ordinary functor, not a \mathcal{V} -functor. This is because for $A, B \in \mathcal{V}$, there does not in general exist a morphism $A \Rightarrow B \rightarrow \mathbf{Set}(A_1, B_1)$ in \mathcal{V} . The functor U_1 will play no role in this paper, but the two \mathcal{V} -functors U_0 and Δ will be important.

Proposition 3.2. *There is a \mathcal{V} -adjunction $U_0 : \mathcal{V} \rightarrow \mathbf{Set} \dashv \Delta : \mathbf{Set} \rightarrow \mathcal{V}$. We write T for the \mathcal{V} -monad $\Delta \circ U_0$, it is a commutative strong \mathcal{V} -monad.*

3.2 The \mathcal{V} -category \mathbf{C}

In the following we define a non-trivial \mathcal{V} -category \mathbf{C} .

Definition 3.3. We define the \mathcal{V} -category \mathbf{C} as following.

- The objects of \mathbf{C} are the same as those of \mathbf{M} and \mathbf{Q} .
- For objects $A, B \in \mathbf{C}$, we define $\mathbf{C}(A, B)$ as the following object of \mathcal{V} ,

$$\mathbf{C}(A, B) = (\mathbf{Q}(A, B), \mathbf{M}(A, B), J_{AB} : \mathbf{M}(A, B) \rightarrow \mathbf{Q}(A, B)),$$

where $J : \mathbf{M} \rightarrow \mathbf{Q}$ is the interpretation functor.

- For every object $A \in \mathbf{C}$, we have a morphism $u_A = (\tilde{\text{Id}}_0, \tilde{\text{Id}}_1) : 1 \rightarrow \mathbf{C}(A, A)$ in \mathcal{V} , where $\tilde{\text{Id}}_1(*) = \text{Id}_A : A \rightarrow A$ in \mathbf{M} and $\tilde{\text{Id}}_0(*) = \text{Id}_A : A \rightarrow A$ in \mathbf{Q} .
- For any $A, B, C \in \mathbf{C}$, we have a morphism $c_{A,B,C} = (c_0, c_1) : \mathbf{C}(A, B) \times \mathbf{C}(B, C) \rightarrow \mathbf{C}(A, C)$ in \mathcal{V} , where $c_0 : \mathbf{Q}(A, B) \times \mathbf{Q}(B, C) \rightarrow \mathbf{Q}(A, C)$ and $c_1 : \mathbf{M}(A, B) \times \mathbf{M}(B, C) \rightarrow \mathbf{M}(A, C)$ are the compositions in \mathbf{Q} and \mathbf{M} , respectively.

3.3 The \mathcal{V} -category $\overline{\mathbf{C}}$

The biset-enriched category \mathbf{C} is symmetric monoidal. However, it is not closed. For that, we will need to work in the \mathcal{V} -enriched presheaf category $\overline{\mathbf{C}}$.

Definition 3.4. We define the \mathcal{V} -category $\overline{\mathbf{C}} = \mathcal{V}^{\mathbf{C}^{\text{op}}}$. Concretely, an object $F \in \overline{\mathbf{C}}$ is a \mathcal{V} -functor $\mathbf{C}^{\text{op}} \rightarrow \mathcal{V}$. Because \mathcal{V} is complete, for any $F, G \in \overline{\mathbf{C}}$, we have a hom-object $\overline{\mathbf{C}}(F, G) \in \mathcal{V}$ that represents \mathcal{V} -natural transformations $F \rightarrow G$.

An object in $\overline{\mathbf{C}}$ is a \mathcal{V} -functor $F : \mathbf{C}^{\text{op}} \rightarrow \mathcal{V}$. This means that for each $A \in \mathbf{C}^{\text{op}}$, there is an object $FA \in \mathcal{V}$. And for any $A, B \in \mathbf{C}^{\text{op}}$ there is a morphism $F_{AB} : \mathbf{C}^{\text{op}}(A, B) \rightarrow FA \Rightarrow FB$ in \mathcal{V} , which is the following commutative diagram.

$$\begin{array}{ccc} \mathbf{M}(B, A) & \xrightarrow{F_{AB}^1} & (FA \Rightarrow FB)_1 = \text{Hom}_{\mathcal{V}}(FA, FB) \\ \downarrow J_{B,A} & & \downarrow p_0 \\ \mathbf{Q}(B, A) & \xrightarrow{F_{AB}^0} & (FA \Rightarrow FB)_0 = \mathbf{Set}((FA)_0, (FB)_0). \end{array}$$

Note that an element $h \in \text{Hom}_{\mathcal{V}}(FA, FB)$ is a pair of function (h_0, h_1) such that the following commutes.

$$\begin{array}{ccc} (FA)_1 & \xrightarrow{h_1} & (FB)_1 \\ \downarrow f & & \downarrow g \\ (FA)_0 & \xrightarrow{h_0} & (FB)_0 \end{array}$$

Thus we define $p_0(h_0, h_1) = h_0$. So a \mathcal{V} -functor $F : \mathbf{C}^{\text{op}} \rightarrow \mathcal{V}$ induces an ordinary functor $F^0 : \mathbf{Q}^{\text{op}} \rightarrow \mathbf{Set}$, where $F^0(A) = (FA)_0$ and the function $\mathbf{Q}(B, A) \rightarrow \mathbf{Set}(F^0A, F^0B)$ is given by F_{AB}^0 for any $A, B \in \mathbf{Q}$.

Proposition 3.5. *The \mathcal{V} -category $\overline{\mathbf{C}}$ is a \mathcal{V} -monoidal closed category, where the tensor product \otimes_{Day} and linear exponential \multimap_{Day} are given by Day's convolution [3]. The tensor unit is defined by $I := yI = \mathbf{C}(-, I)$, where y is the \mathcal{V} -enriched Yoneda embedding functor.*

The \mathcal{V} -category $\overline{\mathbf{C}}$ has coproducts. Day's construction implies that the Day tensor product distributes over the coproducts, and that the \mathcal{V} -enriched Yoneda embedding $y : \mathbf{C} \hookrightarrow \overline{\mathbf{C}}$ is strong monoidal.

The \mathcal{V} -adjunction $U_0 \dashv \Delta$ and the \mathcal{V} -monad T can be lifted to $\overline{\mathbf{C}}$.

Definition 3.6. We define \mathcal{V} -functors $\overline{U}_0(F) := U_0 \circ F : \mathcal{V}^{\mathbf{C}^{\text{op}}} \rightarrow \mathbf{Set}^{\mathbf{C}^{\text{op}}}$, $\overline{\Delta}(F) := \Delta \circ F : \mathbf{Set}^{\mathbf{C}^{\text{op}}} \rightarrow \mathcal{V}^{\mathbf{C}^{\text{op}}}$, and $\overline{T} := \overline{\Delta} \circ \overline{U}_0 : \mathcal{V}^{\mathbf{C}^{\text{op}}} \rightarrow \mathcal{V}^{\mathbf{C}^{\text{op}}}$.

Note that $\overline{\Delta}$ is fully faithful and that \overline{U}_0 is strong monoidal.

Proposition 3.7. *There is a \mathcal{V} -adjunction $\overline{U}_0 : \mathcal{V}^{\mathbf{C}^{\text{op}}} \rightarrow \mathbf{Set}^{\mathbf{C}^{\text{op}}} \dashv \overline{\Delta} : \mathbf{Set}^{\mathbf{C}^{\text{op}}} \rightarrow \mathcal{V}^{\mathbf{C}^{\text{op}}}$.*

Proof. For any $F \in \mathbf{Set}^{\mathbf{C}^{\text{op}}}$, $G \in \overline{\mathbf{C}}$, we need to show that $\mathbf{Set}^{\mathbf{C}^{\text{op}}}(\overline{U}_0 F, G) \cong \mathcal{V}^{\mathbf{C}^{\text{op}}}(F, \overline{\Delta} G)$ that is \mathcal{V} -natural in F and G . This is true since the following isomorphisms follow from properties of *end*.

$$\begin{aligned} \mathcal{V}^{\mathbf{C}^{\text{op}}}(F, \overline{\Delta} G) &\cong \int_{A \in \mathbf{C}} \mathcal{V}(FA, \Delta GA) \cong \int_{A \in \mathbf{C}} \mathbf{Set}(U_0 FA, GA) \cong \int_{A \in \mathbf{C}} \mathcal{V}(\Delta U_0 FA, \Delta GA) \\ &\cong \mathcal{V}^{\mathbf{C}^{\text{op}}}(\overline{\Delta} \overline{U}_0 F, \overline{\Delta} G) \cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}(\overline{U}_0 F, G). \end{aligned} \quad \square$$

Proposition 3.8. *The monad \overline{T} is a commutative strong monad.*

Proposition 3.8 is a consequence of the following more general theorem, whose proof can be found in Appendix D.

Theorem 3.9. *Let \mathcal{V} be a complete, cocomplete, symmetric monoidal closed category. Let \mathbf{A} be a \mathcal{V} -category. If T is a commutative strong \mathcal{V} -monad on \mathcal{V} , then $\overline{T}(F) = T \circ F$ is a commutative strong \mathcal{V} -monad on $\mathcal{V}^{\mathbf{A}^{\text{op}}}$.*

Consider a \mathcal{V} -functor $F : \mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$. For any $A, B \in \mathbf{C}$, $FA \in \mathbf{Set}$, and the map $F_{AB} : \mathbf{C}(B, A) \rightarrow \mathbf{Set}(FA, FB)$ is uniquely determined by the function $F_{AB}^0 : \mathbf{Q}(B, A) \rightarrow \mathbf{Set}(FA, FB)$. So F is uniquely determined by $F^0 : \mathbf{Q}^{\text{op}} \rightarrow \mathbf{Set}$. In fact, the following theorem holds (the proof is in Appendix B).

Theorem 3.10. *We have $\mathbf{Set}^{\mathbf{C}^{\text{op}}} \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}$.*

The following proposition shows the maps in the Kleisli category of \overline{T} are essentially maps in $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$.

Proposition 3.11. *For any $F, G \in \overline{\mathbf{C}}$, we have*

$$\overline{\mathbf{C}}(F, \overline{T}G) = \overline{\mathbf{C}}(F, \overline{\Delta} \overline{U}_0 G) \cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}(\overline{U}_0 F, \overline{U}_0 G) \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, G^0).$$

3.4 A linear-non-linear adjunction in $\overline{\mathbf{C}}$

Suppose $F \in \overline{\mathbf{C}}$ and $V \in \mathcal{V}$. By definition, the *copower* $V \odot F$, if it exists, is an object $V \odot F \in \overline{\mathbf{C}}$ such that the isomorphism $\overline{\mathbf{C}}(V \odot F, G) \cong V \Rightarrow \overline{\mathbf{C}}(F, G)$ is \mathcal{V} -natural in $G \in \overline{\mathbf{C}}$.

Definition 3.12. Let $V \in \mathcal{V}$, $F \in \overline{\mathbf{C}}$. We define the copower $V \odot F$ in $\overline{\mathbf{C}}$ as follows:

$$(V \odot F)(A) = V \times FA : \mathbf{C}^{\text{op}} \rightarrow \mathcal{V}.$$

The fact that the above is indeed a copower can be verified using the calculus of *ends*. For any $F, G \in \overline{\mathbf{C}}$, we have

$$\begin{aligned} \overline{\mathbf{C}}(V \odot F, G) &\cong \int_{A \in \mathbf{C}} V \times FA \Rightarrow GA \cong \int_{A \in \mathbf{C}} V \Rightarrow (FA \Rightarrow GA) \\ &\cong V \Rightarrow \int_{A \in \mathbf{C}} (FA \Rightarrow GA) \cong V \Rightarrow \overline{\mathbf{C}}(F, G). \end{aligned}$$

Definition 3.13. We define \mathcal{V} -functors $p(X) = X \odot I : \mathcal{V} \rightarrow \overline{\mathbf{C}}$ and $\flat(F) = \overline{\mathbf{C}}(I, F) : \overline{\mathbf{C}} \rightarrow \mathcal{V}$.

The \mathcal{V} -functors p and \flat form a linear-non-linear adjunction in the sense of Benton [1].

Theorem 3.14. We have a \mathcal{V} -adjunction $p \dashv \flat$. Moreover, p is strong monoidal.

Proof. We have $\overline{\mathbf{C}}(pX, G) \cong \overline{\mathbf{C}}(X \odot I, G) \cong X \Rightarrow \overline{\mathbf{C}}(I, G) \cong X \Rightarrow \flat(G)$. Moreover, p is a strong monoidal \mathcal{V} -functor. We have $p(1) = 1 \odot yI \cong 1 \times \mathbf{C}(-, I) \cong yI$ and

$$\begin{aligned} p(X) \otimes_{\text{Day}} p(Y) &= \int^{A, B} \mathbf{C}(-, A \otimes B) \times X \times yI(A) \times Y \times yI(B) \\ &\cong X \times Y \times \int^{A, B} \mathbf{C}(-, A \otimes B) \times yI(A) \times yI(B) \cong X \times Y \times yI = p(X \times Y). \quad \square \end{aligned}$$

Theorem 3.15. For any $S, U \in \mathbf{C}$, there is an isomorphism $\flat(yS \multimap_{\text{Day}} yU) \cong \mathbf{C}(S, U)$.

Proof. We have $\flat(yS \multimap_{\text{Day}} yU) = \overline{\mathbf{C}}(I, yS \multimap_{\text{Day}} yU) \cong \overline{\mathbf{C}}(yS, yU) \cong \mathbf{C}(S, U)$. \square

Applying p to the above isomorphism yields $p\flat(yS \multimap_{\text{Day}} yU) \cong p\mathbf{C}(S, U)$. This isomorphism is called the *box/unbox* isomorphism in [17].

4 A reflective subcategory $\widetilde{\mathbf{C}}$ of $\overline{\mathbf{C}}$

The \mathcal{V} -category $\overline{\mathbf{C}}$ itself is not a model for Proto-Quipper with dynamic lifting. For example, it does not have a map $\mathbf{Bit} \rightarrow \overline{\mathbf{T}}\mathbf{Bool}$ for dynamic lifting. Namely, we define $\mathbf{Bool} := yI + yI = \mathbf{C}(-, I) + \mathbf{C}(-, I)$ and $\mathbf{Bit} := y\mathbf{Bit} = \mathbf{C}(-, \mathbf{Bit}) \in \overline{\mathbf{C}}$, where $\mathbf{Bit} \in \mathbf{C}$. Note that $\mathbf{Bit} = I + I$ in \mathbf{Q} . Consider the following

$$\begin{aligned} \overline{\mathbf{C}}(\mathbf{Bit}, \overline{\mathbf{T}}\mathbf{Bool}) &\cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}(\overline{U_0}\mathbf{Bit}, \overline{U_0}\mathbf{Bool}) \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(\mathbf{Bit}^0, \mathbf{Bool}^0) \\ &\cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(\mathbf{Q}(-, \mathbf{Bit}), \mathbf{Q}(-, I) + \mathbf{Q}(-, I)) = \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(\mathbf{Q}(-, I + I), \mathbf{Q}(-, I) + \mathbf{Q}(-, I)). \end{aligned}$$

So a map in $\overline{\mathbf{C}}(\mathbf{Bit}, \overline{\mathbf{T}}\mathbf{Bool})$ is the same as a natural transformation from $\mathbf{Q}(-, I + I)$ to $\mathbf{Q}(-, I) + \mathbf{Q}(-, I)$ in $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$. Moreover, for condition **h** to be satisfied, this natural transformation should be a left inverse of the canonical natural transformation $\mathbf{Q}(-, I) + \mathbf{Q}(-, I) \rightarrow \mathbf{Q}(-, I + I)$. On the other hand, by the Yoneda lemma, every natural transformation from $\mathbf{Q}(-, I + I)$ to $\mathbf{Q}(-, I) + \mathbf{Q}(-, I)$ either takes all of its values in the left component or in the right component of the disjoint union. Therefore, it can't be a left inverse to $\mathbf{Q}(-, I) + \mathbf{Q}(-, I) \rightarrow \mathbf{Q}(-, I + I)$. It follows that dynamic lifting cannot be interpreted in $\overline{\mathbf{C}}$. To fix this, we now consider a reflective subcategory of $\overline{\mathbf{C}}$, in the style of Lambek [14].

Definition 4.1. A \mathcal{V} -functor $F : \mathbf{C}^{\text{op}} \rightarrow \mathcal{V}$ is called *smooth* if $F^0 : \mathbf{Q}^{\text{op}} \rightarrow \mathbf{Set}$ is a product-preserving functor, i.e., $F^0(A + B) \cong F^0A \times F^0B$ for any $A, B \in \mathbf{Q}$.

Observe that for any $A \in \mathbf{C}$, the \mathcal{V} -enriched Yoneda embedding y of A , which is $\mathbf{C}(-, A)$, is smooth. Because $\mathbf{C}(-, A)^0 = \mathbf{Q}(-, A)$, and for any $B_1, B_2 \in \mathbf{Q}$, we have $\mathbf{Q}(B_1 + B_2, A) \cong \mathbf{Q}(B_1, A) \times \mathbf{Q}(B_2, A)$. Thus, the codomain of y consists of smooth \mathcal{V} -functors.

Definition 4.2. We define $\widetilde{\mathbf{C}}$ to be the full \mathcal{V} -subcategory of smooth functors.

Definition 4.3. We define the *Lambek embedding* $\overline{y} : \mathbf{C} \rightarrow \widetilde{\mathbf{C}}$ to be the corestriction of the Yoneda embedding y , i.e., it is the unique \mathcal{V} -functor such that the following diagram commutes.

$$\begin{array}{ccc} \mathbf{C} & & \\ \downarrow \overline{y} & \searrow y & \\ \widetilde{\mathbf{C}} & \xrightarrow{i} & \overline{\mathbf{C}} \end{array}$$

The details of the proof of the following theorem are in Appendix E.

Theorem 4.4. *The \mathcal{V} -category $\tilde{\mathbf{C}}$ is a reflective \mathcal{V} -subcategory of $\overline{\mathbf{C}}$, i.e., the inclusion \mathcal{V} -functor $i : \tilde{\mathbf{C}} \hookrightarrow \overline{\mathbf{C}}$ has a left adjoint \tilde{L} .*

Using results of Day [4, 5] (see also [16] for a more recent exposition), we can furthermore show that $\tilde{\mathbf{C}}$ is symmetric monoidal and \tilde{L} is strong monoidal. See Appendix F for further details. We now give an explicit definition of the monoidal closed structure in $\tilde{\mathbf{C}}$.

Definition 4.5. For any $F, G \in \tilde{\mathbf{C}}$, we define the tensor product, internal hom and tensor unit in $\tilde{\mathbf{C}}$ as $F \otimes_{\text{Lam}} G := \tilde{L}(iF \otimes_{\text{Day}} iG)$, $F \multimap_{\text{Lam}} G := iF \multimap_{\text{Day}} iG$, and $I := \overline{\gamma}I = \mathbf{C}(-, I)$, respectively.

In the above definition, the linear exponential $F \multimap_{\text{Lam}} G$ is well-defined because $iF \multimap_{\text{Day}} iG$ is an object in $\tilde{\mathbf{C}}$ (Theorem F.1).

Theorem 4.6. *The \mathcal{V} -category $\tilde{\mathbf{C}}$ is symmetric monoidal closed. For any $F, G, H \in \tilde{\mathbf{C}}$, there is a \mathcal{V} -natural isomorphism $\tilde{\mathbf{C}}(F \otimes_{\text{Lam}} G, H) \cong \tilde{\mathbf{C}}(F, G \multimap_{\text{Lam}} H)$.*

Proof. We have $\tilde{\mathbf{C}}(F \otimes_{\text{Lam}} G, H) \cong \tilde{\mathbf{C}}(\tilde{L}(iF \otimes_{\text{Day}} iG), H) \cong \overline{\mathbf{C}}(iF \otimes_{\text{Day}} iG, iH) \cong \overline{\mathbf{C}}(iF, iG \multimap_{\text{Day}} iH) \cong \tilde{\mathbf{C}}(F, G \multimap_{\text{Lam}} H)$. \square

4.1 A linear-non-linear adjunction in $\tilde{\mathbf{C}}$

The \mathcal{V} -category $\tilde{\mathbf{C}}$ also admits a linear-non-linear adjunction and, as in $\overline{\mathbf{C}}$, there is a box/unbox isomorphism in $\tilde{\mathbf{C}}$.

Definition 4.7. We define the \mathcal{V} -functors $\tilde{p}(X) = \tilde{L}(p(X)) : \mathcal{V} \rightarrow \tilde{\mathbf{C}}$ and $\tilde{b}(F) = b(iF) : \tilde{\mathbf{C}} \rightarrow \mathcal{V}$.

Theorem 4.8. *We have a \mathcal{V} -adjunction $\tilde{p} \dashv \tilde{b}$. Moreover, \tilde{p} is strong monoidal.*

Proof. We have $\tilde{\mathbf{C}}(\tilde{p}X, F) = \tilde{\mathbf{C}}(\tilde{L}(p(X)), F) \cong \overline{\mathbf{C}}(pX, iF) \cong X \Rightarrow b(iF) \cong X \Rightarrow \tilde{b}(F)$. Moreover, \tilde{p} is strong monoidal because both \tilde{L} and p are strong monoidal. \square

Theorem 4.9. *For any $S, U \in \mathbf{C}$, we have $\mathbf{C}(S, U) \cong \tilde{b}(\overline{\gamma}S \multimap_{\text{Lam}} \overline{\gamma}U)$.*

Proof. We have $\tilde{b}(\overline{\gamma}S \multimap_{\text{Lam}} \overline{\gamma}U) = b(i(\overline{\gamma}S \multimap_{\text{Lam}} \overline{\gamma}U)) = \overline{\mathbf{C}}(I, i(\overline{\gamma}S \multimap_{\text{Lam}} \overline{\gamma}U)) \cong \tilde{\mathbf{C}}(I, \overline{\gamma}S \multimap_{\text{Lam}} \overline{\gamma}U) \cong \tilde{\mathbf{C}}(\overline{\gamma}S, \overline{\gamma}U) \cong \mathbf{C}(S, U)$. \square

4.2 A commutative strong monad on $\tilde{\mathbf{C}}$

The \mathcal{V} -category $\tilde{\mathbf{C}}$ has a commutative strong monad. In the following we write $[\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$ for $\tilde{\mathbf{C}}$ and $\mathcal{V}^{\mathbf{C}^{\text{op}}}$ for $\overline{\mathbf{C}}$. We write $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ for the full subcategory of product-preserving functors of $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$. Consider the following diagram.

$$\begin{array}{ccc} \mathbf{Set}^{\mathbf{C}^{\text{op}}} & \xleftarrow{\overline{U}_0} & \mathcal{V}^{\mathbf{C}^{\text{op}}} \\ L \downarrow \uparrow j & \xleftarrow{\overline{\Delta}} & \downarrow \uparrow i \\ [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}} & \xleftarrow{\overline{U}'_0} & [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}} \end{array}$$

We define the \mathcal{V} -functor $\overline{U}'_0 : [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}} \rightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ by restricting the domain of \overline{U}_0 to $[\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$. Here $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ is the full \mathcal{V} -subcategory of smooth \mathcal{V} -functors. Similarly, $\overline{\Delta}' : [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}} \rightarrow [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$ is a restriction of $\overline{\Delta}$. We have a monoidal adjunction $L \dashv j$, since $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}} \cong [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$, the full subcategory of product-preserving functors, is reflective in $\mathbf{Set}^{\mathbf{Q}^{\text{op}}} \cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}$. We write $\tilde{T} = \overline{\Delta}' \circ \overline{U}'_0$. Observe that \tilde{T} is \overline{T} with a restricted domain.

Proposition 4.10. *By definition, we have $i \circ \bar{\Delta}' \cong \bar{\Delta} \circ j$ and $j \circ \bar{U}' \cong \bar{U}_0 \circ i$, therefore $i \circ \tilde{T} \cong \bar{T} \circ i$. Moreover, $\bar{U}' \circ \tilde{L} \cong L \circ \bar{U}_0$.*

Theorem 4.11. *We have a \mathcal{V} -adjunction $\bar{U}' \dashv \bar{\Delta}' : [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}} \rightarrow [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$. And \bar{U}' is strong monoidal.*

Proof. For any $X \in [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}, Y \in [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$, we have

$$\begin{aligned} [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}'X, Y) &\cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}(j\bar{U}'X, jY) \cong \mathbf{Set}^{\mathbf{C}^{\text{op}}}(\bar{U}_0iX, jY) \\ &\cong \mathcal{V}^{\mathbf{C}^{\text{op}}}(iX, \bar{\Delta}jY) \cong \mathcal{V}^{\mathbf{C}^{\text{op}}}(iX, i\bar{\Delta}'Y) \cong [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(X, \bar{\Delta}'Y). \end{aligned}$$

The \mathcal{V} -functor \bar{U}' is strong monoidal. For any $F, G \in [\mathcal{V}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$, we have $\bar{U}'I \cong \bar{U}_0I \cong I$ and

$$\begin{aligned} \bar{U}'(F \otimes_{\text{Lam}} G) &= \bar{U}'\tilde{L}(iF \otimes_{\text{Day}} iG) \cong L\bar{U}_0(iF \otimes_{\text{Day}} iG) \cong L(\bar{U}_0iF \otimes_{\text{Day}} \bar{U}_0iG) \\ &\cong L(j\bar{U}'F \otimes_{\text{Day}} j\bar{U}'G) = \bar{U}'F \otimes_{\text{Lam}} \bar{U}'G. \end{aligned} \quad \square$$

Theorem 4.12. *There is a \mathcal{V} -natural transformation $\rho : \tilde{L} \circ \bar{T} \rightarrow \tilde{T} \circ \tilde{L}$.*

Proof. For any $F \in \bar{\mathbf{C}}$, let $\eta_F : F \rightarrow i\tilde{L}F$ be the unit and $\varepsilon_F : \tilde{L}iF \rightarrow F$ be the counit (which is an isomorphism). We define ρ_F to be the composition $\tilde{L}\bar{T}F \xrightarrow{\tilde{L}\eta_F} \tilde{L}\bar{T}i\tilde{L}F \xrightarrow{\cong} \tilde{L}i\tilde{T}\tilde{L}F \xrightarrow{\varepsilon_{\tilde{T}F}} \tilde{T}\tilde{L}F$. \square

The natural transformation ρ is one of the components for defining the strength for \tilde{T} .

Theorem 4.13. *The \mathcal{V} -functor \tilde{T} is a commutative strong monad.*

Proof. For any $F, G \in \tilde{\mathbf{C}}$, the strength of \tilde{T} is given by

$$F \otimes_{\text{Lam}} \tilde{T}G = \tilde{L}(iF \otimes_{\text{Day}} i\tilde{T}G) \xrightarrow{\cong} \tilde{L}(iF \otimes_{\text{Day}} \bar{T}iG) \xrightarrow{\tilde{L}} \tilde{L}\bar{T}(iF \otimes_{\text{Day}} iG) \xrightarrow{\rho} \tilde{T}\tilde{L}(iF \otimes_{\text{Day}} iG) = \tilde{T}(F \otimes_{\text{Lam}} G).$$

Note that \bar{t} is the strength for \bar{T} . The verification of the strength diagrams is in Appendix G. \square

Similarly to Proposition 3.11, we have the following theorem for \tilde{T} .

Theorem 4.14. *For any $F, G \in \tilde{\mathbf{C}}$, we have the following \mathcal{V} -natural isomorphisms.*

$$\tilde{\mathbf{C}}(F, \tilde{T}G) \cong [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}'F, \bar{U}'G) \cong [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, G^0).$$

Proof. We have $\tilde{\mathbf{C}}(F, \tilde{T}G) = \tilde{\mathbf{C}}(F, \bar{\Delta}'\bar{U}'G) \cong [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}'F, \bar{U}'G) \cong [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, G^0)$. Note that by Theorem 3.10, $[\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}} \cong [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$. \square

4.3 Dynamic lifting in $\tilde{\mathbf{C}}$

Since $\bar{\mathbf{C}}$ has coproducts and $\tilde{\mathbf{C}}$ is a reflective subcategory, the coproduct of $A, B \in \tilde{\mathbf{C}}$ is defined as $A +' B = \tilde{L}(iA + iB)$. In $\tilde{\mathbf{C}}$, we define $\mathbf{Bool} := \bar{y}I +' \bar{y}I = \tilde{L}(\bar{y}I + \bar{y}I)$ and $\mathbf{Bit} := \bar{y}(\mathbf{Bit})$, where $I, \mathbf{Bit} \in \mathbf{C}$. There exists maps zero, one : $\bar{y}I \rightarrow \mathbf{Bit}$ in $\tilde{\mathbf{C}}$. We are now ready to define a map for dynamic lifting.

Theorem 4.15. *There are \mathcal{V} -natural transformations $\text{init} : \mathbf{Bool} \rightarrow \mathbf{Bit}$ and $\text{dynlift} : \mathbf{Bit} \rightarrow \tilde{T}\mathbf{Bool}$ in $\tilde{\mathbf{C}}$ such that the following diagram commutes.*

$$\begin{array}{ccc} & & \mathbf{Bit} \\ & \nearrow \text{init} & \downarrow \text{dynlift} \\ \mathbf{Bool} & \xrightarrow{\eta} & \tilde{T}\mathbf{Bool} \end{array}$$

Proof. We define $\text{init} = [\text{zero}, \text{one}] : \mathbf{Bool} \rightarrow \mathbf{Bit}$. Firstly, we want to show that $\tilde{T}\text{init} : \tilde{T}\mathbf{Bool} \rightarrow \tilde{T}\mathbf{Bit}$ is an isomorphism. Using Yoneda’s principle, we just need to show $\tilde{\mathbf{C}}(F, \tilde{T}\text{init}) : \tilde{\mathbf{C}}(F, \tilde{T}\mathbf{Bool}) \rightarrow \tilde{\mathbf{C}}(F, \tilde{T}\mathbf{Bit})$ is an isomorphism for any $F \in \tilde{\mathbf{C}}$. By Theorem 4.14, this is equivalent to showing that

$$[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, \text{init}^0) : [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, \mathbf{Bool}^0) \rightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, \mathbf{Bit}^0)$$

is an isomorphism. This is the case because the Lambek embedding $\kappa : \mathbf{Q} \hookrightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ preserves coproducts, $\text{Bit} = I + I \in \mathbf{Q}$, and the map $\text{init}^0 : \kappa I + \kappa I \rightarrow \kappa(I + I)$ is an isomorphism in $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$. We therefore define dynlift as the composition $(\tilde{T}\text{init})^{-1} \circ \eta : \mathbf{Bit} \rightarrow \tilde{T}\mathbf{Bit} \rightarrow \tilde{T}\mathbf{Bool}$. As a result, we have the following commutative diagram.

$$\begin{array}{ccc}
 \mathbf{Bool} & \xrightarrow{\eta} & \tilde{T}\mathbf{Bool} \\
 \downarrow \text{init} & & \downarrow \tilde{T}\text{init} \searrow \text{Id} \\
 \mathbf{Bit} & \xrightarrow{\eta} & \tilde{T}\mathbf{Bit} \xrightarrow{(\tilde{T}\text{init})^{-1}} \tilde{T}\mathbf{Bool}
 \end{array}
 \quad \square$$

4.4 $\tilde{\mathbf{C}}$ is a model for Proto-Quipper with dynamic lifting

Recall that the category \mathbf{Q} is enriched in convex spaces, i.e., the hom-sets of \mathbf{Q} are convex spaces and the composition is bilinear with respect to the convex sum. We have the following theorem, whose proof is in Appendix C.

Theorem 4.16. *The category $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ is enriched in convex spaces. Moreover, the Lambek embedding $\kappa : \mathbf{Q} \hookrightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ preserves the convex sum in \mathbf{Q} .*

The above theorem implies that for any $A, B \in \tilde{\mathbf{C}}$, the Kleisli-hom $\tilde{\mathbf{C}}(A, \tilde{T}B)$ is convex because of the isomorphism $\tilde{\mathbf{C}}(A, \tilde{T}B) \cong [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(A^0, B^0)$ from Theorem 4.14. We are now ready to state our main theorem (see Appendix H for the proof).

Theorem 4.17. *The \mathcal{V} -category $\tilde{\mathbf{C}}$ is a model for Proto-Quipper with dynamic lifting, i.e., it satisfies conditions **a–h** in Definition 2.2.*

5 Conclusion

We constructed a categorical model for dynamic lifting using biset enrichment. We defined a biset-enriched category \mathbf{C} , which combines the categories \mathbf{M} and \mathbf{Q} . We then considered the full subcategory $\tilde{\mathbf{C}}$ of smooth functors and showed that $\tilde{\mathbf{C}}$ is a reflective subcategory in the enriched presheaf category of \mathbf{C} . Finally, we proved that $\tilde{\mathbf{C}}$ is categorical model for dynamic lifting in the sense of [7].

Acknowledgements

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

References

- [1] Nick Benton (1995): *A mixed linear and non-linear logic: proofs, terms and models (extended abstract)*. In: *Proceedings of the 8th Workshop on Computer Science Logic, CSL’94, Selected Papers*, Springer Lecture Notes in Computer Science 933, pp. 121–135, doi:10.1007/BFb0022251.

- [2] Francis Borceux (1994): *Handbook of Categorical Algebra, Volume 2: Categories and Structures*. Cambridge University Press, doi:10.1017/CB09780511525865.
- [3] Brian Day (1970): *On closed categories of functors*. In: *Reports of the Midwest Category Seminar IV, Springer Lecture Notes in Mathematics* 137, pp. 1–38, doi:10.1007/BFb0060438.
- [4] Brian Day (1972): *A reflection theorem for closed categories*. *Journal of Pure and Applied Algebra* 2(1), pp. 1–11, doi:10.1016/0022-4049(72)90021-7.
- [5] Brian Day (1973): *Note on monoidal localisation*. *Bulletin of the Australian Mathematical Society* 8(1), pp. 1–16, doi:10.1017/S0004972700045433.
- [6] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2020): *A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper*. In: *Proceedings of the 12th International Conference on Reversible Computation, RC 2020, Oslo, Norway, Lecture Notes in Computer Science* 12227, Springer, pp. 153–168, doi:10.1007/978-3-030-52482-1_9. Also available from arXiv:2005.08396.
- [7] Peng Fu, Kohei Kishida, Neil J. Ross & Peter Selinger (2023): *Proto-Quipper with Dynamic Lifting*. *Proc. ACM Program. Lang.* 7 (POPL 2023), doi:10.1145/3571204. Also available from arXiv:2204.13041.
- [8] Peng Fu, Kohei Kishida & Peter Selinger (2020): *Linear dependent type theory for quantum programming languages*. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020, Saarbrücken, Germany*, pp. 440–453, doi:10.1145/3373718.3394765. Also available from arXiv:2004.13472.
- [9] Alexander Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *An introduction to quantum programming in Quipper*. In: *Proceedings of the 5th International Conference on Reversible Computation, RC 2013, Victoria, British Columbia, Lecture Notes in Computer Science* 7948, Springer, pp. 110–124, doi:10.1007/978-3-642-38986-3_10. Also available from arXiv:1304.5485.
- [10] Alexander Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: a scalable quantum programming language*. In: *Proceedings of the 34th Annual ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2013, Seattle, ACM SIGPLAN Notices* 48(6), pp. 333–342, doi:10.1145/2499370.2462177. Also available from arXiv:1304.3390.
- [11] G. M. Kelly (1982): *Basic concepts of enriched category theory*. *Lecture Notes of the London Mathematical Society* 64, Cambridge University Press.
- [12] Emmanuel Knill (1996): *Conventions for quantum pseudocode*. Technical Report, Los Alamos National Laboratory, doi:10.48550/arXiv.2211.02559.
- [13] Anders Kock (1972): *Strong functors and monoidal monads*. *Archiv der Mathematik* 23(1), pp. 113–120, doi:10.1007/BF01304852.
- [14] Joachim Lambek (1966): *Completions of categories: Seminar lectures given 1966 in Zürich*. *Lecture Notes in Mathematics* 24, Springer, doi:10.1007/BFb0077265.
- [15] Dongho Lee, Valentin Perrelle, Benoît Valiron & Zhaowei Xu (2021): *Concrete categorical model of a quantum circuit description language with measurement*. In Mikolaj Bojanczyk & Chandra Chekuri, editors: *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, LIPIcs* 213, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 51:1–51:20, doi:10.4230/LIPIcs.FSTTCS.2021.51.
- [16] Octavio Malherbe (2013): *Categorical Models of Computation: Partially Traced Categories and Presheaf Models of Quantum Computation*. Ph.D. thesis, University of Ottawa, Department of Mathematics and Statistics. Available from arXiv:1301.5087.
- [17] Francisco Rios & Peter Selinger (2018): *A categorical model for a quantum circuit description language. Extended Abstract*. In: *Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen, Electronic Proceedings in Theoretical Computer Science* 266, pp. 164–178, doi:10.4204/EPTCS.266.11.
- [18] Neil J. Ross (2015): *Algebraic and Logical Methods in Quantum Computation*. Ph.D. thesis, Dalhousie University, Department of Mathematics and Statistics. Available from arXiv:1510.02198.

A Enriched symmetric monoidal categories

Definition A.1. Let \mathcal{V} be a symmetric monoidal category. A \mathcal{V} -category \mathbf{A} is symmetric monoidal if it is equipped with the following:

- There is an object I , called the *tensor unit*. For all $A, B \in \mathbf{A}$, there is an object $A \otimes B \in \mathbf{A}$. Moreover, for all $A_1, A_2, B_1, B_2 \in \mathbf{A}$, there is a map

$$\text{Tensor} : \mathbf{A}(A_1, B_1) \otimes \mathbf{A}(A_2, B_2) \rightarrow \mathbf{A}(A_1 \otimes A_2, B_1 \otimes B_2)$$

in \mathcal{V} . The tensor product is a bifunctor in the sense that $\text{Tensor} \circ (u_A \otimes u_B) = u_{A \otimes B}$ for the identity maps $u_A, u_B, u_{A \otimes B}$, and the following diagram commutes for any $A_1, A_2, B_1, B_2, C_1, C_2 \in \mathbf{A}$.

$$\begin{array}{ccc} \mathbf{A}(A_1, B_1) \otimes \mathbf{A}(A_2, B_2) \otimes \mathbf{A}(B_1, C_1) \otimes \mathbf{A}(B_2, C_2) & \xrightarrow{c \otimes c} & \mathbf{A}(A_1, C_1) \otimes \mathbf{A}(A_2, C_2) \\ \downarrow \text{Tensor} \otimes \text{Tensor} & & \downarrow \text{Tensor} \\ \mathbf{A}(A_1 \otimes A_2, B_1 \otimes B_2) \otimes \mathbf{A}(B_1 \otimes B_2, C_1 \otimes C_2) & \xrightarrow{c} & \mathbf{A}(A_1 \otimes A_2, C_1 \otimes C_2) \end{array}$$

- There are the following \mathcal{V} -natural isomorphisms in \mathbf{A} and they satisfy the same coherence diagrams as for symmetric monoidal categories, and analogous naturality conditions.

$$l_A : I \otimes A \rightarrow A$$

$$r_A : A \otimes I \rightarrow A$$

$$\gamma_{A,B} : A \otimes B \rightarrow B \otimes A$$

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C)$$

If the \mathcal{V} -category \mathbf{A} is symmetric monoidal, for all maps $f : A_1 \rightarrow B_1, g : A_2 \rightarrow B_2$ in \mathbf{A} , we write $f \otimes g : A_1 \otimes A_2 \rightarrow B_1 \otimes B_2$ as a shorthand for the following composition.

$$I \xrightarrow{f \otimes g} \mathbf{A}(A_1, B_1) \otimes \mathbf{A}(A_2, B_2) \xrightarrow{\text{Tensor}} \mathbf{A}(A_1 \otimes A_2, B_1 \otimes B_2)$$

B Biset-enriched functor categories

Notations. Let \mathbf{A}, \mathbf{B} be \mathcal{V} -categories. For all $A, B \in \mathbf{A}$, we have

$$\mathbf{A}(A, B) = (\mathbf{A}(A, B)_0, \mathbf{A}(A, B)_1, \varphi^A : \mathbf{A}(A, B)_1 \rightarrow \mathbf{A}(A, B)_0).$$

So we write $A \rightarrow_1 B := \mathbf{A}(A, B)_1$ and $A \rightarrow_0 B := \mathbf{A}(A, B)_0$. Moreover, for all $f : A \rightarrow_1 B$, we have $\varphi^A(f) : A \rightarrow_0 B$. A \mathcal{V} -functor $F : \mathbf{A} \rightarrow \mathbf{B}$ gives rise to the following commutative diagram for all $A, B \in \mathbf{A}$.

$$\begin{array}{ccc} \mathbf{A}(A, B)_1 & \xrightarrow{F_{A,B}^1} & \mathbf{B}(FA, FB)_1 \\ \downarrow \varphi^A & & \downarrow \varphi^B \\ \mathbf{A}(A, B)_0 & \xrightarrow{F_{A,B}^0} & \mathbf{B}(FA, FB)_0 \end{array}$$

For all $f : A \rightarrow_1 B$, we have $F_{A,B}^1 f : FA \rightarrow_1 FB$. Similarly, for all $g : A \rightarrow_0 B$, we have $F_{A,B}^0 g : FA \rightarrow_0 FB$.

For any \mathcal{V} -functors $F, G : \mathbf{A} \rightarrow \mathbf{B}$, we define a biset $(F \Rightarrow_0 G, F \Rightarrow_1 G, p : F \Rightarrow_1 G \rightarrow F \Rightarrow_0 G)$ as follows.

$$F \Rightarrow_0 G := \{(\beta_A : FA \rightarrow_0 GA)_{A \in \mathbf{A}} \mid \forall A, B \in \mathbf{A}, \forall g : A \rightarrow_0 B, \beta_B \circ F_{AB}^0 g = G_{AB}^0 \circ \beta_A\}$$

$$F \Rightarrow_1 G := \{(\alpha_A : FA \rightarrow_1 GA)_{A \in \mathbf{A}} \mid \forall A, B \in \mathbf{A}, \forall f : A \rightarrow_1 B, \alpha_B \circ F_{AB}^1 f = G_{AB}^1 \circ \alpha_A, \\ \forall A, B \in \mathbf{A}, \forall g : A \rightarrow_0 B, \varphi^{\mathbf{B}}(\alpha_B) \circ F_{AB}^0 g = G_{AB}^0 \circ \varphi^{\mathbf{B}}(\alpha_A)\}$$

$$p((\alpha_A : FA \rightarrow_1 GA)_{A \in \mathbf{A}}) := (\varphi^{\mathbf{B}}(\alpha_A) : FA \rightarrow_0 GA)_{A \in \mathbf{A}} : F \Rightarrow_1 G \rightarrow F \Rightarrow_0 G$$

Proposition B.1. *Suppose \mathbf{A}, \mathbf{B} are \mathcal{V} -categories. Since the category of bisets \mathcal{V} is complete, the functor category $\mathbf{B}^{\mathbf{A}}$ is \mathcal{V} -enriched. For all \mathcal{V} -functors $F, G : \mathbf{A} \rightarrow \mathbf{B}$, we have*

$$\mathbf{B}^{\mathbf{A}}(F, G) := \int_{A \in \mathbf{A}} \mathbf{B}(FA, GA) \cong (F \Rightarrow_0 G, F \Rightarrow_1 G, p : F \Rightarrow_1 G \rightarrow F \Rightarrow_0 G)$$

Proof. By definition of end, we have the following equalizer diagram in \mathcal{V} .

$$\int_{A \in \mathbf{A}} \mathbf{B}(FA, GA) := eq(u, v) \xrightarrow{k} \prod_{A \in \mathbf{A}} \mathbf{B}(FA, GA) \xrightarrow[u]{u} \prod_{A, B \in \mathbf{A}} \mathbf{A}(A, B) \Rightarrow \mathbf{B}(FA, GB)$$

Note that $u = \langle \text{curry}(c \circ (\pi_A \times G_{AB})) \rangle_{A, B \in \mathbf{A}}$, where $c \circ (\pi_A \times G_{AB})$ is the following.

$$(\prod_A \mathbf{B}(FA, GA)) \times \mathbf{A}(A, B) \xrightarrow{\pi_A \times G_{AB}} \mathbf{B}(FA, GA) \times \mathbf{B}(GA, GB) \xrightarrow{c} \mathbf{B}(FA, GB)$$

We have $v = \langle \text{curry}(c \circ (\pi_B \times F_{AB})) \rangle_{A, B \in \mathbf{A}}$, where $c \circ (\pi_B \times F_{AB})$ is the following.

$$(\prod_A \mathbf{B}(FA, GA)) \times \mathbf{A}(A, B) \xrightarrow{\pi_B \times F_{AB}} \mathbf{B}(FB, GB) \times \mathbf{B}(FA, FB) \xrightarrow{c} \mathbf{B}(FA, GB)$$

We can show $(\int_{A \in \mathbf{A}} \mathbf{B}(FA, GA))_1 = eq(u_1, v_1) \cong F \Rightarrow_1 G$ and $(\int_{A \in \mathbf{A}} \mathbf{B}(FA, GA))_0 = eq(u_0, v_0) \cong F \Rightarrow_0 G$. \square

Theorem B.2. *The biset-enriched categories $\mathbf{Set}^{\mathbf{C}^{\text{op}}}$ and $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$ are isomorphic.*

Proof. Let us define a \mathcal{V} -enriched functor $\Omega : \mathbf{Set}^{\mathbf{C}^{\text{op}}} \rightarrow \mathbf{Set}^{\mathbf{Q}^{\text{op}}}$. On objects, $\Omega(F) = F^0$ for any $F \in \mathbf{Set}^{\mathbf{C}^{\text{op}}}$. Since $F : \mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$ is uniquely determined by F^0 , the function Ω is bijective on objects.

Suppose $F, G : \mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$. We claim that $\mathbf{Set}^{\mathbf{C}^{\text{op}}}(F, G) \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, G^0)$. This will allow us to define $\Omega_{F, G}$ to be this isomorphism. To show $\mathbf{Set}^{\mathbf{C}^{\text{op}}}(F, G) \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, G^0)$, first of all, we have

$$\mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, G^0) = (X, X, \text{Id}),$$

where

$$X = \{(\alpha_A : F^0 A \rightarrow G^0 A)_{A \in \mathbf{Q}} \mid \forall A, B \in \mathbf{Q}, \forall f : A \rightarrow B \in \mathbf{Q}, \alpha_B \circ F_{AB}^0 f = G_{AB}^0 \circ \alpha_A\}.$$

Next,

$$\mathbf{Set}^{\mathbf{C}^{\text{op}}}(F, G) = (F \Rightarrow_1 G, F \Rightarrow_0 G, p),$$

where

$$F \Rightarrow_0 G = \{(\alpha_A : FA \rightarrow_0 GA)_{A \in \mathbf{C}} \mid \forall A, B \in \mathbf{C}, \forall f : A \rightarrow_0 B \in \mathbf{C}, \alpha_B \circ F_{AB}^0 f = G_{AB}^0 f \circ \alpha_A\} \cong X$$

and

$$F \Rightarrow_1 G := \{(\alpha_A : FA \rightarrow_1 GA)_{A \in \mathbf{A}} \mid \forall A, B \in \mathbf{C}, \forall f : A \rightarrow_1 B, \alpha_B \circ F_{AB}^1 f = G_{AB}^1 f \circ \alpha_A, \\ \forall A, B \in \mathbf{C}, \forall g : A \rightarrow_0 B, \varphi^{\text{Set}}(\alpha_B) \circ F_{AB}^0 g = G_{AB}^0 g \circ \varphi^{\text{Set}}(\alpha_A)\}.$$

Since $F_{A,B}^1 = F_{A,B}^0 \circ \varphi^{\text{C}^{\text{op}}}$, and $\varphi^{\text{Set}} = \text{Id}$, and $\varphi^{\text{C}^{\text{op}}}(f) : A \rightarrow_0 B$ for any $f : A \rightarrow_1 B$ with $A, B \in \mathbf{C}$, therefore $\forall A, B \in \mathbf{C}, \forall g : A \rightarrow_0 B, \varphi^{\text{Set}}(\alpha_B) \circ F_{AB}^0 g = G_{AB}^0 g \circ \varphi^{\text{Set}}(\alpha_A)$ implies $\forall A, B \in \mathbf{C}, \forall f : A \rightarrow_1 B, \alpha_B \circ F_{AB}^1 f = G_{AB}^1 f \circ \alpha_A$. So $F \Rightarrow_1 G \cong F \Rightarrow_0 G \cong X$ and $p = \text{Id}$. \square

C Convexity

Let $[0, 1]$ denote the real unit interval.

Definition C.1. A *convexity structure* on a set X is an operation that assigns to all $p, q \in [0, 1]$ with $p + q = 1$ and all $x, y \in X$ an element $px + qy \in X$, subject to the following properties. Throughout, we assume $p + q = 1$.

- (a) $px + qx = x$ for all $x \in X$.
- (b) $px + qy = qy + px$ for all $x, y \in X$.
- (c) $0x + 1y = y$ for all $x, y \in X$.
- (d) $(a + b)(\frac{a}{a+b}x + \frac{b}{a+b}y) + (c + d)(\frac{c}{c+d}z + \frac{d}{c+d}w) = (a + c)(\frac{a}{a+c}x + \frac{c}{a+c}z) + (b + d)(\frac{b}{b+d}y + \frac{d}{b+d}w)$, where $a, b, c, d \in [0, 1]$ with $a + b + c + d = 1$ and all denominators are non-zero.

Remark. Property (d) can best be understood by realizing that both sides of the equation are equal to $ax + by + cz + dw$, decomposed in two different ways into convex sums of two elements at a time. In the literature, we sometimes find a different, but equivalent condition of the form $s(px + qy) + rz = spx + (qs + r)(\frac{qs}{qs+r}y + \frac{r}{qs+r}z)$. The latter axiom is arguably shorter, but harder to read.

We often expand the binary $+$ operation to a multi-arity operation, i.e., $\sum_i p_i x_i$, where $\sum_i p_i = 1$ and $x_i \in X$ for all i .

We say that a category \mathbf{A} is *enriched in convex spaces* if for all $A, B \in \mathbf{A}$, the hom-set $\mathbf{A}(A, B)$ is convex, and composition is bilinear, i.e., for all $f, g \in \mathbf{A}(A, B), e \in \mathbf{A}(C, A), h \in \mathbf{A}(B, C)$ and $p, q \in [0, 1]$ with $p + q = 1$, we have

$$(pf + qg) \circ e = pf \circ e + qg \circ e$$

and

$$h \circ (pf + qg) = ph \circ f + qh \circ g.$$

Theorem C.2. Let \mathbf{A} be a symmetric monoidal category with a coproduct $I + I$, such that tensor distributes over this coproduct. The following are equivalent.

1. The category \mathbf{A} is enriched in convex spaces.

2. There exists a family of maps $\langle\langle p, q \rangle\rangle : I \rightarrow I + I$, where $p, q \in [0, 1]$ with $p + q = 1$, such that the following diagrams commute:

$$\begin{array}{c}
 \begin{array}{ccc}
 I & \xrightarrow{\langle\langle p, q \rangle\rangle} & I + I & \xrightarrow{[\text{Id}, \text{Id}]} & I \\
 & \searrow & \text{Id} & \swarrow & \\
 & & I & &
 \end{array} & &
 \begin{array}{ccc}
 I & \xrightarrow{\langle\langle p, q \rangle\rangle} & I + I \\
 & \searrow & \downarrow [\text{inj}_2, \text{inj}_1] \\
 & & I + I
 \end{array} & &
 \begin{array}{ccc}
 I & \xrightarrow{\text{inj}_2} & I + I \\
 & \searrow & \downarrow \langle\langle 0, 1 \rangle\rangle \\
 & & I + I
 \end{array} \\
 \\
 \begin{array}{ccc}
 & & I & & \\
 & \swarrow & & \searrow & \\
 & I + I & & I + I & \\
 & \downarrow \langle\langle \frac{a}{a+b}, \frac{b}{a+b} \rangle\rangle + \langle\langle \frac{c}{c+d}, \frac{d}{c+d} \rangle\rangle & & \downarrow \langle\langle \frac{a}{a+c}, \frac{c}{a+c} \rangle\rangle + \langle\langle \frac{b}{b+d}, \frac{d}{b+d} \rangle\rangle & \\
 & (I + I) + (I + I) & \xrightarrow{\text{iso}} & (I + I) + (I + I) &
 \end{array}
 \end{array}$$

Here, in the last diagram, we have $a, b, c, d \in [0, 1]$ with $a + b + c + d = 1$, and we assume the denominators are non-zero. The map “iso” is the canonical isomorphism $(A + B) + (C + D) \cong (A + C) + (B + D)$.

Proof. For the left-to-right implication, suppose \mathbf{A} is enriched in convex spaces. We can define

$$\langle\langle p, q \rangle\rangle := p \text{inj}_1 + q \text{inj}_2 : I \rightarrow I + I.$$

It is easy to verify that this definition of $\langle\langle p, q \rangle\rangle$ satisfies the four diagrams above.

We now focus on the right-to-left implication.

- First we need to show that $\mathbf{A}(A, B)$ is convex for all $A, B \in \mathbf{A}$. Given $f, g \in \mathbf{A}(A, B)$, we define $pf + qg$ as follows.

$$A \xrightarrow{\lambda^{-1}} A \otimes I \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} A \otimes (I + I) \xrightarrow{d} A \otimes I + A \otimes I \xrightarrow{\lambda + \lambda} A + A \xrightarrow{[f, g]} B.$$

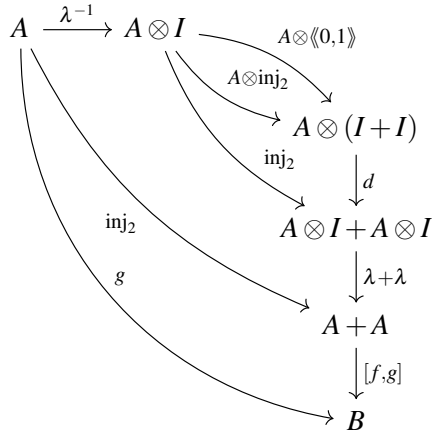
- $pf + qf = f$. This holds because the following diagram commutes.

$$\begin{array}{ccccccc}
 A & \xrightarrow{\lambda^{-1}} & A \otimes I & \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} & A \otimes (I + I) & \xrightarrow{d} & A \otimes I + A \otimes I & \xrightarrow{\lambda + \lambda} & A + A & \xrightarrow{[f, f]} & B \\
 \downarrow f & & \downarrow f \otimes I & & \downarrow f \otimes (I + I) & & \downarrow f \otimes I + f \otimes I & & f + f & \downarrow [\text{Id}, \text{Id}] & \nearrow \\
 B & \xrightarrow{\lambda^{-1}} & B \otimes I & \xrightarrow{B \otimes \langle\langle p, q \rangle\rangle} & B \otimes (I + I) & \xrightarrow{d} & B \otimes I + B \otimes I & \xrightarrow{\lambda + \lambda} & B + B & & \\
 & & \searrow \text{Id} & & \downarrow B \otimes [\text{Id}, \text{Id}] & & \swarrow [\text{Id}, \text{Id}] & & & & \nearrow \lambda \\
 & & & & B \otimes I & & & & & &
 \end{array}$$

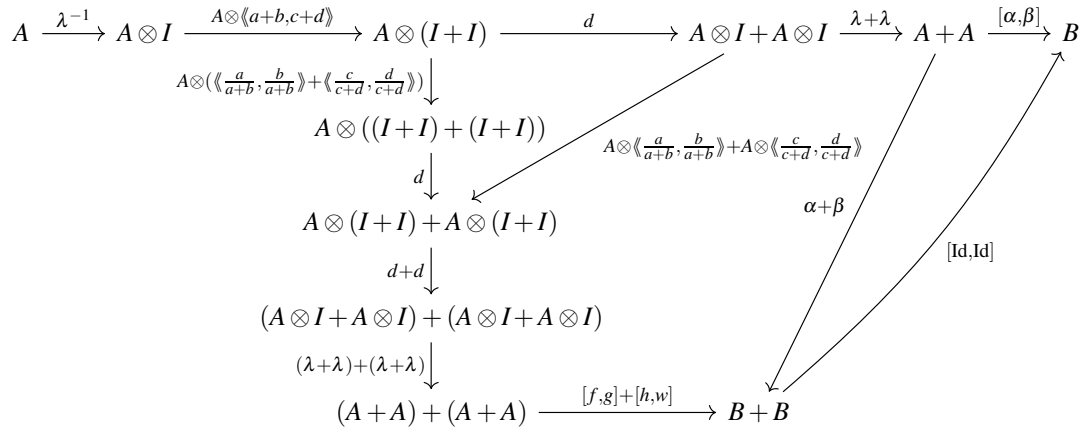
- $pf + qg = qg + pf$. This holds because the following diagram commutes.

$$\begin{array}{ccccccc}
 A & \xrightarrow{\lambda^{-1}} & A \otimes I & \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} & A \otimes (I + I) & \xrightarrow{d} & A \otimes I + A \otimes I & \xrightarrow{\lambda + \lambda} & A + A & \xrightarrow{[f, g]} & B \\
 & & \searrow A \otimes \langle\langle q, p \rangle\rangle & & \downarrow A \otimes [\text{inj}_2, \text{inj}_1] & & \downarrow [\text{inj}_2, \text{inj}_1] & & \downarrow [\text{inj}_2, \text{inj}_1] & & \nearrow [g, f] \\
 & & & & A \otimes (I + I) & \xrightarrow{d} & A \otimes I + A \otimes I & \xrightarrow{\lambda + \lambda} & A + A & &
 \end{array}$$

- $0f + 1g = g$. We have the following commutative diagram.



- $(a + b)(\frac{a}{a+b}f + \frac{b}{a+b}g) + (c + d)(\frac{c}{c+d}h + \frac{d}{c+d}w) = (a + c)(\frac{a}{a+c}f + \frac{c}{a+c}h) + (b + d)(\frac{b}{b+d}g + \frac{d}{b+d}w)$.
Let us write $\alpha = \frac{a}{a+b}f + \frac{b}{a+b}g$ and $\beta = \frac{c}{c+d}h + \frac{d}{c+d}w$. We have the following commutative diagram.



Thus

$$\begin{aligned} & (a + b)(\frac{a}{a+b}f + \frac{b}{a+b}g) + (c + d)(\frac{c}{c+d}h + \frac{d}{c+d}w) \\ &= [\text{Id}, \text{Id}] \circ ([f, g] + [h, w]) \circ ((\lambda + \lambda) + (\lambda + \lambda)) \circ (d + d) \circ d \\ & \quad \circ (A \otimes (\langle \frac{a}{a+b}, \frac{b}{a+b} \rangle + \langle \frac{c}{c+d}, \frac{d}{c+d} \rangle)) \circ (A \otimes \langle a + b, c + d \rangle) \circ \lambda^{-1}. \end{aligned}$$

Similarly, we can show that

$$\begin{aligned} & (a + c)(\frac{a}{a+c}f + \frac{c}{a+c}h) + (b + d)(\frac{b}{b+d}g + \frac{d}{b+d}w) \\ &= [\text{Id}, \text{Id}] \circ ([f, h] + [g, w]) \circ ((\lambda + \lambda) + (\lambda + \lambda)) \circ (d + d) \circ d \\ & \quad \circ (A \otimes (\langle \frac{a}{a+c}, \frac{c}{a+c} \rangle + \langle \frac{b}{b+d}, \frac{d}{b+d} \rangle)) \circ (A \otimes \langle a + c, b + d \rangle) \circ \lambda^{-1}. \end{aligned}$$

Thus we can show

$$(a + b)(\frac{a}{a+b}f + \frac{b}{a+b}g) + (c + d)(\frac{c}{c+d}h + \frac{d}{c+d}w) = (a + c)(\frac{a}{a+c}f + \frac{c}{a+c}h) + (b + d)(\frac{b}{b+d}g + \frac{d}{b+d}w)$$

by the following commutative diagram.

$$\begin{array}{ccc}
& A \otimes I & \\
A \otimes \langle\langle a+b, c+d \rangle\rangle \swarrow & & \searrow A \otimes \langle\langle a+c, b+d \rangle\rangle \\
A \otimes (I+I) & & A \otimes (I+I) \\
\downarrow A \otimes \langle\langle \frac{a}{a+b}, \frac{b}{a+b} \rangle\rangle + \langle\langle \frac{c}{c+d}, \frac{d}{c+d} \rangle\rangle & & \downarrow A \otimes \langle\langle \frac{a}{a+c}, \frac{c}{a+c} \rangle\rangle + \langle\langle \frac{b}{b+d}, \frac{d}{b+d} \rangle\rangle \\
A \otimes ((I+I) + (I+I)) & \xrightarrow{A \otimes \text{iso}} & A \otimes ((I+I) + (I+I)) \\
\downarrow d & & \downarrow d \\
A \otimes (I+I) + A \otimes (I+I) & & A \otimes (I+I) + A \otimes (I+I) \\
\downarrow d+d & & \downarrow d+d \\
(A \otimes I + A \otimes I) + (A \otimes I + A \otimes I) & \xrightarrow{\text{iso}} & (A \otimes I + A \otimes I) + (A \otimes I + A \otimes I) \\
\downarrow (\lambda+\lambda) + (\lambda+\lambda) & & \downarrow (\lambda+\lambda) + (\lambda+\lambda) \\
(A+A) + (A+A) & \xrightarrow{\text{iso}} & (A+A) + (A+A) \\
\downarrow [f,g] + [h,w] & & \downarrow [f,w] + [g,h] \\
B+B & \xrightarrow{[\text{Id}, \text{Id}]} & B \xleftarrow{[\text{Id}, \text{Id}]} B+B
\end{array}$$

- $(pf + qg) \circ e = p(f \circ e) + q(g \circ e)$. This is by the following commutative diagram.

$$\begin{array}{ccccccc}
C & \xrightarrow{e} & A & \xrightarrow{\lambda^{-1}} & A \otimes I & \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} & A \otimes (I+I) & \xrightarrow{d} & A \otimes I + A \otimes I & \xrightarrow{\lambda+\lambda} & A+A & \xrightarrow{[f,g]} & B \\
& & & \searrow \lambda^{-1} & \uparrow e \otimes I & & \uparrow e \otimes (I+I) & & \uparrow e \otimes I + e \otimes I & & \uparrow e+e & & \nearrow [f \circ e, g \circ e] \\
& & & & C \otimes I & \xrightarrow{C \otimes \langle\langle p, q \rangle\rangle} & C \otimes (I+I) & \xrightarrow{d} & C \otimes I + C \otimes I & \xrightarrow{\lambda+\lambda} & C+C & &
\end{array}$$

- $h \circ (pf + qg) = p(h \circ f) + q(h \circ g)$. This is by the following.

$$A \xrightarrow{\lambda^{-1}} A \otimes I \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} A \otimes (I+I) \xrightarrow{d} A \otimes I + A \otimes I \xrightarrow{\lambda+\lambda} A+A \xrightarrow{[f,g]} B \xrightarrow{h} C \quad \square$$

$\xrightarrow{[h \circ f, h \circ g]}$

Theorem C.3. *The category $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ is enriched in convex spaces. Moreover, the Lambek embedding $\kappa : \mathbf{Q} \hookrightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ preserves the convex sum in \mathbf{Q} .*

Proof. By Theorem C.2 (2), there exists a map $\langle\langle p, q \rangle\rangle : I \rightarrow I+I$ in \mathbf{Q} for any $p, q \in [0, 1]$, $p+q=1$, and it satisfies the four diagrams. Since κ preserves coproducts in \mathbf{Q} , the map $\kappa \langle\langle p, q \rangle\rangle : \kappa I \rightarrow \kappa I + \kappa I$ in $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ also satisfies the four diagrams in Theorem C.2 (2). Therefore $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ is enriched in convex spaces.

For all $f, g \in \mathbf{Q}(A, B)$, the convex sum $pf + qg \in \mathbf{Q}(A, B)$ is defined to be the following.

$$A \xrightarrow{\lambda^{-1}} A \otimes I \xrightarrow{A \otimes \langle\langle p, q \rangle\rangle} A \otimes (I+I) \xrightarrow{d} A \otimes I + A \otimes I \xrightarrow{\lambda+\lambda} A+A \xrightarrow{[f,g]} B$$

Since κ preserves coproducts in \mathbf{Q} and it is strong monoidal, we have $\kappa(pf + qg) = p\kappa(f) + q\kappa(g) \in [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(\kappa A, \kappa B)$. □

D Proof of Theorem 3.9

In this section, we assume \mathcal{V} to be a complete, cocomplete, symmetric monoidal closed category. The following proposition is due to Kock [13].

Proposition D.1. *Let $T : \mathcal{V} \rightarrow \mathcal{V}$ be a \mathcal{V} -monad. Then T is a strong monad with strength $t : A \otimes TB \rightarrow T(A \otimes B)$ given by the following commutative diagram. Note that η is the unit of the adjunction $-\otimes A \dashv A \multimap -$.*

$$\begin{array}{ccc} A & \xrightarrow{\text{curry}(t)} & TB \multimap T(A \otimes B) \\ \downarrow \eta & \nearrow T_{B,A \otimes B} & \\ B \multimap A \otimes B & & \end{array}$$

Theorem D.2. *Let T be a strong monad on \mathcal{V} and $F : \mathbf{A}^{\text{op}} \rightarrow \mathcal{V}$ be a \mathcal{V} -functor. For all $A, B \in \mathbf{A}$, we have maps*

$$F_{AB} : \mathbf{A}(B, A) \rightarrow FB \multimap FA$$

and

$$(TF)_{AB} : \mathbf{A}(B, A) \rightarrow TFB \multimap TFA.$$

We have the following commutative diagram.

$$\begin{array}{ccc} \mathbf{A}(B, A) \otimes TFA & & \\ \downarrow t & \searrow \text{uncurry}((TF)_{AB}) & \\ T(\mathbf{A}(B, A) \otimes FA) & \xrightarrow{T\text{uncurry}(F_{AB})} & TFB \end{array}$$

Proof. By currying the diagram above, we just need to show the right triangle commutes in the following diagram.

$$\begin{array}{ccccc} & & \mathbf{A}(B, A) & & \\ & & \downarrow \eta & & \\ & & FA \multimap \mathbf{A}(B, A) \otimes FA & \xrightarrow{(TF)_{AB}} & TFA \multimap T(\mathbf{A}(B, A) \otimes FA) \\ & \swarrow F_{AB} & & \searrow T_{FA, \mathbf{A}(B, A) \otimes FA} & \\ FA \multimap FB & \xrightarrow{T_{FA, FB}} & TFA \multimap TFB & & \\ & \swarrow FA \multimap \text{uncurry}(F_{AB}) & & \swarrow TFA \multimap T\text{uncurry}(F_{AB}) & \\ & & & & \end{array}$$

Note that the bottom square commutes because of the \mathcal{V} -naturality of T . The left triangle commutes by the property of monoidal closedness. The front triangle commutes by definition of $(TF)_{AB}$. The back triangle commutes by Proposition D.1. \square

Theorem D.3. *Let $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A} \rightarrow \mathcal{V}$ be a \mathcal{V} -functor and let T be a strong monad on \mathcal{V} . Then there exists a natural map*

$$\xi_F : \int^{A \in \mathbf{A}} TF(A, A) \rightarrow T \int^{A \in \mathbf{A}} F(A, A).$$

Proof. Recall that by definition of coend, we have the following coequalizers.

$$\begin{aligned} \sum_{A,B \in \mathbf{A}} \mathbf{A}(B,A) \otimes F(A,B) &\xrightarrow[\rho_2]{\rho_1} \sum_{A \in \mathbf{A}} F(A,A) \xrightarrow{e} \int^{A \in \mathbf{A}} F(A,A) \\ \sum_{A,B \in \mathbf{A}} \mathbf{A}(B,A) \otimes TF(A,B) &\xrightarrow[\rho'_2]{\rho'_1} \sum_{A \in \mathbf{A}} TF(A,A) \xrightarrow{e'} \int^{A \in \mathbf{A}} TF(A,A) \end{aligned}$$

For any $A \in \mathbf{A}$, the functor $F(A, -) : \mathbf{A} \rightarrow \mathcal{V}$ gives rise to a map

$$F(A, -)_{BA} : \mathbf{A}(B,A) \rightarrow F(A,B) \multimap F(A,A)$$

for each $B \in \mathbf{A}$. The map ρ_1 is defined as the coproduct pairing $[\text{inj}_A \circ \text{uncurry}(F(A, -)_{BA})]_{A,B \in \mathbf{A}}$. For any $B \in \mathbf{A}$, the functor $F(-, B) : \mathbf{A}^{\text{op}} \rightarrow \mathcal{V}$ gives rise to a map

$$F(-, B)_{AB} : \mathbf{A}(B,A) \rightarrow F(A,B) \multimap F(B,B)$$

for each $A \in \mathbf{A}$. The map ρ_2 is defined as the coproduct pairing $[\text{inj}_B \circ \text{uncurry}(F(-, B)_{AB})]_{A,B \in \mathbf{A}}$. The maps ρ'_1, ρ'_2 are induced similarly.

Consider the following diagram.

$$\begin{array}{ccc} \int^A TF(A,A) & \xrightarrow{\quad \xi \quad} & T \int^A F(A,A) \\ \uparrow e' & & \uparrow Te \\ \sum_A TF(A,A) & \xrightarrow{[\text{inj}_A]_A} & T \sum_A F(A,A) \\ \uparrow \rho'_1 \uparrow \rho'_2 & & \uparrow T\rho_1 \uparrow T\rho_2 \\ \sum_{A,B} \mathbf{A}(B,A) \otimes TF(A,B) & \xrightarrow{\Sigma_{A,B} t} \sum_{A,B} T(\mathbf{A}(B,A) \otimes F(A,B)) & \xrightarrow{[\text{inj}_{A,B}]_{A,B}} T \sum_{A,B} \mathbf{A}(B,A) \otimes F(A,B) \end{array}$$

Note that $[\text{inj}_A]_A$ and $[\text{inj}_{A,B}]_{A,B}$ are coproduct pairings. The morphism $t : \mathbf{A}(B,A) \otimes TF(A,B) \rightarrow T(\mathbf{A}(B,A) \otimes F(A,B))$ is the strength map for T .

To show the existence of ξ , we just need to show $Te \circ [\text{inj}_A]_A \circ \rho'_1 = Te \circ [\text{inj}_A]_A \circ \rho'_2$, which is to show the bottom square commutes for ρ'_1 and $T\rho_1$ (ρ'_2 and $T\rho_2$). This is the case because of the following commutative diagram. Note that the left triangle commutes by Theorem D.2.

$$\begin{array}{ccc} \mathbf{A}(B,A) \otimes TF(A,B) & \xrightarrow{\rho'_1(A,B)} & TF(A,A) \xrightarrow{T\text{inj}_A} T \sum_A F(A,A) \\ \downarrow t & \nearrow T\rho_1(A,B) & \uparrow T\rho_1 \\ T(\mathbf{A}(B,A) \otimes F(A,B)) & \xrightarrow{T\text{inj}_{A,B}} & T \sum_{A,B} \mathbf{A}(B,A) \otimes F(A,B) \end{array}$$

Note that $\rho'_1(A,B)$ is a component of ρ'_1 and $\rho_1(A,B)$ is a component of ρ_1 . By the universal property of the coequalizer e' , there exists a unique arrow

$$\xi : \int^{A \in \mathbf{A}} TF(A,A) \rightarrow T \int^{A \in \mathbf{A}} F(A,A). \quad \square$$

Proposition D.4. Suppose $F : \mathbf{A}^{\text{op}} \rightarrow \mathcal{V}$. For all $B, C \in \mathbf{A}$, the following diagram commutes.

$$\begin{array}{ccc} \mathbf{A}(C, B) \otimes F(B) & \xrightarrow{e_B} & \int^B \mathbf{A}(C, B) \otimes F(B) \\ \text{uncurry}(F_{BC}) \downarrow & & \swarrow y \\ F(C) & & \end{array}$$

Note that y is an isomorphism expressing the Yoneda lemma in the language of coends, and $F_{BC} : \mathbf{A}(C, B) \rightarrow F(B) \multimap F(C)$, and e_B is the unit of the coend.

Proof sketch. Note that the map $\text{uncurry}(F_{BC}) : \mathbf{A}(C, B) \otimes F(B) \rightarrow F(C)$ is \mathcal{V} -natural in B . By the universal property of coends, there exists a map $y : \int^B \mathbf{A}(C, B) \otimes F(B) \rightarrow F(C)$ such that the diagram above commutes. Moreover, y is an isomorphism [11, Chapter 2.4]. \square

Theorem D.5. Let T be a strong monad on \mathcal{V} . For all $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A} \rightarrow \mathcal{V}$, the map $\xi_F : \int^{A \in \mathbf{A}} TF(A, A) \rightarrow T \int^{A \in \mathbf{A}} F(A, A)$ makes the following diagrams commute.

1.

$$\begin{array}{ccc} \int^A F(A, A) & & \\ \downarrow \eta & \searrow f\eta & \\ T \int^A F(A, A) & \xleftarrow{\xi} & \int^A TF(A, A) \end{array}$$

2.

$$\begin{array}{ccccc} \int^A TTF(A, A) & \xrightarrow{\xi} & T \int^A TF(A, A) & \xrightarrow{T\xi} & TT \int^A F(A, A) \\ \downarrow f\mu & & & & \downarrow \mu \\ \int^A TF(A, A) & \xrightarrow{\xi} & T \int^A F(A, A) & & \end{array}$$

3. Suppose $G : \mathbf{A}^{\text{op}} \rightarrow \mathcal{V}$ and $A \in \mathbf{A}$.

$$\begin{array}{ccc} \int^B \mathbf{A}(A, B) \otimes TGB & \xrightarrow{y'} & TGA \\ \downarrow f_t & & \uparrow T_y \\ \int^B T(\mathbf{A}(A, B) \otimes GB) & \xrightarrow{\xi} & T \int^B \mathbf{A}(A, B) \otimes GB \end{array}$$

Note that y', y are isomorphisms induced by the Yoneda lemma.

4. Suppose $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A} \rightarrow \mathcal{V}$ and $X \in \mathcal{V}$.

$$\begin{array}{ccc} (\int^A F(A, A)) \otimes TX & \xrightarrow{t} & T((\int^A F(A, A)) \otimes X) \\ \downarrow \cong & & \downarrow \cong \\ \int^A (F(A, A) \otimes TX) & & \\ \downarrow f_t & & \\ \int^A T(F(A, A) \otimes X) & \xrightarrow{\xi} & T \int^A (F(A, A) \otimes X) \end{array}$$

5. Suppose $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A} \rightarrow \mathcal{V}$ and $X \in \mathcal{V}$.

$$\begin{array}{ccc}
 \int^A (X \otimes TF(A,A)) & \xrightarrow{\int^A t} & \int^A T(X \otimes F(A,A)) \\
 \downarrow \cong & & \downarrow \xi \\
 X \otimes \int^A TF(A,A) & & T \int^A (X \otimes F(A,A)) \\
 \downarrow X \otimes \xi & & \downarrow \cong \\
 X \otimes T \int^A F(A,A) & \xrightarrow{t} & T(X \otimes \int^A F(A,A))
 \end{array}$$

6. Suppose $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A}^{\text{op}} \otimes \mathbf{A} \otimes \mathbf{A} \rightarrow \mathcal{V}$.

$$\begin{array}{ccc}
 \int^A \int^B TF(A,B,A,B) & \xrightarrow{\int^A \xi} & \int^A T \int^B F(A,B,A,B) & \xrightarrow{\xi} & T \int^A \int^B F(A,B,A,B) \\
 \downarrow \cong & & & & \downarrow \cong \\
 \int^{A,B} TF(A,B,A,B) & \xrightarrow{\xi} & T \int^{A,B} F(A,B,A,B)
 \end{array}$$

Proof. 1. We need to show that the following commutes.

$$\begin{array}{ccc}
 \int^A F(A,A) & & \\
 \downarrow \eta & \searrow \int^A \eta & \\
 T \int^A F(A,A) & \xleftarrow{\xi} & \int^A TF(A,A)
 \end{array}$$

Consider the following diagram. We write η_1 for the map $F(A,A) \rightarrow TF(A,A)$ and η_2 for the map $\int^A F(A,A) \rightarrow T \int^A F(A,A)$.

$$\begin{array}{ccccc}
 \int^A F(A,A) & \xrightarrow{\int^A \eta_1} & \int^A TF(A,A) & & \\
 \uparrow e & \searrow \eta_2 & \uparrow e' & \searrow \xi & \\
 & & & & T \int^A F(A,A) \\
 \Sigma_A F(A,A) & \xrightarrow{\Sigma \eta_1} & \Sigma_A TF(A,A) & \xrightarrow{\eta} & T \Sigma_A F(A,A) \\
 & & & \searrow [\text{inj}_A] & \uparrow Te \\
 & & & & T \Sigma_A F(A,A)
 \end{array}$$

We need to show that the top triangle commutes. Since e is an epimorphism, we just need to show $\xi \circ \int^A \eta_1 \circ e = \eta_2 \circ e$. This is the case because the bottom triangle commutes and all three square faces commute. The bottom triangle commutes by the universal property of coproducts. The square with ξ commutes by definition of ξ . Also note that $e' \circ \Sigma \eta_1 = \int^A \eta_1 \circ e$ is a property of coends (see [11, 4.2]).

2. The proof is similar to (1).

3. Next we need to show that the following commutes (where y, y' are isomorphisms induced by the Yoneda lemma).

$$\begin{array}{ccc} \int^B \mathbf{A}(A, B) \otimes TGB & \xrightarrow{f^t} & \int^B T(\mathbf{A}(A, B) \otimes GB) \\ \downarrow y' & & \downarrow \xi \\ TGA & \xleftarrow{T_y} & T \int^B \mathbf{A}(A, B) \otimes GB \end{array}$$

The above diagram commutes because the following diagram commutes for all $A, B \in \mathbf{A}$.

$$\begin{array}{ccccc} & & \mathbf{A}(A, B) \otimes TGB & \xrightarrow{e_1} & \int^B \mathbf{A}(A, B) \otimes TGB \\ & \swarrow \text{uncurry}((TG)_{BA}) & \downarrow t & & \downarrow f^t \\ TGA & \xleftarrow{\text{uncurry}((TG)_{BA})} & T(\mathbf{A}(A, B) \otimes GB) & \xrightarrow{e_2} & TGA & \xrightarrow{y'} & \int^B T(\mathbf{A}(A, B) \otimes GB) \\ & \uparrow T\text{uncurry}(G_{BA}) & \downarrow \text{Id} & \uparrow T_y & \downarrow \xi \\ & & T(\mathbf{A}(A, B) \otimes G(B)) & \xrightarrow{Te_3} & T \int^B \mathbf{A}(A, B) \otimes GB \end{array}$$

Since G and TG are contravariant \mathcal{V} -functors, there are the following maps in \mathcal{V} .

$$G_{BA} : \mathbf{A}(A, B) \rightarrow GB \Rightarrow GA$$

$$(TG)_{BA} : \mathbf{A}(A, B) \rightarrow TGB \Rightarrow TGA$$

The bottom square commutes by the definition of ξ , and the back square (with e_1, e_2) commutes by naturality of coends. The top and the front squares commutes because of Proposition D.4. Thus we just need to show that the left square commutes, i.e.,

$$\begin{array}{ccc} \mathbf{C}(C, B) \otimes TFB & \xrightarrow{t} & T(\mathbf{C}(C, B) \otimes FB) \\ \downarrow u' & \swarrow Tu & \\ TFC & & \end{array}$$

This commutes by Proposition D.2.

4. Next we need to prove that the following commutes.

$$\begin{array}{ccc} (\int^A F(A, A)) \otimes TX & \xrightarrow{t} & T((\int^A F(A, A)) \otimes X) \\ \downarrow \cong & & \downarrow \cong \\ \int^A (F(A, A) \otimes TX) & & \int^A T(F(A, A) \otimes X) \\ \downarrow f^t & & \downarrow \xi \\ \int^A T(F(A, A) \otimes X) & \xrightarrow{\xi} & T \int^A (F(A, A) \otimes X) \end{array}$$

First observe that the following commutes (each arrow is canonical).

$$\begin{array}{ccc} (\sum_A F(A, A)) \otimes TX & \longrightarrow & T((\sum_A F(A, A)) \otimes X) \\ \downarrow \cong & & \downarrow \cong \\ \sum_A (F(A, A) \otimes TX) & & \sum_A T(F(A, A) \otimes X) \\ \downarrow & & \downarrow \\ \sum_A T(F(A, A) \otimes X) & \longrightarrow & T \sum_A (F(A, A) \otimes X) \end{array}$$

Now let us consider the following cube.

$$\begin{array}{ccccc}
 & & (f^A F(A,A)) \otimes TX & \xrightarrow{t} & T((f^A F(A,A)) \otimes X) \\
 & \nearrow & \downarrow \cong & & \nearrow T(e \otimes X) \\
 (\Sigma_A F(A,A)) \otimes TX & \xrightarrow{t} & T((\Sigma_A F(A,A)) \otimes X) & & \\
 \downarrow \cong & & \downarrow & & \downarrow \cong \\
 \Sigma_A(F(A,A) \otimes TX) & \nearrow & f^A(F(A,A) \otimes TX) & & \\
 \downarrow \Sigma_A t & & \downarrow f t & & \\
 \Sigma_A T(F(A,A) \otimes X) & \nearrow & f^A T(F(A,A) \otimes X) & \xrightarrow{\xi} & T f^A(F(A,A) \otimes X) \\
 \downarrow & & \downarrow & & \downarrow \\
 \Sigma_A T(F(A,A) \otimes X) & \xrightarrow{[T \text{inj}]_A} & T \Sigma_A(F(A,A) \otimes X) & \nearrow T e' & \\
 & & \downarrow \cong & &
 \end{array}$$

Note that the top square commutes, by naturality of t . The bottom square commutes, by definition of ξ . The left square involving $\Sigma_A t, f t$ commutes by naturality of coend. The right square and the left top square commute for the same reason. For simplicity, consider the following diagram.

$$\begin{array}{ccccc}
 (\Sigma_{A,B} \mathbf{A}(B,A) \otimes F(A,B)) \otimes X & \rightrightarrows & (\Sigma_A F(A,A)) \otimes X & \twoheadrightarrow & (f^A F(A,A)) \otimes X \\
 \downarrow \cong & & \downarrow \cong & & \downarrow \cong \\
 \Sigma_{A,B}(\mathbf{A}(B,A) \otimes F(A,B) \otimes X) & \rightrightarrows & \Sigma_A(F(A,A) \otimes X) & \twoheadrightarrow & f^A(F(A,A) \otimes X)
 \end{array}$$

Note that the right square is the same square as the right square in the cube. And $- \otimes X$ preserves coequalizers. The left square commutes by naturality. This implies that the right square commutes, by the universal property of coequalizers. Therefore the cube above commutes.

5. Next, we need to show that the following diagram commutes.

$$\begin{array}{ccc}
 f^A(X \otimes TF(A,A)) & \xrightarrow{f t} & f^A T(X \otimes F(A,A)) \\
 \downarrow \cong & & \downarrow \xi \\
 X \otimes f^A TF(A,A) & & T f^A(X \otimes F(A,A)) \\
 \downarrow \text{Id} \otimes \xi & & \downarrow \cong \\
 X \otimes T f^A F(A,A) & \xrightarrow{t} & T(X \otimes f^A F(A,A))
 \end{array}$$

First, observe that the following diagram commutes.

$$\begin{array}{ccc}
 \Sigma_A(X \otimes TF(A,A)) & \longrightarrow & \Sigma_A T(X \otimes F(A,A)) \\
 \downarrow \cong & & \downarrow \\
 X \otimes \Sigma_A TF(A,A) & & T \Sigma_A(X \otimes F(A,A)) \\
 \downarrow & & \downarrow \cong \\
 X \otimes T \Sigma_A F(A,A) & \longrightarrow & T(X \otimes \Sigma_A F(A,A))
 \end{array}$$

Now let us consider the following cube.

$$\begin{array}{ccccc}
 & & f^A(X \otimes TF(A,A)) & \xrightarrow{f^t} & f^A T(X \otimes F(A,A)) \\
 & \nearrow & \downarrow \cong & & \downarrow \xi \\
 \Sigma_A(X \otimes TF(A,A)) & \xrightarrow{\Sigma^t} & \Sigma_A T(X \otimes F(A,A)) & & \\
 \downarrow \cong & & \downarrow & & \downarrow \\
 X \otimes \Sigma_A TF(A,A) & \nearrow & X \otimes f^A TF(A,A) & & T f^A(X \otimes F(A,A)) \\
 & & \downarrow X \otimes \xi & & \downarrow \cong \\
 & & X \otimes T f^A F(A,A) & \xrightarrow{t} & T(X \otimes f^A F(A,A)) \\
 \downarrow & & \downarrow \cong & & \downarrow \\
 X \otimes T \Sigma_A F(A,A) & \xrightarrow{t} & T(X \otimes \Sigma_A F(A,A)) & & \\
 & \nearrow & & & \nearrow Te' \\
 & & & & T \Sigma_A(X \otimes F(A,A)) \\
 & & & & \downarrow \cong \\
 & & & & T f^A(X \otimes F(A,A))
 \end{array}$$

Note that the top square commutes by naturality of coends. The bottom square commutes by naturality of t . The left bottom and the right top square involving ξ commute by definition. The left top and the right bottom square commute for the same reason. Consider the following diagram.

$$\begin{array}{ccccc}
 \Sigma_{A,B}(\mathbf{A}(B,A) \otimes X \otimes F(A,B)) & \rightrightarrows & \Sigma_A(X \otimes F(A,A)) & \twoheadrightarrow & f^A(X \otimes F(A,A)) \\
 \downarrow \cong & & \downarrow \cong & & \downarrow \cong \\
 X \otimes \Sigma_{A,B}(\mathbf{A}(B,A) \otimes F(A,B)) & \rightrightarrows & X \otimes \Sigma_A F(A,A) & \twoheadrightarrow & X \otimes f^A F(A,A)
 \end{array}$$

Note that the right square is the same square as the right bottom square in the cube under the functor T . And $X \otimes -$ preserves coequalizers. The left square commutes by naturality. This implies that the right square commutes, by the universal property of coequalizers. Therefore the cube above commutes.

6. Let $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A}^{\text{op}} \otimes \mathbf{A} \otimes \mathbf{A} \rightarrow \mathcal{V}$. We now need to show that the following diagram commutes.

$$\begin{array}{ccc}
 \int^A \int^B TF(A,B,A,B) & \xrightarrow{f^A \xi} & \int^A T \int^B F(A,B,A,B) & \xrightarrow{\xi} & T \int^A \int^B F(A,B,A,B) \\
 \downarrow f & & & & \downarrow Tf \\
 \int^{A,B} TF(A,B,A,B) & \xrightarrow{\xi} & T \int^{A,B} F(A,B,A,B) & &
 \end{array}$$

First, the isomorphisms f, Tf above are instances of so-called Fubini theorem for coends, which also gives rise to the following commutative diagram for any $F : \mathbf{A}^{\text{op}} \otimes \mathbf{A}^{\text{op}} \otimes \mathbf{A} \otimes \mathbf{A} \rightarrow \mathcal{V}$.

$$\begin{array}{ccc}
 F(A,B,A,B) & \xrightarrow{e_{A,B}} & \int^{A,B} F(A,B,A,B) \\
 \downarrow e_B & & \uparrow f \\
 \int^B F(A,B,A,B) & \xrightarrow{e_A} & \int^A \int^B F(A,B,A,B)
 \end{array}$$

We have the following commutative diagram.

$$\begin{array}{ccccc}
 & & \int^A \int^B TF(A, B, A, B) & \xrightarrow{\int^A \xi} & \int^A T \int^B F(A, B, A, B) & \xrightarrow{\xi} & T \int^A \int^B F(A, B, A, B) \\
 & \swarrow f & \uparrow e'_A & & \uparrow e'_A & \nearrow Te_A & \downarrow Tf \\
 \int^{A,B} TF(A, B, A, B) & & \int^B TF(A, B, A, B) & \xrightarrow{\xi} & T \int^B F(A, B, A, B) & & T \int^{A,B} F(A, B, A, B) \\
 \uparrow e'_{A,B} & \nearrow e'_B & \nearrow Te_B & & \nearrow Te_{A,B} & & \\
 TF(A, B, A, B) & & & \xrightarrow{\xi} & & &
 \end{array}$$

Note that above diagram commutes, by properties of the Fubini theorem, definition of ξ , \mathcal{V} -naturality of coends, and because $e'_{A,B}$ is an epimorphism. \square

Theorem D.6. *Let \mathbf{A} be a \mathcal{V} -category. If T is a commutative strong monad on \mathcal{V} (the strength is given by the map $t_{A,B} : A \otimes TB \rightarrow T(A \otimes B)$ for any $A, B \in \mathbf{A}$), then $\overline{T}(F) = T \circ F$ is a commutative strong \mathcal{V} -monad on $\mathcal{V}^{\mathbf{A}^{\text{op}}}$.*

Proof. It is straightforward to verify that \overline{T} is a monad. We define the strength \bar{t} to be the following composition.

$$\begin{aligned}
 (F \otimes_{\text{Day}} \overline{T}G)(C) &= \int^{(A,B) \in \mathbf{A} \otimes \mathbf{A}} \mathbf{A}(C, A \otimes B) \otimes FA \otimes TGB \\
 &\xrightarrow{\int^{(A,B)} t} \int^{(A,B) \in \mathbf{A} \otimes \mathbf{A}} T(\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) \\
 &\xrightarrow{\xi} T \int^{(A,B) \in \mathbf{A} \otimes \mathbf{A}} (\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) \\
 &= \overline{T}(F \otimes_{\text{Day}} G)(C)
 \end{aligned}$$

Now to show that \overline{T} is a commutative strong monad, we need to show the following diagrams commute.

•

$$\begin{array}{ccc}
 F \otimes_{\text{Day}} G & \xrightarrow{F \otimes_{\text{Day}} \eta} & F \otimes_{\text{Day}} \overline{T}G \\
 \downarrow \eta & \swarrow \bar{t} & \\
 \overline{T}(F \otimes_{\text{Day}} G) & &
 \end{array}$$

To show this, we just need to show that the following diagram commutes for any $C \in \mathbf{A}$.

$$\begin{array}{ccc}
 \int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes GB & \xrightarrow{\int^{A,B} \eta'} & \int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes TGB \\
 \downarrow \eta & \searrow \int^{A,B} \eta & \downarrow \int^{A,B} t \\
 T(\int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) & \xleftarrow{\xi} & \int^{A,B} T(\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB)
 \end{array}$$

Note that $\int^{A,B} \eta'$ is a shorthand for $\int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes \eta$. Similarly, $\int^{A,B} t$ is a shorthand for $\int^{A,B} t_{\mathbf{A}(C, A \otimes B) \otimes FA, GB}$ in the above diagram. The bottom triangle commutes because of Theorem D.5(1). The top triangle commutes by properties of t .

- We need to show that the following diagram commutes.

$$\begin{array}{ccc}
 I \otimes_{\text{Day}} \overline{TF} & \xrightarrow{\overline{t}_{I,F}} & \overline{T}(I \otimes_{\text{Day}} F) \\
 & \searrow \lambda_{\mathcal{T}F} & \downarrow \overline{T}\lambda_F \\
 & & \overline{TF}
 \end{array}$$

If we unfold the definition of the Day tensor, we have the following diagram for any $C \in \mathbf{A}$.

$$\begin{array}{ccccc}
 & & \int^B \mathbf{A}(C, I \otimes B) \otimes TFB & \xrightarrow{\int t} & \int^B T(\mathbf{A}(C, I \otimes B) \otimes FB) \\
 & \swarrow & & \swarrow \int T\lambda & \searrow \xi \\
 \int^B \mathbf{A}(C, B) \otimes TFB & \xrightarrow{\int t', \int \lambda} & \int^B T(\mathbf{A}(C, B) \otimes FB) & & T \int^B \mathbf{A}(C, I \otimes B) \otimes FB \\
 & \searrow \cong & \searrow \xi & & \swarrow T \int \lambda \\
 & & TFC & \xrightarrow{\cong} & T \int^B \mathbf{A}(C, B) \otimes FB
 \end{array}$$

Note that for any $C \in \mathbf{A}$, the following commutes by naturality of t .

$$\begin{array}{ccc}
 \int^B \mathbf{A}(C, I \otimes B) \otimes TFB & \xrightarrow{\int t} & \int^B T(\mathbf{A}(C, I \otimes B) \otimes FB) \\
 \int \lambda_B \downarrow & & \downarrow \int T\lambda_B \\
 \int^B \mathbf{A}(C, B) \otimes TFB & \xrightarrow{\int t'} & \int^B T(\mathbf{A}(C, B) \otimes FB)
 \end{array}$$

Note that $I = \mathbf{A}(-, I) \in \mathcal{V}^{\text{Aop}}$, and $\int t$ is a shorthand for $\int^B t_{\mathbf{A}(C, I \otimes B), FB}$, and $\int \lambda_B$ is a shorthand for $\int^B \mathbf{A}(C, \lambda_B) \otimes \text{Id}_{TFB}$, and $\int T\lambda_B$ is a shorthand for $\int^B T\mathbf{A}(C, \lambda_B) \otimes \text{Id}_{FB}$, and $\int t'$ is a shorthand for $\int^B t_{\mathbf{A}(C, B), FB}$.

The bottom square commutes because of Theorem D.5(3). The right square commutes by the naturality of ξ .

- Next we need to show that the following diagram commutes.

$$\begin{array}{ccc}
 F \otimes_{\text{Day}} \overline{TTG} & \xrightarrow{t} & \overline{T}(F \otimes_{\text{Day}} \overline{TG}) \xrightarrow{\overline{T}t} \overline{TT}(F \otimes_{\text{Day}} G) \\
 \downarrow \text{Id}_F \otimes_{\text{Day}} \mu & & \downarrow \mu \\
 F \otimes_{\text{Day}} \overline{TG} & \xrightarrow{t} & \overline{T}(F \otimes_{\text{Day}} G)
 \end{array}$$

The above diagram commutes because for any $C \in \mathbf{A}$, we have the following commutative diagram.

$$\begin{array}{ccccc}
 \int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes TTGB & \xrightarrow{\int t} & \int^{A,B} T(\mathbf{A}(C, A \otimes B) \otimes FA \otimes TGB) & \xrightarrow{\xi} & T \int^{A,B} (\mathbf{A}(C, A \otimes B) \otimes FA \otimes TGB) \\
 \downarrow \int \text{Id} \otimes \mu & & \downarrow \int Tt & & \downarrow T \int t \\
 \int^{A,B} \mathbf{A}(C, A \otimes B) \otimes FA \otimes TGB & & \int^{A,B} TT(\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) & \xrightarrow{\xi} & T \int^{A,B} T(\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) \\
 \downarrow \int t & \swarrow \int \mu & & & \downarrow T\xi \\
 \int^{A,B} T(\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) & \xrightarrow{\xi} & T \int^{A,B} (\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB) & \xleftarrow{\mu} & TT \int^{A,B} (\mathbf{A}(C, A \otimes B) \otimes FA \otimes GB)
 \end{array}$$

Note that the top right square commutes by naturality of ξ , the bottom diagram commutes by Theorem D.5 (2), and the left diagram commutes by properties of t .

- Next we need to show that the following diagram commutes.

$$\begin{array}{ccc}
(F \otimes_{\text{Day}} G) \otimes_{\text{Day}} \overline{TH} & \xrightarrow{t} & \overline{T}((F \otimes_{\text{Day}} G) \otimes_{\text{Day}} H) \\
\downarrow \alpha & & \downarrow \overline{T}\alpha \\
F \otimes_{\text{Day}} (G \otimes_{\text{Day}} \overline{TH}) & \xrightarrow{\text{Id}_F \otimes_{\text{Day}} t} F \otimes_{\text{Day}} \overline{T}(G \otimes_{\text{Day}} H) \xrightarrow{t} & \overline{T}(F \otimes_{\text{Day}} (G \otimes_{\text{Day}} H))
\end{array}$$

For any $C \in \mathbf{A}$, we have

$$((F \otimes_{\text{Day}} G) \otimes_{\text{Day}} \overline{TH})(C) \cong \int^{B \in \mathbf{A}} \int^{(X,Y) \in \mathbf{A} \otimes \mathbf{A}} \mathbf{A}(C, (X \otimes Y) \otimes B) \otimes FX \otimes GY \otimes THB$$

and

$$\begin{aligned}
(F \otimes_{\text{Day}} (G \otimes_{\text{Day}} \overline{TH}))(C) &\cong \int^{X \in \mathbf{A}} \int^{(Y,B) \in \mathbf{A} \otimes \mathbf{A}} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes THB \\
&\cong \int^{X,Y,B} FX \otimes GY \otimes THB \otimes \int^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A).
\end{aligned}$$

Consider the following diagram. We need to show that the outermost diagram commutes. Note that $\int y, T \int y, a, a'$ are all isomorphisms.

$$\begin{array}{ccccc}
j^{BXY} \mathbf{A}(C, (X \otimes Y) \otimes B) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} T(\mathbf{A}(C, (X \otimes Y) \otimes B) \otimes FX \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{BXY} \mathbf{A}(C, (X \otimes Y) \otimes B) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} T(\mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{YB} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes THB & & T j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB & & T j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{YB} T(\mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB) & & T j^{BXY} j^A \mathbf{A}(C, X \otimes A) \otimes \mathbf{A}(A, Y \otimes B) \otimes FX \otimes GY \otimes HB & & T j^{BXY} j^A \mathbf{A}(C, X \otimes A) \otimes \mathbf{A}(A, Y \otimes B) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes T j^{YB} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB & \xrightarrow{j^t} & j^{AX} T(\mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{YB} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{YB} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB
\end{array}$$

Note that the top square and the top right square commute by naturality of t and ξ . We just need to show that the bottom diagram commutes. The expanded bottom diagram is the following.

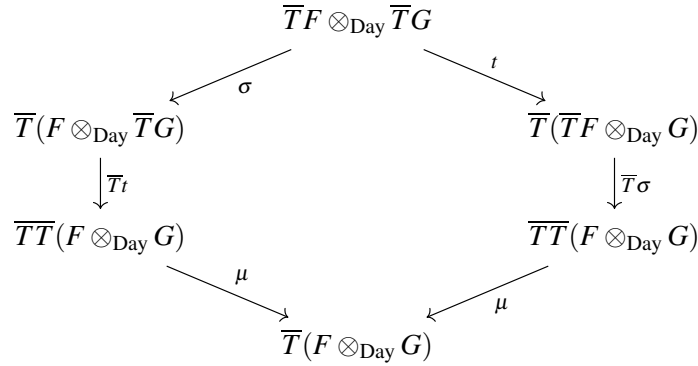
$$\begin{array}{ccccccc}
j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} T(\mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{BXY} \mathbf{A}(C, X \otimes (Y \otimes B)) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{BXY} (j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A)) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} T((j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A)) \otimes FX \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{BXY} (j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A)) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{BXY} j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} j^A T(\mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A)) \otimes FX \otimes GY \otimes HB & \xrightarrow{\xi} & T j^{BXY} j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{BXY} j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A) \otimes FX \otimes GY \otimes THB & \xrightarrow{j^t} & j^{BXY} j^A T(\mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A)) \otimes FX \otimes GY \otimes HB & \xrightarrow{\xi} & T j^{BXY} j^A \mathbf{A}(A, Y \otimes B) \otimes \mathbf{A}(C, X \otimes A) \otimes FX \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} j^{BY} \mathbf{A}(C, X \otimes A) \otimes FX \otimes \mathbf{A}(A, Y \otimes B) \otimes GY \otimes THB & \xrightarrow{j^t} & j^{AX} j^{BY} T(\mathbf{A}(C, X \otimes A) \otimes FX \otimes \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB) & \xrightarrow{\xi} & T j^{AX} j^{BY} \mathbf{A}(C, X \otimes A) \otimes FX \otimes \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{BY} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes THB & \xrightarrow{j^t} & j^{AX} j^{BY} T(\mathbf{A}(C, X \otimes A) \otimes FX \otimes T(\mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB)) & \xrightarrow{\xi} & T j^{AX} j^{BY} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{BY} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB \\
\downarrow j^a & & \downarrow j^a & & \downarrow j^a \\
j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{BY} T(\mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB) & \xrightarrow{j^t} & j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes T j^{BY} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB & \xrightarrow{\xi} & T j^{AX} \mathbf{A}(C, X \otimes A) \otimes FX \otimes j^{BY} \mathbf{A}(A, Y \otimes B) \otimes GY \otimes HB
\end{array}$$

Our goal is to show that the outermost diagram commutes. Note that all the inner diagrams commute, by Theorem D.3(4)–(6) and naturality. Therefore the whole diagram commutes.

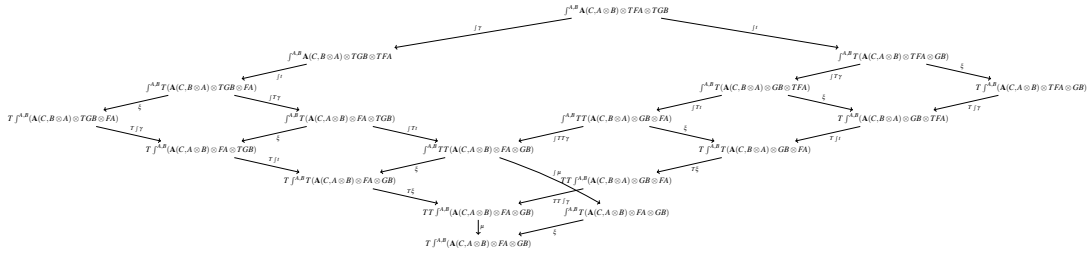
- Lastly, since $\mathcal{V}^{\text{A}^{\text{op}}}$ is a symmetric monoidal \mathcal{V} -category with $\gamma_{F,G} : F \otimes_{\text{Day}} G \rightarrow G \otimes_{\text{Day}} F$, we can define the costrength as the following for any $F, B \in \mathcal{V}^{\text{A}^{\text{op}}}$.

$$\sigma_{F,G} := \overline{T} \gamma_{G,F} \circ t_{G,F} \circ \gamma_{\overline{T}F,G} : \overline{T}F \otimes_{\text{Day}} G \rightarrow \overline{T}(F \otimes_{\text{Day}} G)$$

We need to show that the following diagram commutes.



For any $C \in \mathbf{A}$, the above diagram can be expanded to the following diagram. We need to show the outermost diagram commutes.



It commutes because every inner diagram commutes (by naturality and Theorem D.3(2)). \square

E Proof of Theorem 4.4

We write $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ for the full subcategory of $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$ consisting of product-preserving functors. We write $L : \mathbf{Set}^{\mathbf{Q}^{\text{op}}} \rightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ for the left adjoint of the inclusion functor $i : [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}} \hookrightarrow \mathbf{Set}^{\mathbf{Q}^{\text{op}}}$. We write $\eta : \text{Id} \rightarrow iL$ for the unit of the adjunction.

Definition E.1. We define a function $\tilde{L} : \overline{\mathbf{C}} \rightarrow \tilde{\mathbf{C}}$ as follows.

- For any $F \in \overline{\mathbf{C}}$, we define $\tilde{L}(F) = G$ such that for all $A \in \mathbf{C}$,

$$G(A) = ((iLF^0)A, (FA)_1, (\eta_{F^0})_A \circ h_A : (FA)_1 \rightarrow (iLF^0)A) \in \mathcal{V},$$

where $h_A : (FA)_1 \rightarrow (FA)_0$ and $(\eta_{F^0})_A : F^0A \rightarrow (iLF^0)A$.

For any $A, B \in \mathbf{C}$, we define G_{AB}^0 and G_{AB}^1 by the following.

$$\begin{array}{ccc}
 f \in \mathbf{M}^{\text{op}}(A, B) & \xrightarrow{G_{AB}^1} & (F_{AB}^1 f, (iLF^0)(J_{AB}f)) \in \mathcal{V}(GA, GB) \\
 \downarrow J_{AB} & & \downarrow p_0 \\
 J_{AB}f \in \mathbf{Q}^{\text{op}}(A, B) & \xrightarrow{G_{AB}^0} & (iLF^0)(J_{AB}f) \in \mathbf{Set}((iLF^0)A, (iLF^0)B)
 \end{array}$$

Note that G is smooth, hence $G \in \tilde{\mathbf{C}}$.

Proposition E.2. $\tilde{L} : \overline{\mathbf{C}} \rightarrow \tilde{\mathbf{C}}$ is a \mathcal{V} -functor.

Proof. We just need to show that for all $F, G \in \overline{\mathbf{C}}$, there is a morphism

$$\tilde{L}_{FG} : \overline{\mathbf{C}}(F, G) \rightarrow \tilde{\mathbf{C}}(\tilde{L}F, \tilde{L}G)$$

in \mathcal{V} . This is provided by the following commuting square in \mathbf{Set} .

$$\begin{array}{ccc} \{(\alpha_A : FA \rightarrow GA)_{A \in \mathbf{C}} \mid \alpha \in \mathcal{V}\text{-Nat}(F, G)\} & \xrightarrow{\tilde{L}_{FG}^1} & \{(\beta_A : (i\tilde{L}F)A \rightarrow (i\tilde{L}G)A)_{A \in \mathbf{C}} \mid \beta \in \mathcal{V}\text{-Nat}(i\tilde{L}F, i\tilde{L}G)\} \\ \downarrow & & \downarrow \\ \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, G^0) & \xrightarrow{\tilde{L}_{FG}^0} & [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(LF^0, LG^0) \end{array}$$

We write $\mathcal{V}\text{-Nat}(F, G)$ for the set of \mathcal{V} -natural transformations from F to G . The arrow \tilde{L}_{FG}^0 is given by the functor $L : \mathbf{Set}^{\mathbf{Q}^{\text{op}}} \rightarrow [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$. And the arrow \tilde{L}_{FG}^1 is given by extending the commuting square $\alpha_A : FA \rightarrow GA$ with $\eta : \text{Id} \rightarrow iL$, as in the following diagram. Note that for each \mathcal{V} -natural transformation $\alpha \in \mathcal{V}\text{-Nat}(F, G)$, we have $\alpha^0 : F^0 \rightarrow G^0$.

$$\begin{array}{ccc} (FA)_1 & \xrightarrow{\alpha_A^1} & (GA)_1 \\ \downarrow & & \downarrow \\ (FA)_0 & \xrightarrow{\alpha_A^0} & (GA)_0 \\ \downarrow (\eta_{F^0})_A & & \downarrow (\eta_{G^0})_A \\ (iLF^0)A & \xrightarrow{(iL)\alpha_A^0} & (iLG^0)A \end{array}$$

□

Theorem E.3. *The \mathcal{V} -category $\tilde{\mathbf{C}}$ is a reflective \mathcal{V} -subcategory of $\overline{\mathbf{C}}$, i.e., the inclusion \mathcal{V} -functor $i : \tilde{\mathbf{C}} \hookrightarrow \overline{\mathbf{C}}$ has a left adjoint \tilde{L} .*

Proof sketch. We need to show $\overline{\mathbf{C}}(F, iG) \cong \tilde{\mathbf{C}}(\tilde{L}F, G)$ for any $F \in \overline{\mathbf{C}}, G \in \tilde{\mathbf{C}}$ and it is \mathcal{V} -natural in F and G . We just need to show the following diagram commutes.

$$\begin{array}{ccc} \mathcal{V}\text{-Nat}(F, iG) & \xrightarrow{\cong} & \mathcal{V}\text{-Nat}(i\tilde{L}F, G) \\ \downarrow & & \downarrow \\ \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(F^0, iG^0) & \xrightarrow{\cong} & [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(LF^0, G^0) \end{array}$$

The bottom arrow is an isomorphism because $L \dashv i$. The top arrow is an isomorphism because for any $A \in \mathbf{C}$ and \mathcal{V} -natural transformation $\gamma : F \rightarrow iG$, we have the following commutative diagram.

$$\begin{array}{ccc} (FA)_1 & \xrightarrow{\gamma_A^1} & ((iG)A)_1 \\ \downarrow & & \downarrow \\ (FA)_0 & \xrightarrow{\gamma_A^0} & ((iG)A)_0 \\ (\eta_{F^0})_A \downarrow & \nearrow i(\hat{\gamma}^0)_A & \\ (iLF^0)A & & \end{array}$$

□

F Day's reflection theorem for $\tilde{\mathbf{C}}$

Theorem F.1. *If $H \in \tilde{\mathbf{C}}$, then $G \multimap_{\text{Day}} iH$ is also a smooth functor for any $G \in \overline{\mathbf{C}}$.*

Proof. Suppose $H \in \tilde{\mathbf{C}}$ and $G \in \overline{\mathbf{C}}$. For any $C \in \mathbf{C}$, we have

$$(G \multimap_{\text{Day}} iH)(C) = \int_{A \in \mathbf{C}} GA \Rightarrow iH(C \otimes A) \cong \overline{\mathbf{C}}(G, iH(C \otimes -)).$$

Thus

$$\begin{aligned} (G \multimap_{\text{Day}} H)(C)_0 &\cong \overline{\mathbf{C}}(G, iH(C \otimes -))_0 \cong \mathbf{Set}^{\mathbf{Q}^{\text{op}}}(G^0, (iH)^0(C \otimes -)) \\ &\cong \int_{A \in \mathbf{Q}} \mathbf{Set}(G^0 A, (iH)^0(C \otimes A)) \cong (G^0 \multimap_{\text{Day}} (iH)^0)(C), \end{aligned}$$

where $G^0 \multimap_{\text{Day}} (iH)^0$ is an exponential in $\mathbf{Set}^{\mathbf{Q}^{\text{op}}}$. Since H^0 preserves products, so does $G^0 \multimap_{\text{Day}} (iH)^0$, thus $G \multimap_{\text{Day}} iH$ is smooth and $G \multimap_{\text{Day}} iH \in \tilde{\mathbf{C}}$. The functor $G^0 \multimap_{\text{Day}} (iH)^0$ preserves products in \mathbf{Q}^{op} because for any $C_1, C_2 \in \mathbf{C}$, we have

$$\begin{aligned} (G^0 \multimap_{\text{Day}} iH^0)(C_1 + C_2) &= \int_A \mathbf{Set}(G^0 A, iH^0((C_1 + C_2) \otimes A)) \\ &\cong \int_A \mathbf{Set}(G^0 A, iH^0(C_1 \otimes A + C_2 \otimes A)) \\ &\cong \int_A \mathbf{Set}(G^0 A, iH^0(C_1 \otimes A) \times iH^0(C_2 \otimes A)) \\ &\cong \int_A \mathbf{Set}(G^0 A, iH^0(C_1 \otimes A)) \times \mathbf{Set}(G^0 A, iH^0(C_2 \otimes A)) \\ &\cong \int_A \mathbf{Set}(G^0 A, iH^0(C_1 \otimes A)) \times \int_A \mathbf{Set}(G^0 A, iH^0(C_2 \otimes A)) \\ &= (G^0 \multimap_{\text{Day}} iH^0)(C_1) \times (G^0 \multimap_{\text{Day}} iH^0)(C_2). \quad \square \end{aligned}$$

The above theorem implies that for any $G \in \tilde{\mathbf{C}}, F \in \overline{\mathbf{C}}$, the unit $\eta_{F \multimap_{\text{Day}} iG} : F \multimap_{\text{Day}} iG \rightarrow i\tilde{\mathbf{L}}(F \multimap_{\text{Day}} iG)$ is an isomorphism, which gives rise to the following theorem.

Theorem F.2. *For any $F, H \in \overline{\mathbf{C}}$, we have*

$$\tilde{\mathbf{L}}(F \otimes_{\text{Day}} H) \cong^{\tilde{\mathbf{L}}(\eta_{F \otimes_{\text{Day}} H})} \tilde{\mathbf{L}}(i\tilde{\mathbf{L}}F \otimes_{\text{Day}} H).$$

Proof. For any $G \in \tilde{\mathbf{C}}$, we have the following commutative diagram.

$$\begin{array}{ccc}
\tilde{\mathbf{C}}(\tilde{L}(i\tilde{L}F \otimes_{\text{Day}} H), G) & \xrightarrow{\tilde{\mathbf{C}}(\tilde{L}(\eta_F \otimes_{\text{Day}} H), G)} & \tilde{\mathbf{C}}(\tilde{L}(F \otimes_{\text{Day}} H), G) \\
\downarrow \cong & & \downarrow \cong \\
\overline{\mathbf{C}}(i\tilde{L}F \otimes_{\text{Day}} H, iG) & \xrightarrow{\overline{\mathbf{C}}(\eta_F \otimes_{\text{Day}} H, iG)} & \overline{\mathbf{C}}(F \otimes_{\text{Day}} H, iG) \\
\downarrow \cong & & \downarrow \cong \\
\overline{\mathbf{C}}(i\tilde{L}F, H \multimap_{\text{Day}} iG) & \xrightarrow{\overline{\mathbf{C}}(\eta_F, H \multimap_{\text{Day}} iG)} & \overline{\mathbf{C}}(F, H \multimap_{\text{Day}} iG) \\
\overline{\mathbf{C}}(i\tilde{L}F, \eta_{H \multimap_{\text{Day}} iG}) \downarrow & & \downarrow \overline{\mathbf{C}}(F, \eta_{H \multimap_{\text{Day}} iG}) \\
\overline{\mathbf{C}}(i\tilde{L}F, i\tilde{L}(H \multimap_{\text{Day}} iG)) & \xrightarrow{\overline{\mathbf{C}}(\eta_F, i\tilde{L}(H \multimap_{\text{Day}} iG))} & \overline{\mathbf{C}}(F, i\tilde{L}(H \multimap_{\text{Day}} iG)) \\
\downarrow \cong & \swarrow \text{Id} & \\
\tilde{\mathbf{C}}(\tilde{L}F, \tilde{L}(H \multimap_{\text{Day}} iG)) & & \\
\downarrow \cong & & \\
\tilde{\mathbf{C}}(F, i\tilde{L}(H \multimap_{\text{Day}} iG)) & &
\end{array}$$

The top two squares commute by naturality of the adjunctions, the third square commutes by the bifunctoriality of $\overline{\mathbf{C}}(-, -)$ and the bottom triangle commutes by properties of the adjunction $\tilde{L} \dashv i$. \square

With the help of Theorem F.2, one can verify that \tilde{L} is strong monoidal, e.g., for any $F, G \in \overline{\mathbf{C}}$,

$$\tilde{L}F \otimes_{\text{Lam}} \tilde{L}G = \tilde{L}(i\tilde{L}F \otimes_{\text{Day}} i\tilde{L}G) \cong \tilde{L}(F \otimes_{\text{Day}} G).$$

G Proof of Theorem 4.13

We write $\beta : i \circ \tilde{T} \rightarrow \overline{T} \circ i$ to denote the isomorphism $i \circ \tilde{T} \cong \overline{T} \circ i$.

Theorem G.1. *The following diagrams commute.*

1. Suppose $F \in \tilde{\mathbf{C}}$.

$$\begin{array}{ccc}
iF & \xrightarrow{i\eta^{\tilde{T}}} & i\tilde{T}F \\
& \searrow \eta^{\tilde{T}} & \downarrow \beta \\
& & \overline{T}iF
\end{array}$$

2. Suppose $F \in \overline{\mathbf{C}}$.

$$\begin{array}{ccc}
\tilde{L}F & \xrightarrow{\tilde{L}\eta^{\overline{T}}} & \tilde{L}\overline{T}F \\
& \searrow \eta^{\tilde{T}} & \downarrow \rho \\
& & \tilde{T}\tilde{L}F
\end{array}$$

3. Suppose $F \in \tilde{\mathbf{C}}$.

$$\begin{array}{ccc} i\tilde{T}\tilde{T}F & \xrightarrow{\beta} & \overline{T}i\tilde{T}F & \xrightarrow{\overline{T}\beta} & \overline{T}\overline{T}iF \\ \downarrow i\mu^{\tilde{T}} & & & & \downarrow \mu^{\overline{T}} \\ i\tilde{T}F & \xrightarrow{\beta} & \overline{T}iF & & \end{array}$$

4. Suppose $F \in \overline{\mathbf{C}}$.

$$\begin{array}{ccc} \tilde{L}\overline{T}\overline{T}F & \xrightarrow{\rho} & \tilde{T}\tilde{L}\overline{T}F & \xrightarrow{\tilde{T}\rho} & \tilde{T}\tilde{T}\tilde{L}F \\ \downarrow \tilde{L}\mu^{\overline{T}} & & & & \downarrow \mu^{\tilde{T}} \\ \tilde{L}\overline{T}F & \xrightarrow{\rho} & \tilde{T}\tilde{L}F & & \end{array}$$

5. Suppose $F \in \tilde{\mathbf{C}}$.

$$\begin{array}{ccc} \tilde{L}i\tilde{T}F & \xrightarrow{\tilde{L}\beta} & \tilde{L}\overline{T}iF \\ \downarrow \varepsilon & & \downarrow \rho \\ \tilde{T}F & \xleftarrow{\tilde{T}\varepsilon} & \tilde{T}\tilde{L}iF \end{array}$$

6. Suppose $F \in \overline{\mathbf{C}}, G \in \tilde{\mathbf{C}}$.

$$\begin{array}{ccccc} \tilde{L}\overline{T}(i\tilde{T}G \otimes_D F) & \xrightarrow{\tilde{L}\overline{T}(\beta \otimes_D F)} & \tilde{L}\overline{T}(\overline{T}iG \otimes_D F) & \xrightarrow{\tilde{L}\overline{T}\gamma} & \tilde{L}\overline{T}(F \otimes_D \overline{T}iG) \\ \downarrow \rho & & & & \downarrow \rho \\ \tilde{T}\tilde{L}(i\tilde{T}G \otimes_D F) & \xrightarrow{\tilde{T}\tilde{L}\gamma} & \tilde{T}\tilde{L}(F \otimes_D i\tilde{T}G) & \xrightarrow{\tilde{T}\tilde{L}(F \otimes_D \beta)} & \tilde{T}\tilde{L}(F \otimes_D \overline{T}iG) \end{array}$$

7. Suppose $F, G \in \overline{\mathbf{C}}$.

$$\begin{array}{ccccccc} \tilde{L}(F \otimes_D \overline{T}G) & \xrightarrow{\tilde{L}\overline{T}} & \tilde{L}\overline{T}(F \otimes_D G) & \xrightarrow{\rho} & \tilde{T}\tilde{L}(F \otimes_D G) & \xrightarrow{\tilde{T}\tilde{L}(F \otimes_D \eta^{iL})} & \tilde{T}\tilde{L}(F \otimes_D i\tilde{L}G) \\ & & \downarrow \tilde{L}(F \otimes_D \eta^{iL}) & & & & \uparrow \rho \\ \tilde{L}(F \otimes_D i\tilde{L}\overline{T}G) & \xrightarrow{\tilde{L}(F \otimes_D i\rho)} & \tilde{L}(F \otimes_D i\tilde{T}\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes_D \beta)} & \tilde{L}(F \otimes_D \overline{T}i\tilde{L}G) & \xrightarrow{\tilde{L}\overline{T}} & \tilde{L}\overline{T}(F \otimes_D i\tilde{L}G) \end{array}$$

Proof. 1. We have

$$\overline{\mathbf{C}}(iF, \overline{T}iF) = \overline{\mathbf{C}}(iF, \overline{\Delta}U_0iF) \cong \overline{\mathbf{C}}(\overline{U}_0iF, \overline{U}_0iF) \cong \overline{\mathbf{C}}(j\overline{U}'_0F, j\overline{U}'_0F) \cong \tilde{\mathbf{C}}(\overline{U}'_0F, \overline{U}'_0F) \cong \tilde{\mathbf{C}}(F, \tilde{T}F)$$

2. If we unfold the definition of ρ , we have the following diagram.

$$\begin{array}{ccc}
\tilde{L}F & \xrightarrow{\tilde{L}\eta^{\bar{T}}} & \tilde{L}\bar{T}F \\
& \searrow \eta^{\bar{T}} & \downarrow \tilde{L}\bar{T}\eta^{i\bar{L}} \\
& & \tilde{L}\bar{T}i\tilde{L}F \\
& & \downarrow \tilde{L}\beta^{-1} \\
& & \tilde{L}i\tilde{T}\tilde{L}F \\
& & \downarrow \varepsilon \\
& & \tilde{T}\tilde{L}F
\end{array}$$

Note that we have the following commutative diagram, by naturality and (1).

$$\begin{array}{ccccccc}
F & \xrightarrow{\eta^{\bar{T}}} & \bar{T}F & \xrightarrow{\bar{T}\eta^{i\bar{L}}} & \bar{T}i\tilde{L}F & \xrightarrow{\beta^{-1}} & i\tilde{T}\tilde{L}F \\
\downarrow \eta^{\bar{L}} & & \searrow \eta^{\bar{T}} & & \nearrow i\eta^{\bar{T}} & & \\
i\tilde{L}F & & & & & &
\end{array}$$

Therefore we just need to show the following diagram commutes (and indeed it does).

$$\begin{array}{ccccc}
\tilde{L}F & \xrightarrow{\tilde{L}\eta^{i\bar{L}}} & \tilde{L}i\tilde{L}F & \xrightarrow{\tilde{L}i\eta^{\bar{T}}} & \tilde{L}i\tilde{T}\tilde{L}F \\
& \searrow \text{Id} & \downarrow \varepsilon & & \downarrow \varepsilon \\
& & \tilde{L}F & \xrightarrow{\eta^{\bar{T}}} & \tilde{T}\tilde{L}F
\end{array}$$

3. Since each component of μ and β is an identity, the diagram commutes.
4. If we unfold the definition of ρ , we have the following diagram.

$$\begin{array}{ccccc}
\tilde{L}\bar{T}\bar{T}F & \xrightarrow{\tilde{L}\bar{T}\eta^{i\bar{L}}} & \tilde{L}\bar{T}i\tilde{L}\bar{T}F & \xrightarrow{\tilde{L}\beta^{-1}} & \tilde{L}i\tilde{T}\tilde{L}\bar{T}F & \xrightarrow{\varepsilon} & \tilde{T}\tilde{L}\bar{T}F \\
\downarrow \tilde{L}\mu^{\bar{T}} & & & & \downarrow \tilde{L}i\tilde{T}\tilde{L}\bar{T}\eta^{i\bar{L}} & & \downarrow \tilde{T}\tilde{L}\bar{T}\eta^{i\bar{L}} \\
\tilde{L}\bar{T}F & & & & \tilde{L}i\tilde{T}\tilde{L}\bar{T}i\tilde{L}F & \xrightarrow{\varepsilon} & \tilde{T}\tilde{L}\bar{T}i\tilde{L}F \\
\downarrow \tilde{L}\bar{T}\eta^{\bar{L}} & & & & \downarrow \tilde{L}i\tilde{T}\tilde{L}\beta^{-1} & & \downarrow \tilde{T}\tilde{L}\beta^{-1} \\
\tilde{L}\bar{T}i\tilde{L}F & & & & \tilde{L}i\tilde{T}\tilde{L}i\tilde{T}\tilde{L}F & \xrightarrow{\varepsilon} & \tilde{T}\tilde{L}i\tilde{T}\tilde{L}F \\
\downarrow \tilde{L}\beta^{-1} & & & & \downarrow \tilde{L}i\tilde{T}\varepsilon & & \downarrow \tilde{T}\varepsilon \\
\tilde{L}i\tilde{T}\tilde{L}F & \xleftarrow{\tilde{L}i\mu^{\bar{T}}} & \tilde{L}i\tilde{T}\tilde{T}\tilde{L}F & \xrightarrow{\varepsilon} & \tilde{T}\tilde{T}\tilde{L}F & & \\
\downarrow \varepsilon & & & & \nearrow \mu^{\bar{T}} & & \\
\tilde{T}\tilde{L}F & & & & & &
\end{array}$$

All of the squares commute by naturality. We just need to show the left diagram commutes. It

commutes because the following commutes.

$$\begin{array}{ccccc}
 \overline{T}T\overline{F} & \xrightarrow{\overline{T}\eta^{iL}} & \overline{T}i\tilde{L}\overline{T}\overline{F} & \xrightarrow{\beta^{-1}} & i\tilde{T}\tilde{L}\overline{T}\overline{F} \\
 \downarrow \mu & \searrow \overline{T}T\eta^{iL} & \downarrow \overline{T}iL\eta^{iL} & & \downarrow i\tilde{T}\eta^{iL} \\
 \overline{T}\overline{F} & & \overline{T}\overline{T}i\tilde{L}\overline{F} & \xrightarrow{\overline{T}\eta^{iL}} & \overline{T}i\tilde{L}\overline{T}i\tilde{L}\overline{F} \\
 \downarrow \overline{T}\eta^{iL} & \swarrow \mu & \downarrow \overline{T}\beta^{-1} & \searrow \overline{T}iL\beta^{-1} & \downarrow \beta^{-1} \\
 \overline{T}i\tilde{L}\overline{F} & & \overline{T}i\tilde{T}\tilde{L}\overline{F} & \xrightarrow{\overline{T}\eta^{iL}} & \overline{T}i\tilde{L}i\tilde{T}\tilde{L}\overline{F} \\
 \downarrow \beta^{-1} & & \downarrow \beta^{-1} & \searrow \beta^{-1} & \downarrow i\tilde{T}L\beta^{-1} \\
 i\tilde{T}\tilde{L}\overline{F} & \xleftarrow{i\mu} & i\tilde{T}\tilde{T}\tilde{L}\overline{F} & \xleftarrow{i\tilde{\varepsilon}} & i\tilde{T}\tilde{L}i\tilde{T}\tilde{L}\overline{F}
 \end{array}$$

The above diagram commutes, because all the squares commutes by naturality. The bottom left corner diagram commutes by (3). Note that $i\varepsilon \circ \eta^{iL} = \text{Id}$.

5. If we unfold the definition of ρ , we have the following diagram.

$$\begin{array}{ccc}
 \tilde{L}i\tilde{T}\overline{F} & \xrightarrow{\tilde{L}\beta} & \tilde{L}\overline{T}i\overline{F} \\
 \downarrow \tilde{L}\beta & & \downarrow \tilde{L}\overline{T}\eta^{iL} \\
 \tilde{L}\overline{T}i\overline{F} & \xleftarrow{\tilde{L}\overline{T}i\varepsilon} & \tilde{L}\overline{T}i\tilde{L}i\overline{F} \\
 \downarrow \tilde{L}\beta^{-1} & & \downarrow \tilde{L}\beta^{-1} \\
 \tilde{L}i\tilde{T}\overline{F} & \xleftarrow{\tilde{L}i\tilde{T}\varepsilon} & \tilde{L}i\tilde{T}\tilde{L}i\overline{F} \\
 \downarrow \varepsilon & & \downarrow \varepsilon \\
 \tilde{T}\overline{F} & \xleftarrow{\tilde{T}\varepsilon} & \tilde{T}\tilde{L}i\overline{F}
 \end{array}$$

The bottom two squares commute by naturality. The top square commutes because $i\varepsilon \circ \eta^{iL} = \text{Id}$.

6. See the following commutative diagram.

$$\begin{array}{ccccc}
 \tilde{L}\overline{T}(i\tilde{T}G \otimes_D \overline{F}) & \xrightarrow{\tilde{L}\overline{T}(\beta \otimes_D \overline{F})} & \tilde{L}\overline{T}(\overline{T}iG \otimes_D \overline{F}) & \xrightarrow{\tilde{L}\overline{T}\gamma} & \tilde{L}\overline{T}(F \otimes_D \overline{T}iG) \\
 \downarrow \rho & & \downarrow \rho & & \downarrow \rho \\
 & \nearrow \tilde{T}\tilde{L}(\beta \otimes_D \overline{F}) & \tilde{T}\tilde{L}(\overline{T}iG \otimes_D \overline{F}) & \searrow \tilde{T}\tilde{L}\gamma & \\
 \tilde{T}\tilde{L}(i\tilde{T}G \otimes_D \overline{F}) & \xrightarrow{\tilde{T}\tilde{L}\gamma} & \tilde{T}\tilde{L}(F \otimes_D i\tilde{T}G) & \xrightarrow{\tilde{T}\tilde{L}(F \otimes_D \beta)} & \tilde{T}\tilde{L}(F \otimes_D \overline{T}iG)
 \end{array}$$

7. Consider the following diagram.

$$\begin{array}{ccccccc}
 \tilde{L}(F \otimes_D \overline{T}G) & \xrightarrow{\tilde{L}i} & \tilde{L}\overline{T}(F \otimes_D G) & \xrightarrow{\rho} & \tilde{T}\tilde{L}(F \otimes_D G) & \xrightarrow{\tilde{T}\tilde{L}(F \otimes_D \eta^{iL})} & \tilde{T}\tilde{L}(F \otimes_D i\tilde{L}G) \\
 \downarrow \tilde{L}(F \otimes_D \eta^{iL}) & & \downarrow \tilde{L}(F \otimes_D \eta^{iL}) & & \downarrow \tilde{L}(F \otimes_D \eta^{iL}) & & \uparrow \rho \\
 \tilde{L}(F \otimes_D i\tilde{L}\overline{T}G) & \xrightarrow{\tilde{L}(F \otimes_D i\rho)} & \tilde{L}(F \otimes_D i\tilde{T}\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes_D \beta)} & \tilde{L}(F \otimes_D \overline{T}i\tilde{L}G) & \xrightarrow{\tilde{L}i} & \tilde{L}\overline{T}(F \otimes_D i\tilde{L}G)
 \end{array}$$

Note that the very right diagram and the middle square commute by naturality. The left diagram commutes because the following diagram commutes (with ρ unfolded).

$$\begin{array}{ccccc}
\tilde{L}(F \otimes_D \bar{T}G) & \xrightarrow{\tilde{L}(F \otimes \bar{T}\eta^{i\tilde{L}})} & \tilde{L}(F \otimes_D \bar{T}i\tilde{L}G) & & \\
\tilde{L}(F \otimes \eta^{i\tilde{L}}) \downarrow & \searrow^{\tilde{L}(F \otimes \bar{T}\eta^{i\tilde{L}})} & \tilde{L}(F \otimes_D \bar{T}i\tilde{L}G) & \xrightarrow{\text{Id}} & \tilde{L}(F \otimes_D \bar{T}i\tilde{L}G) \\
\tilde{L}(F \otimes_D i\tilde{L}\bar{T}G) & & \tilde{L}(F \otimes_D \bar{T}i\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes_D \beta^{-1})} & \tilde{L}(F \otimes_D i\tilde{T}\tilde{L}G) \\
\tilde{L}(F \otimes i\tilde{L}\bar{T}\eta^{i\tilde{L}}) \downarrow & \searrow^{\tilde{L}(F \otimes \eta^{i\tilde{L}})} & \tilde{L}(F \otimes_D i\tilde{L}\bar{T}\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes \eta^{i\tilde{L}})} & \tilde{L}(F \otimes_D i\tilde{T}\tilde{L}G) \\
\tilde{L}(F \otimes_D i\tilde{L}\bar{T}i\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes_D i\tilde{L}\beta^{-1})} & \tilde{L}(F \otimes_D i\tilde{L}i\tilde{T}\tilde{L}G) & \xrightarrow{\tilde{L}(F \otimes_D i\tilde{L}\epsilon)} & \tilde{L}(F \otimes_D i\tilde{T}\tilde{L}G)
\end{array}$$

$\tilde{L}(F \otimes_D \beta)$

□

Theorem G.2. *The \mathcal{V} -functor \tilde{T} is a commutative strong monad. The strength is given by $\tilde{t}_{F,G} : F \otimes_{\text{Lam}} \tilde{T}G \rightarrow \tilde{T}(F \otimes_{\text{Lam}} G)$ for any $F, G \in \tilde{\mathbf{C}}$.*

Proof. For any $F, G \in \tilde{\mathbf{C}}$, we define $\tilde{t}_{F,G}$ by the following composition.

$$\begin{aligned}
F \otimes_{\text{Lam}} \tilde{T}G &= \tilde{L}(iF \otimes_{\text{Day}} i\tilde{T}G) \\
&\xrightarrow{\beta} \tilde{L}(iF \otimes_{\text{Day}} \bar{T}iG) \xrightarrow{\tilde{t}} \tilde{L}(\bar{T}(iF \otimes_{\text{Day}} iG)) \xrightarrow{\rho} \tilde{T}(\tilde{L}(iF \otimes_{\text{Day}} iG)) = \tilde{T}(F \otimes_{\text{Lam}} G).
\end{aligned}$$

Now we need to show \tilde{T} is a commutative strong monad.

•

$$\begin{array}{ccc}
F \otimes_{\text{Lam}} G & \xrightarrow{\text{Id}_F \otimes_{\text{Lam}} \eta} & F \otimes_{\text{Lam}} \tilde{T}G \\
\downarrow \eta & \swarrow \tilde{t} & \\
\tilde{T}(F \otimes_{\text{Lam}} G) & &
\end{array}$$

The above diagram commutes because the following diagram commutes (by properties of \tilde{t} and Theorem G.1(1)+(2)).

$$\begin{array}{ccccc}
\tilde{L}(iF \otimes_{\text{Day}} iG) & \xrightarrow{\tilde{L}(\text{Id} \otimes_{\text{D}} i\eta^{\tilde{T}})} & \tilde{L}(iF \otimes_{\text{Day}} i\tilde{T}G) & \xrightarrow{\tilde{L}(\text{Id} \otimes_{\text{D}} \beta)} & \tilde{L}(iF \otimes_{\text{Day}} \bar{T}iG) \\
\downarrow \eta^{\tilde{T}} & \searrow^{\tilde{L}(\text{Id} \otimes_{\text{D}} \eta^{\tilde{T}})} & \tilde{L}(iF \otimes_{\text{Day}} \bar{T}iG) & \xrightarrow{\tilde{L}} & \tilde{L}(iF \otimes_{\text{Day}} \bar{T}iG) \\
\tilde{T}\tilde{L}(iF \otimes_{\text{Day}} iG) & \xrightarrow{\tilde{L}\eta^{\tilde{T}}} & \tilde{L}\bar{T}(iF \otimes_{\text{Day}} iG) & \xrightarrow{\rho} & \tilde{L}\bar{T}(iF \otimes_{\text{Day}} iG)
\end{array}$$

•

$$\begin{array}{ccc}
F \otimes_{\text{Lam}} \tilde{T}\tilde{T}G & \xrightarrow{\tilde{t}} & \tilde{T}(F \otimes_{\text{Lam}} \tilde{T}G) \xrightarrow{\tilde{T}\tilde{t}} \tilde{T}\tilde{T}(F \otimes_{\text{Lam}} G) \\
\downarrow F \otimes_{\text{Lam}} \mu^{\tilde{T}} & & \downarrow \mu^{\tilde{T}} \\
F \otimes_{\text{Lam}} \tilde{T}G & \xrightarrow{\tilde{t}} & \tilde{T}(F \otimes_{\text{Lam}} G)
\end{array}$$

The above diagram commutes because the following diagram commutes (all the inner diagrams commute, by naturality, properties of \tilde{t} , and Theorem G.1(3)+(4)).

$$\begin{array}{ccccc}
 \tilde{L}(iF \otimes_D i\tilde{T}\tilde{T}G) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D \beta)} & \tilde{L}(iF \otimes_D \tilde{T}i\tilde{T}G) & \xrightarrow{\tilde{L}\tilde{t}} & \tilde{L}\tilde{T}(iF \otimes_D i\tilde{T}G) \\
 \downarrow \tilde{L}(\text{Id} \otimes_D i\mu^{\tilde{T}}) & & \downarrow \tilde{L}(\text{Id} \otimes_D \tilde{T}\beta) & & \downarrow \rho \\
 \tilde{L}(iF \otimes_D i\tilde{T}G) & & \tilde{L}(iF \otimes_D \tilde{T}\tilde{T}iG) & & \tilde{T}\tilde{L}(iF \otimes_D i\tilde{T}G) \\
 \downarrow \tilde{L}(\text{Id} \otimes_D \beta) & \swarrow \tilde{L}(\text{Id} \otimes_D \mu^{\tilde{T}}) & \downarrow \tilde{L}\tilde{t} & \swarrow \tilde{L}\tilde{T}(\text{Id} \otimes_D \beta) & \downarrow \tilde{T}\tilde{L}(\text{Id} \otimes_D \beta) \\
 \tilde{L}(iF \otimes_D \tilde{T}iG) & & \tilde{L}\tilde{T}(iF \otimes_D \tilde{T}iG) & \xrightarrow{\rho} & \tilde{T}\tilde{L}(iF \otimes_D \tilde{T}iG) \\
 \downarrow \tilde{L}\tilde{t} & & \downarrow \tilde{L}\tilde{T}\tilde{t} & & \downarrow \tilde{T}\tilde{L}\tilde{t} \\
 \tilde{L}\tilde{T}(iF \otimes_D iG) & \xleftarrow{\tilde{L}\mu^{\tilde{T}}} & \tilde{L}\tilde{T}\tilde{T}(iF \otimes_D iG) & \xrightarrow{\rho} & \tilde{T}\tilde{L}\tilde{T}(iF \otimes_D iG) \\
 \downarrow \rho & & & & \downarrow \tilde{T}\rho \\
 \tilde{T}\tilde{L}(iF \otimes_D iG) & \xleftarrow{\mu^{\tilde{T}}} & & & \tilde{T}\tilde{T}\tilde{L}(iF \otimes_D iG)
 \end{array}$$

$$\begin{array}{ccc}
 (F \otimes_{\text{Lam}} G) \otimes_{\text{Lam}} \tilde{T}H & \xrightarrow{\tilde{t}} & \tilde{T}((F \otimes_{\text{Lam}} G) \otimes_{\text{Lam}} H) \\
 \downarrow \alpha & & \downarrow \tilde{T}\alpha \\
 F \otimes_{\text{Lam}} (G \otimes_{\text{Lam}} \tilde{T}H) & \xrightarrow{\text{Id}_F \otimes_{\text{Lam}} \tilde{t}} F \otimes_{\text{Lam}} \tilde{T}(G \otimes_{\text{Lam}} H) & \xrightarrow{\tilde{t}} \tilde{T}(F \otimes_{\text{Lam}} (G \otimes_{\text{Lam}} H))
 \end{array}$$

Again, the above diagram commutes because the following diagram commutes (all of the inner diagrams commute by naturality, properties of \tilde{t} , and Theorem G.1(7)).

$$\begin{array}{ccccc}
 \tilde{L}(i\tilde{L}(iF \otimes_D iG) \otimes_D i\tilde{T}H) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D \beta)} & \tilde{L}(i\tilde{L}(iF \otimes_D iG) \otimes_D \tilde{T}iH) & \xrightarrow{\tilde{L}\tilde{t}} & \tilde{L}\tilde{T}(i\tilde{L}(iF \otimes_D iG) \otimes_D iH) \\
 \downarrow (\tilde{L}(\eta^{\tilde{L}} \otimes_D \text{Id}))^{-1} & & \downarrow (\tilde{L}(\eta^{\tilde{L}} \otimes_D \text{Id}))^{-1} & & \downarrow \rho \\
 \tilde{L}((iF \otimes_D iG) \otimes_D i\tilde{T}H) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D \beta)} & \tilde{L}((iF \otimes_D iG) \otimes_D \tilde{T}iH) & \xrightarrow{\tilde{L}\tilde{t}} & \tilde{L}\tilde{T}((iF \otimes_D iG) \otimes_D iH) \\
 \downarrow \tilde{L}\alpha & & \downarrow \tilde{L}\alpha & & \downarrow \tilde{T}(\tilde{L}(\eta^{\tilde{L}} \otimes_D \text{Id}))^{-1} \\
 \tilde{L}(iF \otimes_D (iG \otimes_D i\tilde{T}H)) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D (\text{Id} \otimes_D \beta))} & \tilde{L}(iF \otimes_D (iG \otimes_D \tilde{T}iH)) & & \tilde{T}\tilde{L}((iF \otimes_D iG) \otimes_D iH) \\
 \downarrow \tilde{L}(\text{Id} \otimes_D \eta^{\tilde{L}}) & \swarrow \tilde{L}(\text{Id} \otimes_D \eta^{\tilde{L}}) & \downarrow \tilde{L}(\text{Id} \otimes_D \tilde{t}) & \swarrow \tilde{L}\tilde{T}\alpha & \downarrow \tilde{T}\tilde{L}\alpha \\
 \tilde{L}(iF \otimes_D i\tilde{L}(iG \otimes_D i\tilde{T}H)) & & \tilde{L}(iF \otimes_D \tilde{T}(iG \otimes_D iH)) & & \tilde{T}\tilde{L}(iF \otimes_D (iG \otimes_D iH)) \\
 \downarrow \tilde{L}(\text{Id} \otimes_D i\tilde{L}(\text{Id} \otimes_D \beta)) & \swarrow \tilde{L}(\text{Id} \otimes_D \tilde{t}) & \downarrow \tilde{L}\tilde{t} & \swarrow \rho & \downarrow \tilde{T}(\tilde{L}(\text{Id} \otimes_D \eta^{\tilde{L}})) \\
 \tilde{L}(iF \otimes_D i\tilde{L}(iG \otimes_D \tilde{T}iH)) & & \tilde{L}\tilde{T}(iF \otimes_D (iG \otimes_D iH)) & & \tilde{T}\tilde{L}(iF \otimes_D i\tilde{L}(iG \otimes_D iH)) \\
 \downarrow \tilde{L}(\text{Id} \otimes_D i\tilde{L}) & \swarrow \tilde{L}(\text{Id} \otimes_D \eta^{\tilde{L}}) & & & \uparrow \rho \\
 \tilde{L}(iF \otimes_D i\tilde{L}\tilde{T}(iG \otimes_D iH)) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D i\rho)} & \tilde{L}(iF \otimes_D i\tilde{T}\tilde{L}(iG \otimes_D iH)) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D \beta)} & \tilde{L}(iF \otimes_D \tilde{T}i\tilde{L}(iG \otimes_D iH)) & \xrightarrow{\tilde{L}\tilde{t}} & \tilde{L}\tilde{T}(iF \otimes_D i\tilde{L}(iG \otimes_D iH))
 \end{array}$$

$$\begin{array}{ccc}
 I \otimes_{\text{Lam}} \tilde{T}F & \xrightarrow{\tilde{t}_{I,F}} & \tilde{T}(I \otimes_{\text{Lam}} F) \\
 & \searrow \lambda_{\tilde{T}F} & \downarrow \tilde{T}\lambda_F \\
 & & \tilde{T}F
 \end{array}$$

The above diagram commutes because the following commutes (all the inner diagrams commute,

by naturality, property of \tilde{i} , and Theorem G.1 (5)).

$$\begin{array}{ccccc}
 \tilde{L}(I \otimes_D i\tilde{T}F) & \xrightarrow{\tilde{L}(\text{Id} \otimes_D \beta)} & \tilde{L}(I \otimes_D \tilde{T}iF) & \xrightarrow{\tilde{L}i} & \tilde{L}\tilde{T}(I \otimes_D iF) \\
 \downarrow \tilde{L}\lambda & & \downarrow \tilde{L}\lambda & \swarrow \tilde{L}\tilde{T}\lambda & \downarrow \rho \\
 \tilde{L}i\tilde{T}F & \xrightarrow{\tilde{L}\beta} & \tilde{L}\tilde{T}iF & & \tilde{T}\tilde{L}(I \otimes_D iF) \\
 \downarrow \varepsilon & & & \searrow \rho & \downarrow \tilde{T}L\lambda \\
 \tilde{T}F & \xleftarrow{\tilde{T}\varepsilon} & \tilde{T}\tilde{L}iF & & \tilde{T}\tilde{L}iF
 \end{array}$$

- Lastly, since $\tilde{\mathbf{C}}$ is symmetric monoidal with $\gamma_{F,G} : F \otimes_{\text{Lam}} G \rightarrow G \otimes_{\text{Lam}} F$. We define the costrength $\sigma_{F,G} := \tilde{T}\gamma_{G,F} \circ \tilde{i}_{G,F} \circ \gamma_{\tilde{T}F,G} : \tilde{T}F \otimes_{\text{Lam}} G \rightarrow \tilde{T}(F \otimes_{\text{Lam}} G)$. We need to show the following diagram commutes.

$$\begin{array}{ccc}
 & \tilde{T}F \otimes_{\text{Lam}} \tilde{T}G & \\
 \swarrow \sigma & & \searrow \tilde{i} \\
 \tilde{T}(F \otimes_{\text{Lam}} \tilde{T}G) & & \tilde{T}(\tilde{T}F \otimes_{\text{Lam}} G) \\
 \downarrow \tilde{T}\tilde{i} & & \downarrow \tilde{T}\sigma \\
 \tilde{T}\tilde{T}(F \otimes_{\text{Lam}} G) & & \tilde{T}\tilde{T}(F \otimes_{\text{Lam}} G) \\
 \swarrow \mu^{\tilde{T}} & & \searrow \mu^{\tilde{T}} \\
 & \tilde{T}(F \otimes_{\text{Lam}} G) &
 \end{array}$$

Again, the above diagram commutes because the following diagram commutes (all the inner diagrams commute, by naturality, properties of \tilde{i} , and Theorem G.1(4)+(6)).

□

H Proof of Theorem 4.17

Theorem H.1. *The \mathcal{V} -category $\tilde{\mathbf{C}}$ is a model for Proto-Quipper with dynamic lifting.*

Proof. We have already shown that $\tilde{\mathbf{C}}$ satisfies conditions **a–e** and **g–h**. In the following we will focus on condition **f**.

- First we need to define a functor $\psi : \mathbf{M} \hookrightarrow V(\tilde{\mathbf{C}})$. We define it as the following composition.

$$\mathbf{M} \cong V(\mathbf{C}) \xrightarrow{V\bar{\gamma}} V(\tilde{\mathbf{C}})$$

The functor ψ is strong monoidal, because $V\bar{\gamma}$ is strong monoidal.

- Next we need to define a functor $\phi : \mathbf{Q} \hookrightarrow Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))$. We write

$$\theta_{F,G} : \tilde{\mathbf{C}}(F, \tilde{T}G) \cong [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}_0'F, \bar{U}_0'G)$$

for any $F, G \in \tilde{\mathbf{C}}$. We also write $\Omega : [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}} \cong [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$. We have

$$Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))(F, G) = \mathcal{V}(1, \tilde{\mathbf{C}}(F, \tilde{T}G))$$

$$\begin{aligned} &\stackrel{\mathcal{V}(1, \theta_{F,G})}{\cong} \mathcal{V}(1, [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}_0'F, \bar{U}_0'G)) \stackrel{\Omega_{\bar{U}_0'F, \bar{U}_0'G}}{\cong} [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(F^0, G^0) \end{aligned}$$

for any $F, G \in \tilde{\mathbf{C}}$. The category $Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))$ is enriched in convex spaces because $[\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}$ is enriched in convex spaces.

Now we define ϕ . On objects, we define

$$\phi(S) = \bar{y}S = \mathbf{C}(-, S).$$

On morphisms, for any $S, U \in \mathbf{Q}$, we define $\phi_{S,U}$ by the following composition of isomorphisms.

$$\mathbf{Q}(S, U) \stackrel{\kappa_{S,U}}{\cong} [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}(\kappa S, \kappa U) = [\mathbf{Set}^{\mathbf{Q}^{\text{op}}}]_{\text{prod}}((\bar{y}S)^0, (\bar{y}U)^0)$$

$$\stackrel{\Omega_{\bar{y}S, \bar{y}U}^{-1}}{\cong} \mathcal{V}(1, [\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}(\bar{U}_0'\bar{y}S, \bar{U}_0'\bar{y}U)) \stackrel{\mathcal{V}(1, \theta_{\bar{y}S, \bar{y}U}^{-1})}{\cong} Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))(\bar{y}S, \bar{y}U)$$

Since the Lambek embedding $\kappa_{S,U}$ preserves convex sum (Theorem 4.16) and the composition $\mathcal{V}(1, \theta_{\bar{y}S, \bar{y}U}^{-1}) \circ \Omega_{\bar{y}S, \bar{y}U}^{-1}$ preserves convex sum, we conclude that ϕ preserves convex sum.

Next we need to show ϕ is strong monoidal. Since κ is strong monoidal, we have the natural

isomorphisms $I \stackrel{e'}{\cong} \kappa I$ and $\kappa S \otimes \kappa U \stackrel{m'_{S,U}}{\cong} \kappa(S \otimes U)$ for any $S, U \in \mathbf{Q}$. Recall that for any $S \in \mathbf{Q}$, $\kappa S = \mathbf{Q}(-, S) = \bar{U}_0'\mathbf{C}(-, S)$ and \bar{U}_0' is strong monoidal. Via the isomorphism Ω (which preserves the monoidal structure) we have the following natural isomorphisms in $[\mathbf{Set}^{\mathbf{C}^{\text{op}}}]_{\text{prod}}$.

$$\Omega^{-1}(e') : \bar{U}_0'\mathbf{C}(-, I) \rightarrow \bar{U}_0'\mathbf{C}(-, I)$$

$$\Omega^{-1}(m'_{S,U}) : \bar{U}_0'(\mathbf{C}(-, S) \otimes \mathbf{C}(-, U)) \rightarrow \bar{U}_0'\mathbf{C}(-, S \otimes U).$$

Now let $m_{S,U} = \theta_{\bar{y}S, \bar{y}U}^{-1}(\Omega^{-1}(m'_{S,U})) : \mathbf{C}(-, S) \otimes \mathbf{C}(-, U) \rightarrow \tilde{T}\mathbf{C}(-, S \otimes U)$ and $e = \theta_{\bar{y}I, \bar{y}I}^{-1}(\Omega^{-1}(e')) : \mathbf{C}(-, I) \rightarrow \tilde{T}\mathbf{C}(-, I)$. It is obvious that e is an isomorphism in $Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))$. The inverse of $m_{S,U}$ is defined as $\theta_{\bar{y}S, \bar{y}U}^{-1}(\Omega^{-1}(m'_{S,U}{}^{-1}))$, which can be verified. We can furthermore show that $m_{S,U}$ is natural and that $e, m_{S,U}$ satisfies the strength diagrams for any $S, U \in \mathbf{Q}$. For example, showing $m_{S,U}$ is natural in $Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}}))$, via the adjunction $\bar{U}_0' \dashv \bar{\Delta}'$, is equivalent to the naturality of $\Omega^{-1}(m'_{S,U})$.

- Lastly, we want to show that the following diagram commutes.

$$\begin{array}{ccc}
\mathbf{M}(S, U) & \xrightarrow{\psi_{S,U}} & \mathcal{V}(1, \tilde{\mathbf{C}}(\bar{y}S, \bar{y}U)) = V(\tilde{\mathbf{C}})(\bar{y}S, \bar{y}U) \\
\downarrow J_{S,U} & & \downarrow E_{S,U} \\
\mathbf{Q}(S, U) & \xleftarrow[\phi_{S,U}^{-1}]{} & \mathcal{V}(1, \tilde{\mathbf{C}}(\bar{y}S, \tilde{T}\bar{y}U)) = Kl_{V\tilde{T}}(V(\tilde{\mathbf{C}})(\bar{y}S, \bar{y}U))
\end{array}$$

Let $f \in \mathbf{M}(S, U)$. We write $(f, J_{S,U}f)$ for the corresponding map in $\mathcal{V}(1, \mathbf{C}(S, U))$. It corresponds to the following map in $\tilde{\mathbf{C}}$ via the enriched Yoneda embedding.

$$\mathbf{C}(-, S) \xrightarrow{\bar{y}(f, J_{S,U}f)} \mathbf{C}(-, U)$$

Applying η_U to the above map, we have the following.

$$\mathbf{C}(-, S) \xrightarrow{\bar{y}(f, J_{S,U}f)} \mathbf{C}(-, U) \xrightarrow{\eta_U} \tilde{T}\mathbf{C}(-, U)$$

So for any $A \in \mathbf{C}$, we have the following.

$$\begin{array}{ccccc}
\mathbf{M}(A, S) & \xrightarrow{\mathbf{M}(A, f)} & \mathbf{M}(A, U) & \xrightarrow{J_{A,U}} & \mathbf{Q}(A, U) \\
\downarrow J_{A,S} & & \downarrow J_{A,U} & & \downarrow \text{Id} \\
\mathbf{Q}(A, S) & \xrightarrow{\mathbf{Q}(A, J_{S,U}f)} & \mathbf{Q}(A, U) & \xrightarrow{\text{Id}} & \mathbf{Q}(A, U)
\end{array}$$

Since $\phi_{S,U}^{-1} = \kappa_{S,U}^{-1} \circ \Omega_{\bar{U}_0 \bar{y}S, \bar{U}_0 \bar{y}U} \circ \mathcal{V}(1, \theta_{\bar{y}S, \bar{y}U})$, we have

$$\begin{aligned}
\phi_{S,U}^{-1}(\eta_U \circ \bar{y}(f, J_{S,U}f)) &= \kappa_{S,U}^{-1} \Omega_{\bar{U}_0 \bar{y}S, \bar{U}_0 \bar{y}U} \mathcal{V}(1, \theta_{\bar{y}S, \bar{y}U})(\eta_U \circ \bar{y}(f, J_{S,U}f)) \\
&= \kappa_{S,U}^{-1} \Omega_{\bar{U}_0 \bar{y}S, \bar{U}_0 \bar{y}U}(\bar{U}_0 \bar{y}(f, J_{S,U}f)) = \kappa_{S,U}^{-1}(\mathbf{Q}(-, J_{S,U}f)) = J_{S,U}f \in \mathbf{Q}(S, U).
\end{aligned}$$

□

Symbolic Synthesis of Clifford Circuits and Beyond

Matthew Amy

School of Computing Science
Simon Fraser University, Burnaby, Canada
meamy@sfu.ca

Owen Bennett-Gibbs

Department of Mathematics and Statistics
McGill University, Montreal, Canada
owen.bennett-gibbs@mail.mcgill.ca

Neil J. Ross

Department of Mathematics and Statistics
Dalhousie University, Halifax, Canada
neil.jr.ross@dal.ca

Path sums are a convenient symbolic formalism for quantum operations with applications to the simulation, optimization, and verification of quantum protocols. Unlike quantum circuits, path sums are not limited to unitary operations, but can express arbitrary linear ones. Two problems, therefore, naturally arise in the study of path sums: the unitarity problem and the extraction problem. The former is the problem of deciding whether a given path sum represents a unitary operator. The latter is the problem of constructing a quantum circuit, given a path sum promised to represent a unitary operator.

In this paper, we show that the unitarity problem is **co-NP**-hard in general, but that it is in **P** when restricted to Clifford path sums. We then provide an algorithm to synthesize a Clifford circuit from a unitary Clifford path sum. The circuits produced by our extraction algorithm are of the form C_1HC_2 , where C_1 and C_2 are Hadamard-free circuits and H is a layer of Hadamard gates. We also provide a heuristic generalization of our extraction algorithm to arbitrary path sums. While this algorithm is not guaranteed to succeed, it often succeeds and typically produces natural looking circuits. Alongside applications to the optimization and decompilation of quantum circuits, we demonstrate the capability of our algorithm by synthesizing the standard quantum Fourier transform directly from a path sum.

1 Introduction

The circuit model is ubiquitous in quantum computing, from hardware assembly code to the high-level description of algorithms. Quantum compilation typically amounts to a series of circuit-to-circuit transformations, lowering a circuit, described programmatically in a *circuit description language* over a high-level gate set, down through a series of progressively restrictive gate sets with more fault-tolerance and hardware constraints. Accordingly, quantum algorithms are frequently described at the level of quantum circuits, plugging together inputs and outputs of large, complicated circuits. An exception is the classical oracles used in many quantum algorithms, which are often described at the level of classical programs or Boolean logic, and then *synthesized* as a high-level quantum circuit.

Despite this, the circuit model is often a less than ideal representation of quantum computations. Semantically, circuits expose little information about a computation to the naked eye, particularly when low-level gate sets like Clifford+ T are used. Likewise, reasoning about quantum circuits is often a difficult affair involving re-write rules derived from circuit *relations*. Complete sets of relations are only known for a small number of low-level non-universal gate sets [28, 3, 22], or higher-level gate sets [12, 21] which result in a large degree of overhead when compiled. Even with complete sets of relations,

simplification of quantum circuits using re-write rules is costly and highly local, yielding results which are typically far from optimal.

Recently, alternative models of quantum computation such as those based on diagrammatic calculi [14, 7], have risen in popularity. These models have seen success in circuit simplification, among other applications, due in part to more effective re-writing methods. However, as most existing quantum computers ultimately run on circuit-like languages, a key component in using such models for circuit transformations is the ability to synthesize or *extract* a circuit back from the representation. This problem has seen a great deal of attention recently in the context of graphical calculi, resulting in methods for the extraction of Clifford and Clifford+ T circuits from ZX diagrams admitting a *generalized flow* [17, 8], as well as theoretical results studying the hardness of this extraction problem [11].

In this paper, we study the problem of synthesizing a unitary circuit given a symbolic expression of a linear operator as a *sum-over-paths* or *path sum* [2]. As any linear operator between 2^n -dimensional Hilbert spaces is representable in this form, we first consider the problem of deciding whether a path sum representations a unitary transformation and show that it is generically **coNP**-hard. Restricting to path sums representing Clifford operators we show that the unitarity problem is in **P**, and that a unitary circuit can be synthesized in time polynomial in the number of qubits. This extraction algorithm produces circuits of the form C_1HC_2 , where C_1 and C_2 are Hadamard-free circuits decomposable as a product of S, X, CZ, and CNOT gates, and H is a layer of Hadamard gates. As a consequence we obtain a simple, constructive proof of the 7-stage Bruhat decomposition of the Clifford group [13, 23].

For non-Clifford operations, we give a heuristic for the unitary synthesis of general sums. While our heuristic does not always produce a circuit even if the path sum represents a unitary operator, it succeeds often in practice and typically returns efficient, natural circuits. Alongside circuit optimization, this heuristic has applications to the *decompilation* of quantum circuits, whereby a circuit over a low-level gate set such as Clifford+ T is re-written over a higher-level gate set such as multiply-controlled Toffoli gates. We further demonstrate the capability of our algorithm by synthesizing the typical quantum Fourier transform circuit directly from its specification as a sum-over-paths.

2 Path sums

We begin by briefly reviewing the theory of *path sums* [2, 30]. A *path sum representation* of a linear operator $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ is an expression for Ψ as a sum indexed by binary variables such as

$$\Psi|\vec{x}\rangle = \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x}, \vec{y})} |f(\vec{x}, \vec{y})\rangle, \quad (1)$$

where $\mathcal{N} \in \mathbb{C} \setminus \{0\}$ is a *normalization factor*, and $P : \mathbb{Z}_2^m \times \mathbb{Z}_2^k \rightarrow \mathbb{R}$ and $f : \mathbb{Z}_2^m \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_2^n$ are real- and Boolean-valued multilinear polynomials, respectively. The path sum in [Equation \(1\)](#) is said to be *amplitude-balanced* because the normalization factor \mathcal{N} is independent of \vec{x} and \vec{y} . We sometimes denote the path sum representation of an operator Ψ by $|\Psi\rangle$.

[Equation \(1\)](#) provides a representation the operator Ψ in the sense that instantiating the binary variables \vec{x} and \vec{y} on both sides of the equality yields a true equation. For this reason, we think of a path sum as a symbolic description of the action of a linear operator on computational basis states.

Example 2.1. The phase gates S and T, as well as the Hadamard gate H, can be represented by path sums as follows, where $\omega = e^{i\pi/4}$:

- $S|x\rangle = i^x|x\rangle$,

- $T|x\rangle = \omega^x|x\rangle$, and
- $H|x\rangle = \frac{1}{\sqrt{2}}\sum_y(-1)^{xy}|y\rangle$.

As path sums involve arithmetic and polynomials over Boolean variables in various rings, it is useful to recall that Boolean algebra can be embedded in any (unital) ring \mathcal{R} using the *lifting* construction defined in [2, Lemma 7.1.6] and reproduced below.

$$\begin{aligned} \bar{0} &= 0_{\mathcal{R}} & \overline{f \wedge g} &= \bar{f} \cdot \bar{g} \\ \bar{1} &= 1_{\mathcal{R}} & \overline{f \oplus g} &= \bar{f} + \bar{g} - (2 \cdot \bar{f} \cdot \bar{g}) \end{aligned}$$

Lifting allows one to use Boolean expressions of variables inside path sums coherently, leading to more natural expressions, as in the following example.

Example 2.2. The gates CNOT and TOF admit the following path sum representations:

- $\text{CNOT}|x_1x_2\rangle = |x_1\rangle|x_1 \oplus x_2\rangle$ and
- $\text{TOF}|x_1x_2x_3\rangle = |x_1\rangle|x_2\rangle|x_1 \oplus (x_2 \cdot x_3)\rangle$.

The sum-over-paths representations of linear operators has been studied extensively in the context of quantum information [15, 9, 27, 24, 19, 10]. Recent work on the connection to graphical calculi [20, 30] has shown that path sums form a universal model for linear operators over 2^n -dimensional Hilbert spaces through direct translations from universal graphical calculi such as the ZH-calculus [7].

Proposition 2.3 (Unversality). *Any linear operator $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ admits a representation as a path sum.*

Example 2.4. Path sums can represent linear operators between spaces of different dimensions. The operators $\eta : \mathbb{C} \rightarrow \mathbb{C}^2 \otimes \mathbb{C}^2$ and $\varepsilon : \mathbb{C}^2 \otimes \mathbb{C}^2 \rightarrow \mathbb{C}$, which act, respectively, as the unit and counit in the category **FdHilb** [30], can be written as the following path sums:

- $\eta| \rangle = \sum_y |yy\rangle$ and
- $\varepsilon|x_1x_2\rangle = \frac{1}{2}\sum_y(-1)^{y(x_1+x_2)}| \rangle$.

When a path sum represents a row or column vector as above, we drop any empty $| \rangle$. A path sum with a single output dimension, representing a \mathbb{C} -valued linear map, is said to be *dimensionless*.

In contrast to graphical calculi, which have a *compositional* structure, path sums are effectively *global* expressions of a linear operator. In other words, the composition (sequential or parallel) of two linear operators is reified into an expression of the form of Equation (1). This is accomplished through the substitution of *free variables* — variables that are not summed over, corresponding to inputs of the operator as elements of the computational basis. We denote the free variables of a path sum $|\Psi\rangle$ by $FV(|\Psi\rangle)$. A path sum with free variables may be thought of as a symbolic state vector in indeterminates $\vec{x} = FV(|\Psi\rangle)$. Hence we use the notation $|\Psi(\vec{x})\rangle$ to denote a path sum expression for the operator Ψ with free variables \vec{x} . We use $|\Psi(x)\rangle$ to denote a path sum with a distinguished free variable x .

Through the lifting operation described above, we can define a notion of substitution for path sums with free variables. In particular, given a free variable x appearing in a path sum we may substitute x with any Boolean expression f in all relevant contexts (the phase or the state).

Definition 2.5 (Substitution). Let $|\Psi(x)\rangle$ be a path sum with free variable x and let f be a Boolean expression. Then the *substitution* of x with f is denoted $|\Psi(f)\rangle$.

Reasoning with local binders and free variables in the path sums requires care to avoid variable capture. For instance, let $|\Psi(x)\rangle = \sum_y |x\rangle$. It can be observed that $|\Psi(x)\rangle$ represents the linear operator $\Psi = 2I$. However, if x is substituted with the free variable y , the \sum_y captures y , giving the path sum $|\Psi(y)\rangle = \sum_y |y\rangle$, representing the vector $|0\rangle + |1\rangle$. We assume that substitution is capture-avoiding unless otherwise noted.

A variable may also be *bound* by summing over its possible values. A bound variable may be locally viewed as a free variable by pulling the summation outside of an expression. Indeed, if $|\Psi(x)\rangle$ is a path sum with free variable x , then $\sum_x |\Psi(x)\rangle$ is a path sum with free variables $FV(|\Psi\rangle) \setminus \{x\}$. We sometimes refer to bound variables as *path* variables.

Example 2.6. Recall that the Hadamard gate can be represented as $H|x\rangle = \frac{1}{\sqrt{2}} \sum_y (-1)^{xy} |y\rangle$. Alternatively, the Hadamard gate can also be written as $H|x\rangle = \sum_y |\Psi(x, y)\rangle$ where $|\Psi(x, y)\rangle = \frac{1}{\sqrt{2}} (-1)^{xy} |y\rangle$ is a path sum in the free variables x and y .

By [Proposition 2.3](#), any linear operator admits a path sum representation. In particular, the composition or tensor product of any two linear operators can also be represented as a path sum. The following proposition gives explicit expression for these constructions in the language of path sums.

Proposition 2.7 (Parallel & Sequential composition). *Let $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ and $\Phi : \mathbb{C}^{2^s} \rightarrow \mathbb{C}^{2^t}$ be two linear operators and let $\Psi|\vec{x}\rangle = \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x}, \vec{y})} |f(\vec{x}, \vec{y})\rangle$ and $\Phi|\vec{w}\rangle = \mathcal{M} \sum_{\vec{z} \in \mathbb{Z}_2^l} e^{2\pi i Q(\vec{w}, \vec{z})} |g(\vec{w}, \vec{z})\rangle$ be expressions of Ψ and Φ as path sums. Then*

$$\begin{aligned} \Psi \otimes \Phi |\vec{x}\rangle |\vec{w}\rangle &= \mathcal{N} \mathcal{M} \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^l} e^{2\pi i [P(\vec{x}, \vec{y}) + Q(\vec{w}, \vec{z})]} |f(\vec{x}, \vec{y})\rangle \otimes |g(\vec{w}, \vec{z})\rangle \\ \Phi \circ \Psi |\vec{x}\rangle &= \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x}, \vec{y})} |\Phi(f(\vec{x}, \vec{y}))\rangle \end{aligned}$$

as path sums, where the latter is well-formed if and only if $s = n$.

The parallel and sequential composition of path sums provides a method to compute a symbolic expression for a circuit over a set of basic gates with known path sums. Moreover, for typical gate sets of interest, this representation has size polynomial in the size of the circuit. We give one such result below [[2](#), Corollary 2.15] for the class of circuits which will be most relevant for the purposes of this paper.

Proposition 2.8 (Efficiency for Clifford+T). *Any circuit over Clifford+T gates of volume V can be expressed as a path sum which has size polynomial in V and can be computed in time polynomial in V .*

Equational reasoning A major utility of the path sum representation [[2](#)] comes from the ability to perform *equational reasoning*. Complete equational theories of Clifford unitaries [[2](#)] and more general stabilizer operations [[30](#)] have previously been developed. We reformulate these theories here using locally free variables to simplify their presentation.

Proposition 2.9. *Let Ψ be a path sum such that $y \notin FV(\Psi)$ and let f be a Boolean expression such that $x, y \notin FV(f)$. Then the following equations hold.*

$$\sum_y |\Psi\rangle = 2|\Psi\rangle \tag{2}$$

$$\sum_{x,y} (-1)^{y(x+f)} |\Psi(x)\rangle = 2|\Psi(f)\rangle \tag{3}$$

$$\sum_y i^y (-1)^{yf} |\Psi\rangle = \omega \sqrt{2} (-i)^f |\Psi\rangle \tag{4}$$

$$\sum_y |\Psi(y)\rangle = \sum_y |\Psi(y+f)\rangle \tag{5}$$

Equations (2), (3) and (4) are restatements of [2, Proposition 3.1]. Equation (5) is a generalization of the (ket) rule given in [30, Figure 3] and can be derived from Equation (3) as follows:

$$\begin{aligned}
 \sum_y |\Psi(y)\rangle &= \sum_y \left[\frac{1}{2} \sum_{x,z} (-1)^{z(x+(y+f))} |\Psi(y)\rangle \right] && \text{By Equation (3) where } x, z \notin FV(f) \cup FV(\Psi) \\
 &= \sum_x \left[\frac{1}{2} \sum_{y,z} (-1)^{z(y+(x+f))} |\Psi(y)\rangle \right] && \text{Basic arithmetic} \\
 &= \sum_x |\Psi(x+f)\rangle && \text{By Equation (3)} \\
 &= \sum_y |\Psi(y+f)\rangle && \text{Since } y \notin FV(f) \cup FV(\Psi)
 \end{aligned}$$

The first equality above uses the instance $\sum_{x,z} (-1)^{z(x+f')} |\Psi(y)\rangle = 2 |\Psi(y)\rangle$ of Equation (3), where $f' = y + f$ and $|\Psi(y)\rangle$ is viewed as a path sum with zero occurrences of the free variable x .

Example 2.10. Consider the dimensionless path sum $\frac{1}{\sqrt{2}} \sum_y t^y$. By Equation (4) it follows that $\frac{1}{\sqrt{2}} \sum_y t^y = \omega$. Hence, Equation (4) symbolically encodes the fact that $\omega = \frac{1+i}{\sqrt{2}}$.

Relationship to post-selected circuits As with the ZX-calculus and variants, path sum expressions correspond naturally to circuits with ancillas and postselection. Given a path sum expression of a linear operator $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ of the form of Equation (1), a circuit implementing Ψ up to a constant scalar factor can be achieved through postselection as follows. First, prepare k ancillary qubits in the state $\frac{1}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} |\vec{y}\rangle$ by applying Hadamard gates to the $|0\rangle^{\otimes k}$ state. The symbolic state is then prepared up to some garbage $|g(\vec{x}, \vec{y})\rangle$ via the unitary transformation

$$\Psi_{PS} : |\vec{x}\rangle \otimes |\vec{y}\rangle \mapsto e^{2\pi i P(\vec{x}, \vec{y})} |f(\vec{x}, \vec{y})\rangle \otimes |g(\vec{x}, \vec{y})\rangle.$$

Finally, the garbage is discarded by postselecting $\mathbb{H}^{\otimes m+k-n} |g(\vec{x}, \vec{y})\rangle = \mathcal{N}' \sum_{\vec{z}} (-1)^{\vec{z} \cdot g(\vec{x}, \vec{y})} |\vec{z}\rangle$ on $\vec{z} = \vec{0}$.

Since postselected quantum circuits are believed to be strictly more powerful than non-postselected circuits [1], the rest of this paper focuses on the question of synthesizing unitary circuits for path sums representing unitary transformations, up to normalization.

3 Unitarity testing

We are interested in the synthesis of *unitary* quantum circuits implementing a path sum. As a path sum may represent an arbitrary linear operator, a natural question to ask is whether a given path sum represents a unitary transformation, and hence can be extracted to a unitary circuit. We call this the *unitarity testing* problem for path sums and formulate it as a decision problem below.

Definition 3.1 (UNITARY). *UNITARY* is the set of path-sums $|\Psi\rangle$ where Ψ is a unitary transformation.

The unitarity problem is clearly decidable since we can always explicitly compute a matrix representation of Ψ from a path sum $|\Psi\rangle$. However, since the size of the corresponding matrix is exponential in n , this solution is not efficient. As we show in this section, one should not hope for an efficient solution in general.

Recall that the complexity class **co-NP** consists of the decision problems whose complement belongs to **NP**, and hence is widely believed to be intractable. A canonical complete problem for **co-NP** is the

tautology problem, recognizing the set of propositional formulas over the connectives $\{\neg, \wedge, \vee\}$ which are satisfied by every variable assignment. We view a propositional formula in n distinct free variables as a function $\mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$, using the standard interpretation $\neg x := 1 + x$, $x \wedge y := xy$ and $x \vee y := x + y - xy$. The application of φ to some $\vec{x} \in \mathbb{Z}_2^n$ is denoted by $\varphi(\vec{x})$.

Definition 3.2 (Tautology). A propositional formula φ in n variables is a *tautology* if $\varphi(\vec{x}) = 1$ for every $\vec{x} \in \mathbb{Z}_2^n$, written $\varphi \equiv 1$.

Definition 3.3 (TAUT). *TAUT* is the set of all propositional formulas that are tautologies.

Theorem 3.4 (Karp's 21 NP-complete problems). *The TAUT problem is co-NP-complete.*

To reduce TAUT to UNITARY, our goal is to encode a propositional formula φ as a path sum representing the linear operator $\Phi|\vec{x}\rangle = \varphi(\vec{x})|\vec{x}\rangle$ which is the identity if $\varphi \equiv 1$, and non-unitary otherwise. To do so we establish an encoding of φ as a dimensionless path sum of the form $\varphi(\vec{x}) = \mathcal{N} \sum_{\vec{y}} (-1)^{P(\vec{x}, \vec{y})}$, where P is a multilinear Boolean polynomial.

It can readily be observed that $x = 2^{-1} \sum_{y \in \mathbb{Z}_2} (-1)^{y(1+x)}$ for any $x \in \mathbb{Z}_2$. This gives an immediate encoding of any propositional formula in a path sum by extending the lifting discussed in Section 2 to propositional negation and disjunction via the equations $\overline{\neg\varphi} = 1 - \overline{\varphi}$, $\overline{\varphi \vee \psi} = \overline{\varphi} + \overline{\psi} - \overline{\varphi} \cdot \overline{\psi}$, and then setting

$$\varphi(\vec{x}) = 2^{-1} \sum_y (-1)^{y(1+\overline{\varphi(\vec{x})})}.$$

However, the lifting of a propositional formula φ may generally have size exponential in the size of φ . To obtain a polynomial size encoding we rely on the *Tseytin transformation* [29].

Given two propositional formulas φ and ψ , we write $\varphi \leftrightarrow \psi$ for the logical equality of φ and ψ , which is satisfied by an assignment \vec{x} if and only if $\varphi(\vec{x}) = \psi(\vec{x})$. The Tseytin transformation takes a propositional formula φ with k distinct subterms and returns an equisatisfiable conjunction of at most $O(k)$ constant-depth formulas by assigning a fresh propositional variable to the value of each subterm. For instance, given a propositional formula $\varphi = x_1 \wedge (x_2 \vee \neg x_3)$, the Tseytin transformation of φ , denoted $\mathcal{T}(\varphi)$, is

$$\mathcal{T}(\varphi) = z_1 \wedge (z_1 \leftrightarrow x_1 \wedge z_2) \wedge (z_2 \leftrightarrow x_2 \vee z_3) \wedge (z_3 \leftrightarrow \neg x_3).$$

Note that $FV(\varphi) \subseteq FV(\mathcal{T}(\varphi))$ and that the satisfying assignments of φ and $\mathcal{T}(\varphi)$ are in a 1-to-1 correspondence and agree on $FV(\varphi)$.

Given a propositional formula φ , we can encode the Tseytin transformation $\mathcal{T}(\varphi)$ of φ in a dimensionless sum over the free variables of φ using the following encoding of logical equality

$$(\varphi \leftrightarrow \psi)(\vec{x}) = \sum_y (-1)^{y\overline{\varphi(\vec{x})} + y\overline{\psi(\vec{x})}}.$$

Note that for a clause of the Tseytin transformation $z \leftrightarrow \varphi$ where φ has constant depth, $\overline{\varphi}$ has constant size. If we denote the clauses of $\mathcal{T}(\varphi)$ by $z_1 \leftrightarrow c_1, \dots, z_k \leftrightarrow c_k$, we may encode $\mathcal{T}(\varphi)$ as a polynomial-size sum by taking the product of each clause and distributing over the summations:

$$\begin{aligned} \mathcal{T}(\varphi)(\vec{x}, \vec{z}) &= z_1 \prod_i (z_i \leftrightarrow c_i)(\vec{x}) = 2^{-1} \sum_y (-1)^{y(1+z_1)} \prod_i 2^{-1} \sum_{y_i} (-1)^{y_i(z_i + \overline{c_i}(\vec{x}))} \\ &= 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))}. \end{aligned}$$

Finally, since the satisfying assignments of φ and $\mathcal{T}(\varphi)$ are in a 1-to-1 correspondence, we see that

$$\varphi(\vec{x}) = \sum_{\vec{z}} \mathcal{T}(\varphi)(\vec{x}, \vec{z}) = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))}.$$

Proposition 3.5. Let φ be a propositional formula in n variables and let $\mathcal{T}(\varphi) = z_1 \wedge (\bigwedge_{i=1}^k z_i \leftrightarrow c_i)$. Then for any $\vec{x} \in \mathbb{Z}_2^n$,

$$\varphi(\vec{x}) = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i (z_i + \bar{c}_i(\vec{x}))}$$

where the sum on the right hand side has size polynomial in k .

Remark 3.6. The encoding of φ in **Proposition 3.5** is interesting because it gives a polynomial-size expression in the same variables as φ . This is in contrast to the propositional Tseytin transformation which gives an encoding over a superset of free variables, and hence only remains equi-satisfiable. In particular, $\mathcal{T}(\cdot)$ does not preserve tautologies, whilst our encoding does when viewed as a $\{0, 1\}$ -valued function.

Given the encoding of φ above, we can now prove **co-NP**-hardness of the unitarity testing problem by a reduction from TAUT.

Theorem 3.7. The unitarity testing problem is **co-NP**-hard

Proof. By many-one reduction from TAUT to UNITARY. Given a propositional formula φ in n variables, define $\Psi : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ to be the linear operator given by $\Psi|\vec{x}\rangle = \varphi(\vec{x})|\vec{x}\rangle$. By **Proposition 3.5**, Ψ admits the following representation as a path sum

$$\Psi|\vec{x}\rangle = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i (z_i + \bar{c}_i(\vec{x}))} |\vec{x}\rangle$$

which has size polynomial in the number of subterms of φ . Hence all that remains is to show that Ψ is unitary if and only if φ is a tautology. It suffices to observe that, for any $\vec{x} \in \mathbb{Z}_2^n$, $\Psi|\vec{x}\rangle = \varphi(\vec{x})|\vec{x}\rangle = |\vec{x}\rangle$ if $\varphi(\vec{x}) = 1$ and $\vec{x} \in \mathbb{Z}_2^n$, $\Psi|\vec{x}\rangle = \varphi(\vec{x})|\vec{x}\rangle = 0$ otherwise. In particular, if $\varphi(\vec{x}) = 1$ for all $\vec{x} \in \mathbb{Z}_2^n$, then $\Psi = I_n$. Otherwise, there exists $\vec{x} \in \mathbb{Z}_2^n$ such that $\Psi|\vec{x}\rangle = 0$ and hence Ψ is non-unitary, as required. \square

4 Clifford synthesis

In this section we look at the problems of synthesis and unitarity testing in the restricted case of Clifford operations. The synthesis of Clifford circuits has applications both to randomized benchmarking, as well as to the design and analysis of error correction circuits. We first review the definition of the Clifford group.

Definition 4.1 (Pauli group). The n -qubit Pauli group \mathcal{P}_n is the group of n -fold tensor products of Pauli operators $\{I, X, Y, Z\}$.

Definition 4.2 (Clifford group). The n -qubit Clifford group is the group $\mathcal{C}_n = \{U \in U_{2^n} \mid U \mathcal{P}_n U^\dagger = \mathcal{P}_n\}$.

A well-known consequence of the Gottesman-Knill theorem is the fact that, up to global phases, the Clifford group is generated by $\{H, S, \text{CNOT}\}$. We may use this fact to give a convenient path sum representation of Clifford operations.

Proposition 4.3. Every Clifford operator $\Psi : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^n}$ can be written as a sum of the form

$$\Psi|\vec{x}\rangle = \frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x}, \vec{y})} (-1)^{Q(\vec{x}, \vec{y})} |f(\vec{x}, \vec{y})\rangle \quad (6)$$

where $\omega = e^{2\pi i/8}$, $l \in \mathbb{Z}_8$, $L : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_4$ is linear, $Q : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_2$ is pure quadratic, and $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_2$ is affine.

Proof. As the Clifford group generators CNOT, S, and H can be written in the form of Equation (6), it only remains to show that the composition of two such sums can be written in the form of Equation (6). It suffices to note that substitution of a variable with an affine Boolean expression does not increase the degree of Q or f , while substitution in L produces a quadratic form with degree 2 terms divisible by 2. \square

We call an expression of the form of Equation (6) a *Clifford path sum*. In the context of *stabilizer states* — states $|\psi\rangle = C|\vec{0}\rangle$ for some $C \in \mathcal{C}_n$ — this representation is well-known by various names, including the *quadratic form expansion* [10] and the *affine representation* [16, 26]. Through the inclusion of free parameters we can represent stabilizer states, Clifford unitaries, or Clifford circuits with ancillas in this form.

We next define a *normal form* for Clifford path sums which will be useful for circuit synthesis.

Definition 4.4 (Normal form). A Clifford path sum for Ψ is in *normal form* if, up to a reordering of qubits,

$$\Psi|\vec{x}\rangle = \frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x}, \vec{y})} (-1)^{Q(\vec{x}, \vec{y})} |\vec{y}\rangle \otimes |f(\vec{x}, \vec{y})\rangle, \quad (7)$$

where $l \in \mathbb{Z}_8$, $L : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_4$ is linear, $Q : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_2$ is pure quadratic, and $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \rightarrow \mathbb{Z}_2$ is affine.

The normal form above corresponds to re-writing the sum over a minimal set of vectors spanning the affine subspace of \mathbb{Z}_2^n given by $\{f(\vec{x}, \vec{y}) \mid \vec{y} \in \mathbb{Z}_2^k\}$. The following proposition states that Equations (2), (3), (4) and (5) suffice to re-write a unitary Clifford path sum into normal form.

Proposition 4.5. *Let $|\Psi\rangle$ be a Clifford path sum. There exists a re-writing procedure which will terminate with $|\Psi\rangle$ in normal form if Ψ is unitary and runs in time polynomial in the size of $|\Psi\rangle$.*

Proof. For each path variable y_i , if there exists j such that $f_j(\vec{x}, \vec{y}) = y_i \oplus f'_j(\vec{x}, \vec{y})$, Equation (5) can be applied to substitute y_i with $y_i \oplus f'_j(\vec{x}, \vec{y})$. If no such j exists, either Ψ is unitary and one of Equations (2), (3) and (4) necessarily applies to eliminate y_i [2], or no rule applies and Ψ is non-unitary. \square

Remark 4.6. Proposition 4.5 also holds for non-square $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ with $m \leq n$ so long as Ψ is an *isometry* — that is, if Ψ corresponds to a Clifford circuit with some ancillas or fixed inputs.

Corollary 4.7. *The unitarity testing problem for Clifford path sums is in P.*

Proof. Given a Clifford path sum $|\Psi\rangle$, we can construct a path sum representation of Ψ^\dagger efficiently using η , ε , and negating L . Then by Proposition 4.5 we can normalize the path sum representations of $\Psi\Psi^\dagger$ and $\Psi^\dagger\Psi$ each in polynomial time. If either fails to produce a normal form, then one of $\Psi\Psi^\dagger$ or $\Psi^\dagger\Psi$ is non-unitary and hence Ψ is non-unitary. If both are reduced to normal form, it suffices to observe that we can check whether a Clifford path sum in normal form represents the identity transformation in polynomial time. \square

It is now straightforward to compute a circuit implementing a (unitary) Clifford path sum from its normalized form. If we decompose f , L , and Q as $f(\vec{x}, \vec{y}) = f_x(\vec{x}) + f_y(\vec{y}) + b$, $L(\vec{x}, \vec{y}) = L_x(\vec{x}) + L_y(\vec{y})$, and

$Q(\vec{x}, \vec{y}) = Q_x(\vec{x}) + Q_y(\vec{y}) + \sum_{i=1}^k y_i R_i(\vec{x})$ then the normal form can be written as the following sequence of transformations:

$$\begin{aligned} |\vec{x}\rangle &\mapsto \omega^l i^{L_x(\vec{x})} (-1)^{Q_x(\vec{x})} |\vec{x}\rangle \\ |\vec{x}\rangle &\mapsto |R(\vec{x})\rangle |f_x(\vec{x})\rangle \\ |R(\vec{x})\rangle |f_x(\vec{x})\rangle &\mapsto \frac{1}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{\sum_i y_i R_i(\vec{x})} |\vec{y}\rangle |f_x(\vec{x})\rangle \\ |\vec{y}\rangle |f_x(\vec{x})\rangle &\mapsto |\vec{y}\rangle |f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}\rangle \\ |\vec{y}\rangle |f(\vec{x}, \vec{y})\rangle &\mapsto i^{L_y(\vec{y})} (-1)^{Q_y(\vec{y})} |\vec{y}\rangle |f(\vec{x}, \vec{y})\rangle \end{aligned}$$

This gives a circuit of the form $U(\mathbb{H}^{\otimes k} \otimes \mathbb{I}_{n-k})V$ where U and V are generalized permutations contained in the Clifford group. Moreover, $|\vec{x}\rangle \mapsto |R(\vec{x})\rangle |f_x(\vec{x})\rangle$ is the only operator which may be non-unitary, and in particular is unitary if and only if Ψ is. Note that the unitarity of Ψ hence forces $\{R_i\}$ to be linearly independent and for R_i to be non-zero. This is summarized in [Algorithm 1](#).

Algorithm 1 Clifford synthesis algorithm

1. Normalize $|\Psi\rangle$ in the form $\frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x}, \vec{y})} (-1)^{Q(\vec{x}, \vec{y})} |\vec{y}\rangle \otimes |f(\vec{x}, \vec{y})\rangle$ up to qubit reordering.
 2. Decompose f , L , and Q as $f(\vec{x}, \vec{y}) = f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}$, $L(\vec{x}, \vec{y}) = L_x(\vec{x}) + L_y(\vec{y})$ and $Q(\vec{x}, \vec{y}) = Q_x(\vec{x}) + Q_y(\vec{y}) + \sum_i y_i R_i(\vec{x})$ where each R_i is linear.
 3. Synthesize circuits for the following linear transformations:
 - $D|\vec{x}\rangle = i^{L_x(\vec{x})} (-1)^{Q_x(\vec{x})} |\vec{x}\rangle$
 - $U|\vec{x}\rangle = |R(\vec{x})\rangle |f_x(\vec{x})\rangle$
 - $V|\vec{y}\rangle |f_x(\vec{x})\rangle = |\vec{y}\rangle |f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}\rangle$
 - $P|\vec{y}\rangle |f(\vec{x}, \vec{y})\rangle = i^{L_y(\vec{y})} (-1)^{Q_y(\vec{y})} |\vec{y}\rangle |f(\vec{x}, \vec{y})\rangle$
 4. Return $\omega^l PV(\mathbb{H}^{\otimes k} \otimes \mathbb{I}_{n-k})UD$ with qubits appropriately reordered.
-

Theorem 4.8. *Let $\Psi : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^n}$ be expressed as a Clifford path sum. If Ψ is unitary, then [Algorithm 1](#) produces a circuit over $\{\omega, \text{CNOT}, \text{X}, \text{CZ}, \text{S}, \text{H}\}$ implementing Ψ in time polynomial in the size of the expression. Moreover, this circuit can be written up to global phase as an 8-stage circuit of the form*

$$S \cdot \text{CZ} \cdot \text{CNOT} \cdot \text{H} \cdot \text{CNOT} \cdot \text{X} \cdot \text{CZ} \cdot \text{S}$$

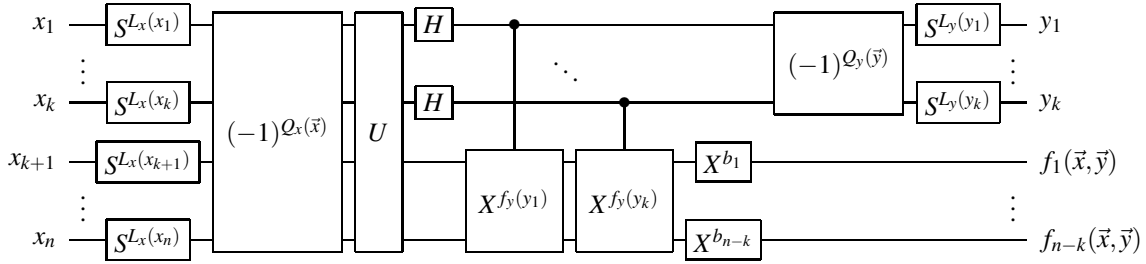
Proof. That $\Psi = \omega^l PV(\mathbb{H}^{\otimes k} \otimes \mathbb{I}_{n-k})UD$ follows by an easy calculation.

By [Proposition 4.5](#), Ψ can be written up to a permutation of qubits in normal form in polynomial time. Since L_x and L_y are linear, and Q_x and Q_y are pure quadratic, D and P can each be synthesized using a single stage each of S and CZ gates — one S^m gate for each non-zero term of $L_{\{x,y\}}$ and one CZ gate for each non-zero term of $Q_{\{x,y\}}$. Likewise, since $f_y(\vec{y})$ is linear, V can be synthesized in time polynomial in n using a single stage each of CNOT and X gates — one gate for each non-zero entry of $f_y(\vec{y})$ and \vec{b} . Finally, $U|\vec{x}\rangle = |R(\vec{x})\rangle \otimes |f_x(\vec{x})\rangle$ can be synthesized over $\{\text{CNOT}\}$ in polynomial time using Gaussian elimination if and only if U is invertible. Moreover, since

$$U = \omega^{-l} (\mathbb{H}^{\otimes k} \otimes \mathbb{I}_{n-k}) V^\dagger P^\dagger \Psi D^\dagger,$$

it follows that U is invertible if and only if Ψ is unitary. □

We give a diagrammatic presentation of [Theorem 4.8](#) showing the circuit schematically below.



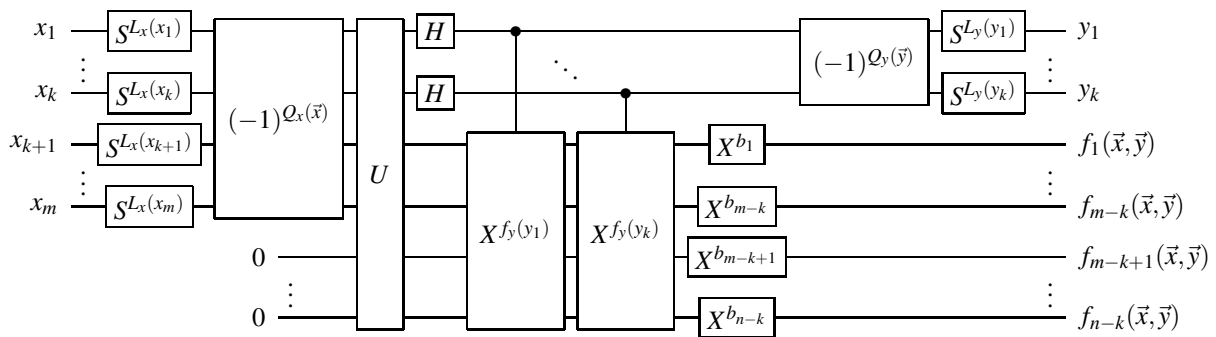
Discussion In [23] a 7 stage decomposition of the Clifford group of the form $S \cdot CZ \cdot C \cdot H \cdot C \cdot CZ \cdot S$ was given, where C is circuit implementing an affine permutation. As affine permutations require both CNOT and X gates to implement without ancillas — and moreover X can not be written in the form $S \cdot CZ \cdot CNOT \cdot H \cdot CNOT \cdot CZ \cdot S$ — our projective decomposition reduces the equivalent 9-stage projective decomposition of [23] to 8 stages.

It can also be observed that with a minor modification, [Algorithm 1](#) suffices to synthesize Clifford circuits with ancillas, including circuits for preparing stabilizer states. In particular, if $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ is an isometric Clifford path sum with $m \leq n$, the only modification needed is in the synthesis of $U|\vec{x}\rangle = |R(\vec{x})\rangle \otimes |f_x(\vec{x})\rangle$. If indeed $m \leq n$, then a Clifford circuit with ancillas exists and can be synthesized if and only if $\{R_i\} \cup \{(f_x)_i\}$ contains m linearly independent (row) vectors. This produces a 5-stage circuit of the form $H \cdot CNOT \cdot X \cdot CZ \cdot S$ for the preparation of an arbitrary stabilizer state up to global phase. This stabilizer state decomposition was previously given in [26].

Corollary 4.9. A Clifford normal form $|\vec{x}\rangle \mapsto \frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x}, \vec{y})} (-1)^{Q(\vec{x}, \vec{y})} |\vec{y}\rangle \otimes |f(\vec{x}, \vec{y})\rangle$ from m to $n \geq m$ qubits can be implemented with Clifford gates and ancillas initialized in the $|0\rangle$ state if and only if

$$\text{rank}(\{R_i\} \cup \{(f_x)_i\}) = m.$$

The circuit is shown schematically below:



5 General synthesis

We now consider the more challenging problem of synthesizing a unitary circuit from an arbitrary path sum. Our method attempts to iteratively reduce the number of summed variables in a path sum by alternately applying generalized permutations and Hadamard gates to the symbolic state. Recall that a

(unitary) *generalized permutation* is a permutation matrix whose nonzero entries are elements of $\mathbb{T} = \{z \in \mathbb{C} \mid |z| = 1\}$. The generalized permutations are generated by the gates

$$\Lambda_k(X) : |\vec{x}\rangle |y\rangle \mapsto |\vec{x}\rangle |y \oplus \prod_i x_i\rangle \quad \text{and} \quad \Lambda_k(R_Z(\theta)) : |\vec{x}\rangle \mapsto e^{2\pi i \theta \prod_i x_i} |\vec{x}\rangle.$$

Together with the Hadamard gate this forms an exactly universal set as it includes every single-qubit unitary along with the CNOT gate.

The following fact forms the basis of our synthesis algorithm. It gives a condition on a path sum which allows a summed variable to be eliminated by multiplication with a Hadamard gate.

Proposition 5.1. *Let $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ be a linear operator where*

$$\Psi |\vec{x}\rangle = \mathcal{N} \sum_{z \in \mathbb{Z}_2} \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{zQ(\vec{x}, \vec{y})} e^{2\pi i P(\vec{x}, \vec{y})} |z\rangle \otimes |f(\vec{x}, \vec{y})\rangle.$$

Then $(H \otimes I_{n-1})\Psi |\vec{x}\rangle = \sqrt{2} \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x}, \vec{y})} |Q(\vec{x}, \vec{y})\rangle \otimes |f(\vec{x}, \vec{y})\rangle$.

Proof. By Equation (3), since $(H \otimes I_{n-1})\Psi |\vec{x}\rangle = \frac{1}{\sqrt{2}} \mathcal{N} \sum_z \sum_{\vec{y}} (-1)^{zQ(\vec{x}, \vec{y}) + zz'} e^{2\pi i P(\vec{x}, \vec{y})} |z'\rangle \otimes |f(\vec{x}, \vec{y})\rangle$ □

Note that Proposition 5.1 is essentially an inversion of the H gate, $H^\dagger : \sum_z (-1)^{xz} |z\rangle \mapsto |x\rangle$. We say that a variable z is *reducible* in the path sum $|\Psi\rangle$ if $|\Psi\rangle$ is in the form of Proposition 5.1.

Definition 5.2 (Reducible). A variable z is *reducible* in an expression of $\Psi : \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n}$ if, up to qubit reordering, it has the form

$$\Psi |\vec{x}\rangle = \mathcal{N} \sum_{z \in \mathbb{Z}_2} \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{zQ(\vec{x}, \vec{y})} e^{2\pi i P(\vec{x}, \vec{y})} |z\rangle \otimes |f(\vec{x}, \vec{y})\rangle.$$

At a high level, our algorithm proceeds by attempting to synthesize a generalized permutation which will leave some path variable reducible. If the process terminates with remaining summed variables, or an unsynthesizeable ground term $e^{2\pi i P(\vec{x})} |f(\vec{x})\rangle$, the algorithm fails to produce a circuit. Algorithm 2 gives the high-level algorithm in pseudo-code.

Algorithm 2 General path sum synthesis algorithm

1. Set C to the empty circuit and normalize $|\Psi\rangle$ using Equations (2), (3) and (4)
 2. For each remaining path variable y in $|\Psi\rangle$
 - (a) If there exists a generalized permutation U such that y is reducible in $U^\dagger |\Psi\rangle$,
 - i. $|\Psi\rangle \leftarrow (H \otimes I_{n-1})U^\dagger |\Psi\rangle$
 - ii. Append $U(H \otimes I_{n-1})$ to C
 - iii. Go to step 1
 3. If path variables remain or Ψ is non-unitary, fail. Otherwise, append Ψ^\dagger to C and return C .
-

Finding such a generalized permutation is highly non-trivial. Our method applies a series of symbolic simplifications, corresponding to $\Lambda_k(X)$ and $\Lambda_k(R_Z(\theta))$ gates, to the term $e^{2\pi i P(\vec{x}, \vec{y})} |f(\vec{x}, \vec{y})\rangle$. If these simplifications fail to leave any variable reducible, we fall back to an exponential-time procedure aimed at computing a substitution of the form in Equation (5) which will make some variable reducible. These heuristics are described in Appendix A.

6 Experiments

To test the performance and utility of our synthesis methods, we implemented [Algorithms 1](#) and [2](#) in the FEYNMAN¹ software package. In this section, we briefly detail our investigations into applications to the optimization and decompilation of circuits, and to specification-based synthesis. All Clifford circuits synthesized have been checked for correctness using the method of [\[2\]](#). For [Algorithm 2](#), as the method of [\[2\]](#) often fails to verify circuits extracted using [Equation \(5\)](#), we instead validated correctness of our synthesis procedure by verifying the individual synthesis steps each on 1000 unitary path sums extracted from randomly generated Clifford+ T circuits. [Table 1](#) gives some statistics from experiments re-synthesizing random circuits. Random circuits were generated by selecting a given number of gates on a given number of qubits, taken from the $\{\text{CNOT}, \text{H}, \text{S}\}$ and $\{\text{CNOT}, \text{H}, \text{T}\}$ gate sets for Clifford and Clifford+ T , respectively.

	n	# gates	# circuits	avg. time (s)	avg. change (+/-)	success
Clifford	20	500	1000	0.137	+19.2%	–
	20	1000	1000	0.481	-12.9%	–
	50	500	1000	0.264	+90.7%	–
	50	1000	1000	1.518	+129.1%	–
Clifford+ T	20	100	1000	0.010	+48.9%	99.9%
	20	200	1000	0.045	+93.7%	94.9%
	20	300	1000	0.097	+115.9%	74.7%
	50	100	1000	0.016	+33.5%	100.0%
	50	200	1000	0.044	+49.0%	100.0%
	50	300	1000	0.104	+79.4%	99.6%

Table 1: Re-synthesis results for randomly generated circuits on n qubits. Avg. change gives the average percent increase (+) or decrease (-) in the re-synthesized gate count compared to the original circuit. Success gives the percentage of circuits successfully re-synthesized.

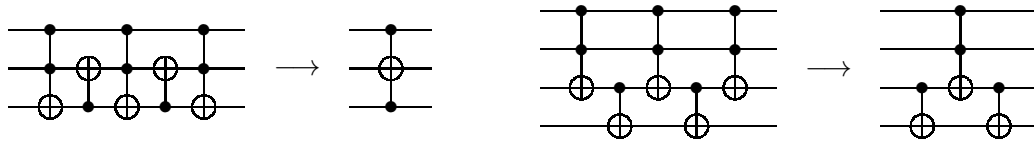
Circuit optimization One of the key factors in phase folding optimizations [\[4, 25\]](#) is the placement of Hadamard gates. It was shown in [\[5\]](#) that the T -count in a Clifford+ T circuit can be upper bounded by $O(hn^2)$, where h is the number of Hadamard layers in the circuit. As our Clifford synthesis algorithm produces circuits with just a single layer of Hadamard gates, it is natural to ask whether we can optimize T -count by reducing the number of Hadamard layers in Clifford+ T circuits.

We implemented a Clifford sub-circuit normalization method (the `-clifford` pass in FEYNOPT) using [Algorithm 1](#) to re-synthesize simple greedily chosen Clifford sub-circuits. We tested the effect on T -count optimization by applying Clifford normalization together with phase folding and compared it against [\[18\]](#) on the benchmark set of [\[4\]](#). In all but 4 benchmarks, the same T -count was achieved by normalizing greedily chosen Clifford sub-circuits and applying phase polynomial optimizations. In two of those cases, `qcla-com7` and `cs1a-mux3`, our method produced lower T -count circuits — 94 (down from 95) and 60 (down from 62), respectively. For the other two cases, `ham15-med` and `adder8`, our method produced worse results — 230 (up from 212) and 215 (up from 173), respectively.

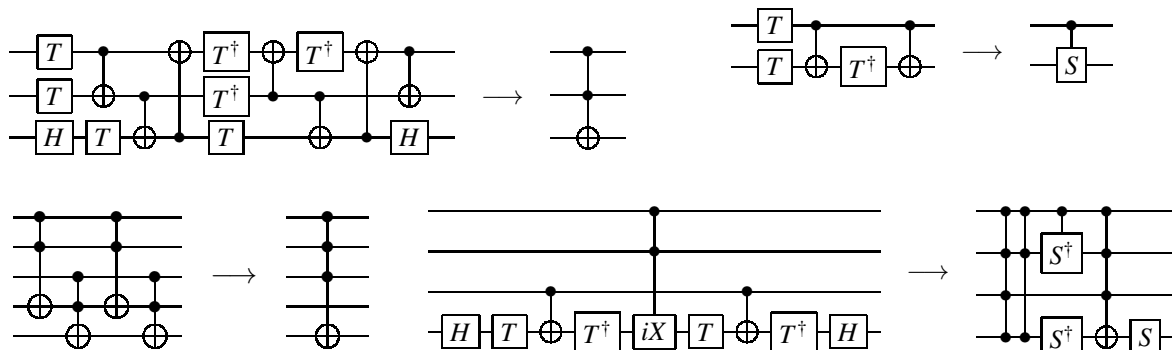
More broadly, we might expect to be able to optimize a circuit by resynthesizing its simplified path sum using the general synthesis algorithm [Algorithm 2](#). This is often effective when the path sum is simple, as in the resynthesized circuits corresponding to sub-circuits of the `adder8` benchmark below, but as the path sum becomes increasingly complex extraction typically performs worse than human designs.

¹Available at <https://github.com/meamy/feynman>.

We leave it as an avenue for future work to make symbolic synthesis practical for circuit optimization, and in particular to develop effective peephole optimization procedures.



Decompilation An interesting application of our symbolic synthesis algorithm is to the *decompilation* of quantum circuits. Classically, decompilation is the process of translating a program in a low-level language to equivalent high-level source code, typically used for reverse engineering or recompilation. As the gate set targeted by Algorithm 2 is quite high-level, in many cases it can be used to effectively decompile lower-level circuits. This decompilation can potentially help developers to examine the high-level structure of a low-level circuit, and also allow optimizations targeting higher level gate sets to be performed on circuits written over low-level gate sets, such as Clifford+ T . Below we give some examples of standard circuits from the literature decompiled using Algorithm 2. The decompiler can be accessed with the `-decompile` option in FEYNOPT.



The bottom right circuit above is a relative phase Toffoli gate implementation taken from [6]. The utility of decompilation is apparant here, as both the fact that it implements a Toffoli up to phase and the exact form of the relative phase can be readily observed from the decompiled circuit.

Specification-based synthesis In [2] it was noted that path sums offer a convenient form of logical specification for many quantum computations, being very close to the “textbook” specification. Algorithm 2 gives a method of synthesizing a circuit directly from such a specification. Such specifications include not only classical reversible functions such as $|x_1x_2x_3\rangle \mapsto |x_1x_2(x_3 \oplus x_1x_2)\rangle$, which can be synthesized by existing reversible circuit synthesis methods, but also classical functions “in the phase,” up to relative phases, or inside superpositions. We illustrate this by using Algorithm 2 to synthesize the quantum Fourier transform.

Recall that the n -qubit quantum fourier transform can be expressed as $QFT_n |\vec{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\vec{y} \in \mathbb{Z}_2^n} \omega_{2^n}^{\vec{x}\vec{y}} |\vec{y}\rangle$ where $\vec{x}\vec{y}$ is the integer product of \vec{x} and \vec{y} . In the 3 qubit case, expanding the integer multiplication to a multilinear polynomial we have

$$QFT_3 |x_1x_2x_3\rangle = \frac{1}{\sqrt{2^3}} \sum_{y_1,y_2,y_3} \omega^{x_3y_3} i^{x_3y_2+x_2y_3} (-1)^{x_3y_1+x_2y_2+x_1y_3} |y_1y_2y_3\rangle$$

where y_1 is reducible, and in particular

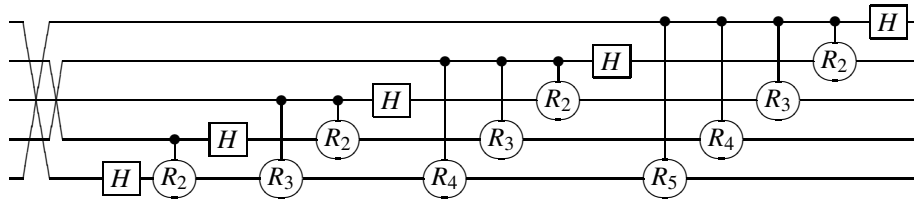
$$(\mathbf{H} \otimes \mathbf{I}_2)QFT_3 |x_1 x_2 x_3\rangle = \frac{1}{\sqrt{2^2}} \sum_{y_2, y_3} \omega^{x_3 y_3} i^{x_3 y_2 + x_2 y_3} (-1)^{x_2 y_2 + x_1 y_3} |x_3 y_2 y_3\rangle.$$

While neither of y_2 or y_3 are reducible above, the $\omega^{x_3 y_3}$ and $i^{x_3 y_2}$ terms can be eliminated by applying controlled- T^\dagger and $-S^\dagger$ gates, respectively, leaving y_2 reducible:

$$(\Lambda(S^\dagger) \otimes \mathbf{I})(\text{SWAP} \otimes \mathbf{I})(\mathbf{I} \otimes \Lambda(T^\dagger))(\text{SWAP} \otimes \mathbf{I})(\mathbf{H} \otimes \mathbf{I}_2)QFT_3 |x_1 x_2 x_3\rangle = \frac{1}{\sqrt{2^2}} \sum_{y_2, y_3} i^{x_2 y_3} (-1)^{x_2 y_2 + x_1 y_3} |x_3 y_2 y_3\rangle.$$

After eliminating y_2 , the process repeats for y_1 , leaving a final permutation to be synthesized.

A 5 qubit QFT circuit synthesized with our implementation is shown verbatim below, where $R_k := R_Z(1/2^k)$. We were able to synthesize instances on up to 50 qubits in just seconds on a desktop computer.



7 Conclusion

In this paper we looked at the problem of synthesis of unitary quantum circuits from symbolic expressions as sums-over-paths. We showed that we cannot hope to efficiently synthesize a circuit from a general path sum efficiently, as the problem of checking whether there the path sum represents a unitary transformation is itself **co-NP**-hard. A stronger result was given recently for the extraction of ZX-diagrams [11], though their work did not address the complexity of the potentially easier problem of unitarity testing. The problem of unitarity testing for ZX-diagrams is likewise believed to be intractable [31].

For the restricted case of Clifford operations, we showed that a circuit can be synthesized efficiently in the form $C_1 H C_2$ for Hadamard-free Clifford circuits C_1 and C_2 . For more general path sums we gave a heuristic based on symbolic manipulation and simplification of the sum. We experimentally validated our method, showing that most path sums corresponding to unitary transformations can in fact be synthesized. Moreover, our algorithm is capable of producing natural, high-level circuit designs for some path sums, including the quantum Fourier transform. It remains as a course of future work however to develop a complete synthesis algorithm, as well as to reduce the cost of synthesized circuits.

References

- [1] Scott Aaronson (2005): *Quantum Computing, Postselection, and Probabilistic Polynomial-Time*. *Proceedings of the Royal Society A* 461(2063), pp. 3473–3482, doi:[10.1098/rspa.2005.1546](https://doi.org/10.1098/rspa.2005.1546). arXiv:[quant-ph/0412187](https://arxiv.org/abs/quant-ph/0412187).
- [2] Matthew Amy (2018): *Towards Large-Scale Functional Verification of Universal Quantum Circuits*. In: *Proceedings of the 15th International Conference on Quantum Physics and Logic, QPL'18*, pp. 1–21, doi:[10.4204/EPTCS.287.1](https://doi.org/10.4204/EPTCS.287.1). arXiv:[1805.06908](https://arxiv.org/abs/1805.06908).

- [3] Matthew Amy, Jianxin Chen & Neil J. Ross (2017): *A Finite Presentation of CNOT-Dihedral Operators*. In: *Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL'17*, pp. 84–97, doi:[10.4204/EPTCS.266.5](https://doi.org/10.4204/EPTCS.266.5). arXiv:[1701.00140](https://arxiv.org/abs/1701.00140).
- [4] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T-depth optimization of Clifford+T circuits via matroid partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:[10.1109/TCAD.2014.2341953](https://doi.org/10.1109/TCAD.2014.2341953). arXiv:[1303.2042](https://arxiv.org/abs/1303.2042).
- [5] Matthew Amy & Michele Mosca (2019): *T-count optimization and Reed-Muller codes*. *IEEE Transactions on Information Theory* 65(8), pp. 4771–4784, doi:[10.1109/TIT.2019.2906374](https://doi.org/10.1109/TIT.2019.2906374). arXiv:[1601.07363](https://arxiv.org/abs/1601.07363).
- [6] Matthew Amy & Neil J. Ross (2021): *The phase/state duality in reversible circuit design*. *Physical Review A* 104, p. 052602, doi:[10.1103/PhysRevA.104.052602](https://doi.org/10.1103/PhysRevA.104.052602). arXiv:[2105.13410](https://arxiv.org/abs/2105.13410).
- [7] Miriam Backens & Aleks Kissinger (2018): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*. In: *Proceedings of the 15th International Conference on Quantum Physics and Logic, QPL'18*, pp. 23–42, doi:[10.4204/eptcs.287.2](https://doi.org/10.4204/eptcs.287.2). arXiv:[1805.02175](https://arxiv.org/abs/1805.02175).
- [8] 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](https://doi.org/10.22331/q-2021-03-25-421). arXiv:[2003.01664](https://arxiv.org/abs/2003.01664).
- [9] Dave Bacon, Wim van Dam & Alexander Russell (2008): *Analyzing algebraic quantum circuits using exponential sums*. Available at <https://www.cs.ucsb.edu/~vandam/LeastAction.pdf>.
- [10] Niel de Beaudrap & Steven Herbert (2022): *Fast Stabiliser Simulation with Quadratic Form Expansions*. *Quantum* 6, p. 803, doi:[10.22331/q-2022-09-15-803](https://doi.org/10.22331/q-2022-09-15-803). arXiv:[2109.08629](https://arxiv.org/abs/2109.08629).
- [11] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-Diagrams Can Be #P-Hard*. In: *Proceeds of the The 49th International Colloquium on Automata, Languages and Programming, ICALP'22*, pp. 119:1–119:19, doi:[10.4230/LIPICS.ICALP.2022.119](https://doi.org/10.4230/LIPICS.ICALP.2022.119). arXiv:[2202.09194](https://arxiv.org/abs/2202.09194).
- [12] Xiaoning Bian & Peter Selinger (2021): *Generators and Relations for $U_n(\mathbb{Z}[1/2, i])$* . In: *Proceedings of the 18th International Conference on Quantum Physics and Logic, QPL'21*, pp. 145–164, doi:[10.4204/eptcs.343.8](https://doi.org/10.4204/eptcs.343.8). arXiv:[2105.14047](https://arxiv.org/abs/2105.14047).
- [13] Sergey Bravyi & Dmitri L. Maslov (2021): *Hadamard-Free Circuits Expose the Structure of the Clifford Group*. *IEEE Transactions on Information Theory* 67, pp. 4546–4563, doi:[10.1109/tit.2021.3081415](https://doi.org/10.1109/tit.2021.3081415). arXiv:[2003.09412](https://arxiv.org/abs/2003.09412).
- [14] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In: *Proceeds of the The 35th International Colloquium on Automata, Languages and Programming, ICALP'08*, pp. 298–310, doi:[10.1007/978-3-540-70583-3_25](https://doi.org/10.1007/978-3-540-70583-3_25).
- [15] Christopher M. Dawson, Andrew P. Hines, Duncan Mortimer, Henry L. Haselgrove, Michael A. Nielsen & Tobias J. Osborne (2005): *Quantum computing and polynomial equations over the finite field \mathbb{Z}_2* . *Quantum Information and Computation* 5(2), pp. 102–112, doi:[10.26421/QIC5.2-2](https://doi.org/10.26421/QIC5.2-2). arXiv:[quant-ph/0408129](https://arxiv.org/abs/quant-ph/0408129).
- [16] Jeroen Dehaene & Bart De Moor (2003): *Clifford group, stabilizer states, and linear and quadratic operations over $GF(2)$* . *Phys. Rev. A* 68, p. 042318, doi:[10.1103/PhysRevA.68.042318](https://doi.org/10.1103/PhysRevA.68.042318). arXiv:[quant-ph/0304125](https://arxiv.org/abs/quant-ph/0304125).
- [17] 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](https://doi.org/10.22331/q-2020-06-04-279). arXiv:[1902.03178](https://arxiv.org/abs/1902.03178).
- [18] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Phys. Rev. A* 102, p. 022406, doi:[10.1103/PhysRevA.102.022406](https://doi.org/10.1103/PhysRevA.102.022406). arXiv:[1903.10477](https://arxiv.org/abs/1903.10477).
- [19] Dax Enshan Koh, Mark D Penney & Robert W Spekkens (2017): *Computing quopit Clifford circuit amplitudes by the sum-over-paths technique*. *Quantum Information and Computation* 17(13&14), pp. 1081–1095, doi:[10.26421/QIC17.13-14-1](https://doi.org/10.26421/QIC17.13-14-1). arXiv:[1702.03316](https://arxiv.org/abs/1702.03316).

- [20] Louis Lemonnier, John van de Wetering & Aleks Kissinger (2020): *Hypergraph simplification: Linking the path-sum approach to the ZH-calculus*. In: *Proceedings of the 17th International Conference on Quantum Physics and Logic*, QPL'20, pp. 188–212, doi:[10.4204/EPTCS.340.10](https://doi.org/10.4204/EPTCS.340.10). arXiv:[2003.13564](https://arxiv.org/abs/2003.13564).
- [21] Sarah Meng Li, Neil J. Ross & Peter Selinger (2021): *Generators and Relations for the Group $O_n(\mathbb{Z}[1/2])$* . In: *Proceedings of the 18th International Conference on Quantum Physics and Logic*, QPL'21, pp. 210–264, doi:[10.4204/eptcs.343.11](https://doi.org/10.4204/eptcs.343.11). arXiv:[2106.01175](https://arxiv.org/abs/2106.01175).
- [22] Justin Makary, Neil J. Ross & Peter Selinger (2021): *Generators and Relations for Real Stabilizer Operators*. In: *Proceedings of the 18th International Conference on Quantum Physics and Logic*, QPL'21, pp. 14–36, doi:[10.4204/eptcs.343.2](https://doi.org/10.4204/eptcs.343.2). arXiv:[2109.05655](https://arxiv.org/abs/2109.05655).
- [23] Dmitri Maslov & Martin Roetteler (2018): *Shorter Stabilizer Circuits via Bruhat Decomposition and Quantum Circuit Transformations*. *IEEE Transactions on Information Theory* 64, pp. 4729–4738, doi:[10.1109/TIT.2018.2825602](https://doi.org/10.1109/TIT.2018.2825602). arXiv:[1705.09176](https://arxiv.org/abs/1705.09176).
- [24] Ashley Montanaro (2017): *Quantum circuits and low-degree polynomials over \mathbb{F}_2* . *Journal of Physics A: Mathematical and Theoretical* 50(8), p. 084002, doi:[10.1088/1751-8121/aa565f](https://doi.org/10.1088/1751-8121/aa565f). arXiv:[1607.08473](https://arxiv.org/abs/1607.08473).
- [25] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs & Dmitri Maslov (2018): *Automated Optimization of Large Quantum Circuits with Continuous Parameters*. *npj Quantum Information* 4(1), p. 23, doi:[10.1038/s41534-018-0072-4](https://doi.org/10.1038/s41534-018-0072-4). arXiv:[1710.07345](https://arxiv.org/abs/1710.07345).
- [26] Maarten Van den Nest (2010): *Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond*. *Quantum Information and Computation* 10(3&4), pp. 0258–0271, doi:[10.26421/QIC10.3-4-6](https://doi.org/10.26421/QIC10.3-4-6). arXiv:[0811.0898](https://arxiv.org/abs/0811.0898).
- [27] Terry Rudolph (2009): *Simple encoding of a quantum circuit amplitude as a matrix permanent*. *Physical Review A* 80, p. 054302, doi:[10.1103/PhysRevA.80.054302](https://doi.org/10.1103/PhysRevA.80.054302). arXiv:[0909.3005](https://arxiv.org/abs/0909.3005).
- [28] Peter Selinger (2015): *Generators and relations for n -qubit Clifford operators*. *Logical Methods in Computer Science* Volume 11, Issue 2, doi:[10.2168/LMCS-11\(2:10\)2015](https://doi.org/10.2168/LMCS-11(2:10)2015). arXiv:[1310.6813](https://arxiv.org/abs/1310.6813).
- [29] Grigori S Tseitin (1983): *On the complexity of derivation in propositional calculus*. In: *Automation of reasoning*, Springer, pp. 466–483, doi:[10.1007/978-3-642-81955-1_28](https://doi.org/10.1007/978-3-642-81955-1_28).
- [30] Renaud Vilmart (2021): *The Structure of Sum-Over-Paths, its Consequences, and Completeness for Clifford*. In: *Foundations of Software Science and Computation Structures, FoSSaCS'21* 12650, Luxembourg, Luxembourg, pp. 531–550, doi:[10.1007/978-3-030-71995-1_27](https://doi.org/10.1007/978-3-030-71995-1_27). arXiv:[2003.05678](https://arxiv.org/abs/2003.05678).
- [31] John van de Wetering (2021): Private communication.

A Finding generalized permutations

In this appendix we detail our method for finding a generalized permutation in step 2.(a) of [Algorithm 2](#).

Compared to Clifford operators, simplification via [Proposition 2.9](#) may not always leave the path sum in a reducible state. For instance, the path sum expression below, corresponding to a unitary transformation, is fully reduced with respect to [Equations \(2\), \(3\) and \(4\)](#):

$$\frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 y_1 - x_2 y_2} (-1)^{x_1 y_1 + x_1 y_2 + x_2 y_1 y_2} |y_1\rangle |y_2\rangle$$

However, neither y_1 nor y_2 are reducible due to the quadratic terms $x_2 y_1$ and $x_2 y_2$ in the exponent of i . At the moment, it is unclear how to proceed symbolically to find a generalized permutation that will make either path variable reducible in the above expression.

Our heuristic method of producing a generalized permutation proceeds in increasingly costly circuit stages in an attempt to synthesize as efficient circuits as possible. Rather than attempt to synthesize a distinct generalized permutation for every path variable y as described in [Algorithm 2](#), we first apply a

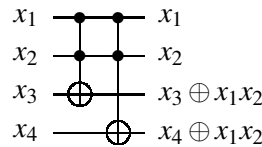
sequence of simplification stages generically to both reduce the redundant synthesis work, and produce simpler circuits in practice. The sequence of stages is given in Algorithm 3, and the individual synthesis steps are described in detail below.

Algorithm 3 Generalized permutation synthesis heuristic

1. Apply affine simplifications to the output state $|f(\vec{x}, \vec{y})\rangle$
 2. Apply non-linear simplifications to the phase $e^{2\pi i P(\vec{x}, \vec{y})}$
 3. Apply non-linear simplifications to the output state $|f'(\vec{x}, \vec{y})\rangle$
 4. Apply non-linear simplifications to the phase $e^{2\pi i P'(\vec{x}, \vec{y})}$
 5. If no path variable is reducible, attempt degree reduction on each variable
-

Affine simplifications As X and $CNOT$ gates are relatively inexpensive, the first stage of our generalize permutation synthesis attempts to simplify the output $|f(\vec{x}, \vec{y})\rangle$ of the path sum as much as possible using only these affine transformations. In order to reduce the number of high-degree terms, which would otherwise require expensive multiply-controlled Toffoli gates, we perform affine simplifications on a *linearization* of f . Specifically, we write each $f_i(\vec{x}, \vec{y})$ as a sparse vector $\vec{u}_i \in \mathbb{Z}_2^{2^{n+k}}$ using reverse lexicographic order for the encoding of monomials, then set $A = [u_1 \ u_2 \ \dots \ u_n]^T$ and use Gaussian elimination to compute a sequence of $CNOT$ gates reducing A to echelon form. The example below illustrates our method.

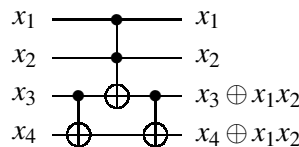
Example A.1. Consider the path sum $|x_1\rangle |x_2\rangle |x_3 \oplus x_1x_2\rangle |x_4 \oplus x_1x_2\rangle$. This could naturally be synthesized using two non-linear Toffolis to eliminate the x_1x_2 terms from the third and fourth qubits. The resulting circuit is shown below:



Alternatively, we can write the output as a (sparse) linear system over all monomials in x_1, x_2, x_3, x_4 as shown below:

	x_1x_2	x_4	x_3	x_2	x_1
x_1	0	0	0	0	1
x_2	0	0	0	1	0
$x_3 \oplus x_1x_2$	1	0	1	0	0
$x_4 \oplus x_1x_2$	1	1	0	0	0

We use reverse lexicographic order so that reduction to echelon form will prioritize the number of high degree terms. Reducing this to echelon form results in a single $CNOT$ gate and reduces the state to $|x_1\rangle |x_2\rangle |x_3 \oplus x_1x_2\rangle |x_4 \oplus x_3\rangle$. Synthesizing this remaining transformation gives the overall circuit



Phase simplifications To reduce and simplify the number of terms in the phase $e^{2\pi i P(\vec{x}, \vec{y})}$ of a path sum controlled R_Z gates with continuous parameters are used. In particular, given an n -dimensional path sum $e^{2\pi i \theta \prod_i x_i} |\vec{x}\rangle$, we can reduce the phase term by applying a $\Lambda_n(R_Z(-\theta))$ gate, since

$$\Lambda_n(R_Z(-\theta)) : e^{2\pi i \theta \prod_i x_i} |\vec{x}\rangle \mapsto |\vec{x}\rangle.$$

As the output of the path sum is in some state $|f(\vec{x}, \vec{y})\rangle$, to apply the above rule we first need to apply a *change of frame* by setting $f_i(\vec{x}, \vec{y}) = z_i$ and writing the phase polynomial $P(\vec{x}, \vec{y})$ as $P'(\vec{x}, \vec{y}, \vec{z})$. This is achieved by, for each f_i , letting l be the *largest* (non-zero) degree term of f_i and substituting $l \leftarrow z_i \oplus l \oplus f_i(\vec{x}, \vec{y})$ in the path sum.

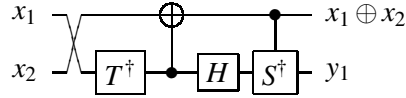
Example A.2. Consider the irreducible path sum $\frac{1}{\sqrt{2}} \sum_{y_1} \omega^{-x_1} i^{x_1 y_1 - x_2 y_1} (-1)^{x_1 x_2 y_1} |x_1 \oplus x_2\rangle |y_1\rangle$. Substituting $[x_2 \leftarrow z_1 \oplus x_1, y_1 \leftarrow z_2]$ gives the re-framed path sum

$$\omega^{-x_1} i^{-z_1 z_2} (-1)^{x_1 z_2} |z_1\rangle |z_2\rangle.$$

Applying a controlled S gate to eliminate the term $i^{-z_1 z_2}$ and rolling back the substitutions gives

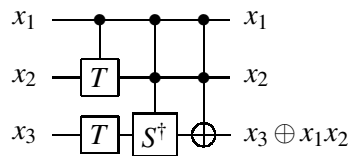
$$\omega^{-x_1} (-1)^{x_1 y_1} |x_1 \oplus x_2\rangle |y_1\rangle.$$

The variable y_1 is now reducible, so we can finish synthesis by applying a Hadamard to the second qubit, then synthesizing the final generalized permutation $|x_1 x_2\rangle \mapsto \omega^{-x_1} |x_1 \oplus x_2\rangle |x_1\rangle$. The resulting circuit is given below:



In our implementation, we apply phase simplifications both before and after non-linear simplifications in the state. This is so that we can effectively utilize high degree terms in the state to simplify high degree terms in the phase with phase gates on fewer qubits. The following example illustrates this effect.

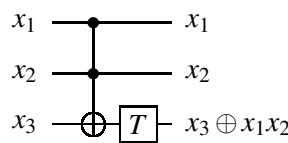
Example A.3. Consider the path sum $\omega^{x_3 + x_1 x_2} i^{-x_1 x_2 x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1 x_2\rangle$. Eliminating the $x_1 x_2$ term in qubit 3 before simplifying the phase results in the following circuit:



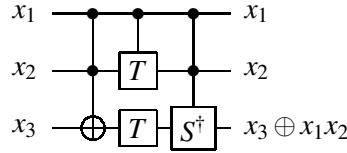
However, by re-framing the sum with the substitution $[x_1 \leftarrow z_1, x_2 \leftarrow z_2, z_1 z_2 \leftarrow z_3 \oplus x_3]$ we find

$$\omega^{x_3 + x_1 x_2} i^{-x_1 x_2 x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1 x_2\rangle \equiv \omega^{z_3} |z_1\rangle |z_2\rangle |z_3\rangle$$

which can now be simplified with a single T gate. Note that the substitution is applied left to right, rather than as a simultaneous substitution. The resulting circuit is shown below:



The final substitution $z_1 z_2 \leftarrow z_3 \oplus x_3$ may seem counter-intuitive, as we could instead substitute $x_3 \leftarrow z_3 \oplus z_1 z_2$. We choose a monomial of maximal degree to substitute in order to avoid inadvertently increasing the degree of the phase polynomial. For instance, if the initial path sum was instead $\omega^{x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1 x_2\rangle$, substituting $x_3 \leftarrow z_3 \oplus z_1 z_2$ results in a re-framed sum of $\omega^{z_3 + z_1 z_2} i^{-z_1 z_2 z_3} |z_1\rangle |z_2\rangle |z_3\rangle$ and the final circuit



By substituting the highest degree monomial instead, we avoid this issue and synthesize the simpler circuit placing the T gate to the left of the Toffoli.

Non-linear simplifications The non-linear simplification step of our synthesis algorithm reduces the number of non-linear terms in the output $|f(\vec{x}, \vec{y})\rangle$ by applying multiply-controlled Toffoli gates $\Lambda_k(X)$ via the rule

$$\Lambda_k(X) : |\vec{x}\rangle |f \oplus \prod_i x_i\rangle \mapsto |\vec{x}\rangle |f\rangle.$$

Our method for non-linear simplifications uses a naïve heuristic whereby a set of variables

$$V = \{v \mid f_i(\vec{x}, \vec{y}) = v \text{ for some } i\}$$

is computed. Any term in $f(\vec{x}, \vec{y})$ which is a product of variables contained in V is then eliminated with an appropriately controlled Toffoli gate.

This method is far from optimal, and in particular misses cases which can be factorized as a cascade of Toffoli gates. While better reversible synthesis methods exist, the lack of a known permutation to synthesize *a priori* in our case makes it difficult to apply such methods directly. An interesting avenue for future work would be to re-synthesize the permutation discovered through this process of simplification using state-of-the-art methods.

Degree reduction In many cases, the simplifications previously described fail to leave some path variable in a reducible position. When this happens, our last resort is to fall back to an exponential time procedure we call *degree reduction*. The idea of is to reduce the degree of the (non-Boolean) parts of a phase polynomial restricted to a particular variable, as these terms serve as roadblocks for reduction. This can in some cases be accomplished by applying variable substitutions in such a way as to cancel out terms involving a particular variable.

To illustrate degree reduction, recall the irreducible path sum expression from the beginning of this section,

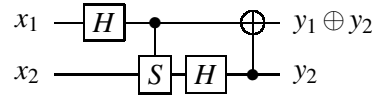
$$\frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 y_1 - x_2 y_2} (-1)^{x_1 y_1 + x_1 y_2 + x_2 y_1 y_2} |y_1\rangle |y_2\rangle$$

The exponent of i cannot be directly reduced via simple phase simplifications, as both terms depend on x_2 . However, we can *indirectly* eliminate one of these terms by applying Equation (5), substituting y_1

with $y_1 \oplus y_2$:

$$\begin{aligned} & \frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 y_1 - x_2 y_2} (-1)^{x_1 y_1 + x_1 y_2 + x_2 y_1 y_2} |y_1\rangle |y_2\rangle \\ &= \frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 (y_1 \oplus y_2) - x_2 y_2} (-1)^{x_1 (y_1 \oplus y_2) + x_1 y_2 + x_2 (y_1 \oplus y_2) y_2} |y_1 \oplus y_2\rangle |y_2\rangle \quad \text{by Equation (5)} \\ &= \frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 y_1} (-1)^{x_1 y_1 + x_2 y_2} |y_1 \oplus y_2\rangle |y_2\rangle \end{aligned}$$

The final expression above can be simplified to $\frac{1}{\sqrt{2^2}} \sum_{y_1, y_2} i^{x_2 y_1} (-1)^{x_1 y_1 + x_2 y_2} |y_1\rangle |y_2\rangle$ by applying a *CNOT* gate, which leaves y_2 in a reducible position. The resulting circuit is given below:



The above example is relatively easy to spot, but more complicated cases may require substitution of multiple variables, or even non-linear substitutions. Our heuristic revolves around computing a type of *cover* for the quotient $2(P/y)$, where y is the candidate for degree reduction.

Lemma A.4. *Let $P \in \mathbb{R}[y, x_1, \dots, x_n]$ such that $4(P/y) \equiv 0 \pmod{2}$. If there exists a set $S \subseteq \{1, \dots, n\}$ such that $4(P/x_i) \equiv 0 \pmod{2}$ for all $i \in S$ and*

$$\sum_{i \in S} 2(P/x_i) \equiv 2(P/y) \pmod{2}$$

then $2(P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S]/y) \equiv 0 \pmod{2}$

Proof. First recall that $\overline{x \oplus y} = x + y - 2xy$. Hence

$$2P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S] = 2P + \sum_{i \in S} 2y(P/x_i) + \sum_{i \in S} 4x_i y(P/x_i)$$

Taking the quotient by y we see

$$\begin{aligned} 2(P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S]/y) &= 2(P/y) + \sum_{i \in S} 2(P/x_i) + \sum_{i \in S} 4x_i(P/x_i) \\ &\equiv 2(P/y) + 2(P/y) \pmod{2} \\ &\equiv 0 \pmod{2} \end{aligned}$$

□

In the context of path sums, **Lemma A.4** tells us that if $e^{2\pi i P(\vec{x}, \vec{y})}$ can be written as $i^{y_i} Q_j(\vec{x}, \vec{y}) e^{2\pi i R(\vec{x}, \vec{y})}$ for some i , and there exists a subset of path variables $\{y_j \mid j \neq i\}$ such that $e^{2\pi i P(\vec{x}, \vec{y})} = i^{y_j} Q_j(\vec{x}, \vec{y}) e^{2\pi i R_j(\vec{x}, \vec{y})}$ and $i^{y_j} Q_j(\vec{x}, \vec{y}) = i^{Q(\vec{x}, \vec{y})}$, then the simultaneous substitution $y_j \leftarrow y_j \oplus y_i$ will eliminate the term $i^{y_i} Q(\vec{x}, \vec{y})$. Additional simplifications in the state may then be further required to leave the path sum in a reducible state, as in the previous above.

Dynamic Qubit Routing with CNOT Circuit Synthesis for Quantum Compilation

Arianne Meijer - van de Griend

Department of Computer Science
University of Helsinki
ariannemeijer@gmail.com

Sarah Meng Li

Department of Combinatorics & Optimization
Institute for Quantum Computing, University of Waterloo
sarah.li@uwaterloo.ca

Many quantum computers have constraints regarding which two-qubit operations are locally allowed. To run a quantum circuit under those constraints, qubits need to be mapped to different quantum registers, and multi-qubit gates need to be routed accordingly. Recent developments have shown that compiling strategies based on Steiner tree provide a competitive tool to route CNOTs. However, these algorithms require the qubit map to be decided before routing. Moreover, the qubit map is fixed throughout the computation, i.e. the logical qubit will not be moved to a different physical qubit register. This is inefficient with respect to the CNOT count of the resulting circuit.

In this paper, we propose the algorithm *PermRowCol* for routing CNOTs in a quantum circuit. It dynamically remaps logical qubits during the computation, and thus results in fewer output CNOTs than the algorithms *Steiner-Gauss* [14] and *RowCol* [27].

Here we focus on circuits over CNOT only, but this method could be generalized to a routing and mapping strategy on Clifford+T circuits by slicing the quantum circuit into subcircuits composed of CNOTs and single-qubit gates. Additionally, *PermRowCol* can be used in place of *Steiner-Gauss* in the synthesis of phase polynomials as well as the extraction of quantum circuits from ZX-diagrams.

1 Introduction

Recent strides in quantum computing have made it possible to execute quantum algorithms on real quantum hardware [4, 29]. Contrary to classical computing, efficient quantum circuits are necessary for successful execution due to the decoherence of qubits [18]. If a quantum circuit takes too long to execute, it will not produce any usable results. Moreover, due to poor gate fidelities, each additional gate in the quantum circuit adds a small error to the computation. In the absence of fault-tolerant quantum computers, circuits with more gates produce less accurate results. Therefore, we need to reduce the gate complexity of the executed quantum circuits. This requires resource-efficient algorithms and improved quantum compiling procedures.

When mapping a quantum circuit to the physical layer, one has to consider the numerous constraints imposed by the underlying hardware architecture. For example, in a superconducting quantum computer [24], connectivity of the physical qubits restricts multi-qubit operations to adjacent qubits. These restrictions are known as *connectivity constraints* and can be represented by a *connected graph* (also known as a *topology*). Each vertex represents a distinct physical qubit. When two qubits are adjacent, there is an edge between the corresponding vertices.

Thus, we are interested in improving the routing of a quantum circuit onto a quantum computer. Current routing strategies are dominated by SWAP-based approaches [15, 26, 23, 28]. These strategies move the logical qubits around on different quantum registers. The drawback of this is that every SWAP-gate adds 3 CNOTs to the circuit (Figure 1.a), adding only more gates to the original circuit. As a result, it will take much longer to execute a routed quantum circuit, and thus introduce more errors to the computation.

Additionally, these SWAP-based strategies can be replaced by a *bridge template* (or *bridge*) that acts like a remote CNOT. As shown in Figure 1.b, the bridge template only requires 4 CNOTs while swapping the qubits would have cost 7 CNOTs (Figure 1.c). The CNOT ladders in the bridge template can be generalized for remote CNOTs with more qubits in between. Note that usually in SWAP-based strategies, the last SWAP (Figure 1.c) is omitted, resulting in 4 CNOTs instead of 7. Therefore, unlike with SWAP gates, the subsequent parts of the circuit cannot benefit from the new qubit placement because the bridge template does not move the qubits. Thus, we need to make a trade-off between swapping qubits and remote CNOTs when there is a sequence of CNOTs to be routed.

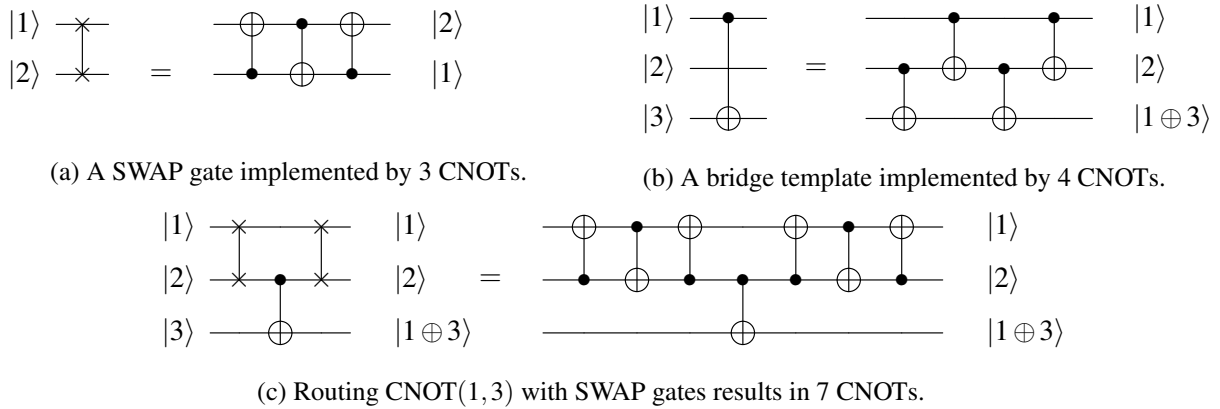


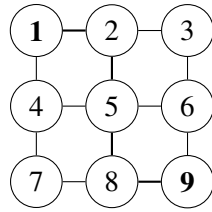
Figure 1: Visualize how a SWAP gate and a bridge template is implemented by CNOTs respectively. Note that for a single CNOT constrained by a 3-qubit line topology, the bridge gate is the same as the output of the *Steiner-Gauss* algorithm .

Alternatively, we can use Steiner-tree based synthesis [2, 14, 17, 11, 25, 10] to find a generalized bridge gate for a CNOT circuit. The new sequence of CNOTs has the same effect on the logical states as the original CNOT circuit, and every gate is permitted by the hardware's connectivity constraint. This is done by changing the rigid representation of the quantum circuit to a more flexible one (Section 2.1), with which we could synthesize a routed circuit (Section 2.3). This way, we can make global improvements to the CNOT circuit re-synthesis more easily. By re-synthesis, we refer to the process of turning a CNOT circuit into a parity matrix and synthesizing an equivalent CNOT circuit from that parity matrix, up to permutation.

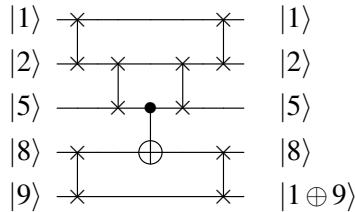
The *Steiner-Gauss* algorithm provides the first Steiner-tree based synthesis approach and it is used to synthesize CNOT circuits [14, 17]. Later on, it is used for synthesizing circuits over CNOT and R_z gates [17, 11, 25], and further for circuits over CNOT, R_z , and NOT gates. Finally, this algorithm is generalized for Clifford+ T circuits with the *slice-and-build* protocol based on the locality of Hadamard gates in the circuits [10].

One major drawback of the existing Steiner-tree based methods is that they are not flexible with respect to the mapping of qubits. Logical qubits of the synthesized circuit will always be stored in the same physical qubit registers where they were originally allocated. However, this is not always optimal. Figure 2 shows an example where there exists a smaller synthesized circuit by reallocating logical qubits to the physical registers (Figure 2.c) than using *Steiner-Gauss* (Figure 2.d). Note that with the fixed qubit map, *Steiner-Gauss* produces fewer CNOTs than the SWAP-based method (Figure 2.b). Thus, we conclude that the synthesized circuit in Figure 2.d contains implicit SWAP gates.

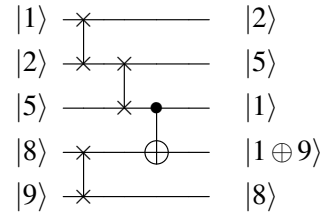
Moreover, remapping logical outputs of a quantum circuit to physical registers based on the original



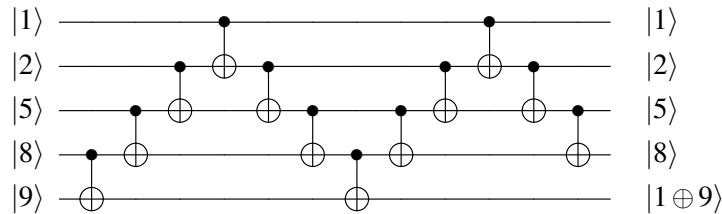
(a) The 9-qubit square grid.



(b) SWAP template with a fixed qubit map.



(c) SWAP template with dynamic qubits maps.



(d) Steiner-Gauss with a fixed qubit map [14].

Figure 2: Given the constrained topology in Figure 2.a, three different routing strategies are compared for implementing CNOT(1,9). The CNOT count corresponding to each template is 19, 10, and 12 respectively.

qubit mapping can be done by a classical operation. Hence, such operation is considered trivial in quantum computing. In other words, preparing a qubit in a register, routing it to a different register and measuring it does not influence the computation of a quantum circuit. Thus, it is desirable to have a synthesis procedure that permits dynamic qubit maps.

The first CNOT synthesis procedures [17, 14] under topological constraints are based on *Gaussian elimination*: the parity matrix representing the synthesized CNOT circuit is eliminated to the identity matrix (Section 2.3). In Section 2.4, we show that dynamically changing the qubit maps is the same as eliminating the parity matrix into the identity matrix up to permutation. To do this, we need to determine to which permutation of the identity matrix to synthesize a priori, which is not a trivial task. However, by adjusting the algorithm *RowCol* [27], we can determine the new qubit map whilst synthesizing a CNOT circuit.

Here, we propose the algorithm *PermRowCol*: a new Steiner-tree based synthesis method for CNOT circuits re-synthesis under topological constraints. It dynamically determines the output qubit maps. This method could be generalized to synthesizing an arbitrary quantum circuit, as articulated in Section 6.2.

The paper is structured as follows. In Section 2, we introduce the Steiner-tree based synthesis approach. In Section 3, we describe our algorithm *PermRowCol*. In Section 4, we show how well it performs against *Steiner-Gauss* [14] and *RowCol* [27]. In Section 5, we discuss the results and other open problems. Finally, we explain some ongoing improvements for *PermRowCol* in Section 6. Look-

ing forward, our goal is to fairly compare *PermRowCol* with CNOT re-synthesis algorithms in quantum compilers such as SABRE [15], Qiskit [26], and TKET [23].

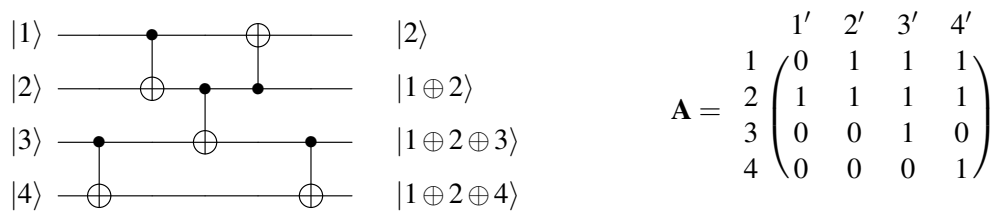
2 Preliminaries

Here we introduce the core concepts required to understand the proposed algorithm *PermRowCol*. In Section 2.1, we define the matrix representation of a CNOT circuit. In Section 2.2, we describe the concepts of Steiner tree. In Section 2.3, we use it for synthesizing a CNOT circuit under a constrained topology. In Section 2.4, we explain the core idea behind algorithm *PermRowCol*: the dynamic mapping of qubits using permutation matrices.

2.1 The parity matrix of a CNOT circuit

In this paper, we consider circuits composed of only CNOTs, and call them *CNOT circuits*. CNOT is short for "controlled not". It acts on two qubits: a control and a target. We write $\text{CNOT}(c,t)$ to denote a CNOT applied between a control qubit c and a target qubit t . The control qubit c decides whether a NOT gate is applied to the target qubit t . When $|c\rangle = |0\rangle$, $\text{CNOT}(c,t)$ acts trivially on $|t\rangle$, leaving it unchanged. Otherwise, $|t\rangle = |0\rangle$ is changed to $|t\rangle = |1\rangle$ and vice versa. Alternatively, we write that $\text{CNOT}(c,t)$ changes $|t\rangle$ to $|c \oplus t\rangle$, where \oplus denotes addition modulo 2.

For a CNOT circuit, we can keep track of its state evolution by checking which qubits appear in the summation modulo 2 at the circuit output. In Figure 3.a, there are 5 CNOTs acting on 4 qubits, whose overall behaviour is described by the sum of some logical qubits on each output wire. We call such a sum a *parity term* because it keeps track of whether a logical qubit participates in the sum or not. As such, we can write a parity term as a binary string whose length is equal to the number of qubits in the circuit. In this representation, a 0 means that the corresponding logical qubit is not present in the sum, and a 1 means otherwise.



(a) A CNOT circuit acting on 4 qubits.

(b) Parity matrix for figure (a).

Figure 3: The matrix representation of a 4-qubit CNOT circuit.

We can use the output parities of a CNOT circuit to create a square matrix where each column represents a parity term and each row represents an input qubit. This matrix is called a *parity matrix* and its properties are well-studied in [1, 22, 19]. Figure 4 shows two examples of constructing the parity matrix for the given CNOT circuits. Additionally, adding a CNOT to the circuit corresponds to a row operation on the parity matrix. That is, adding the row indexed by the target to the row indexed by the control, while keeping the target-indexed row unchanged (Figure 12.b).

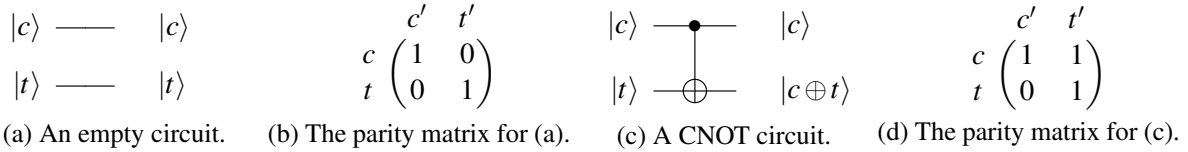


Figure 4: Example constructions of parity matrices.

This means that we can extract a CNOT circuit from a parity matrix by adding rows of the parity matrix to other rows until we obtain the identity matrix. In the literature, Gaussian elimination is a commonly used algorithm for CNOT circuit synthesis [19, 14, 27, 10]. Here, we refer readers to textbooks on linear algebra for more details about Gaussian elimination.

2.2 Steiner tree

We use Steiner trees to enforce the connectivity constraints when synthesizing a semantically equivalent CNOT circuit from the parity matrix. Note that Steiner trees are not the only approach to carry out the architecture-aware CNOT circuit synthesis procedure (e.g., see [6] for an alternative), but it is the method we use here.

We start with the basics; a *graph* is an order pair $G = (V_G, E_G)$, where V_G is a set of *vertices* and E_G is a set of edges. Each *edge* is defined as $e = (u, v)$ where $u, v \in V_G$. The *degree* of a vertex is the number of edges that are incident to that vertex. Graphs also have a *weight* assigned to each edge by a weight function $\omega_E : E_G \rightarrow \mathbb{R}$.

The connectivity graph of a quantum computer is generally considered a *simple* graph, meaning that it is an *undirected* graph (i.e., $(u, v) \equiv (v, u)$) with all edge weights equal to 1. It has at most one edge between two distinct vertices (i.e. $\forall (e, e') \in E_G : e \neq e'$), and no self-loops (i.e., $(u, u) \notin E_G$).

Some graphs are *connected*, meaning that for every vertex there exists a sequence of edges (a *path*) from which we can go from that vertex to any other vertex in the graph. A graph that is not connected is *disconnected*. The topology of a quantum computer needs to be connected if we want any pair of arbitrary qubits to interact with each other. A *cut vertex* is a vertex that when removed, the graph will become disconnected. We use *non-cutting vertex* to mean a vertex that is not a cut vertex.

A *subgraph* $G' = (V'_G, E'_G)$ of G is a graph that is wholly contained in G such that $V'_G \subseteq V_G, E'_G \subseteq E_G$, and for all $(u, v) \in E'_G, u, v \in V'_G$.

A *tree* is an undirected connected graph that has no path which starts and ends at the same vertex. A tree is *acyclic*. A *minimum spanning tree* T of a connected graph G is a subgraph of G with the same set of vertices V_G and a subset of the edges E_G such that the sum of the edge weights is minimal and T is still connected.

A Steiner tree is similar to a minimum spanning tree. It is defined as follows.

Definition 2.1 (Steiner tree). Given a graph $G = (V_G, E_G)$ with a weight function ω_E and a set of vertices $S \subseteq V_G$, a Steiner tree $T = (V_T, E_T)$ is a tree that is a subgraph of G such that $S \subseteq V_T$ and the sum of edge weights in E_T is minimized. The vertices in S are called *terminals* while those in $V_T \setminus S$ are called *Steiner nodes*.

Figure 5 demonstrates a solution to the Steiner tree problem on a 12-qubit grid, with $S = \{1, 6, 7, 11\}$. In Figure 5.a, nodes in S are coloured in red. The edges of the Steiner tree T are highlighted in green and the Steiner nodes can be read off from the graph. For example, in Figure 5.b, they are the unfilled node on the paths of $T: V_T \setminus S = \{4, 5, 8\}$. Note that the solution to a Steiner tree problem may not be unique.

For the graph G and a set of terminals S in Figure 5.a, Figure 5.b and Figure 5.c provide two solutions. In this example and the remainder of this paper, we assume each edge is of unit weight without loss of generality.

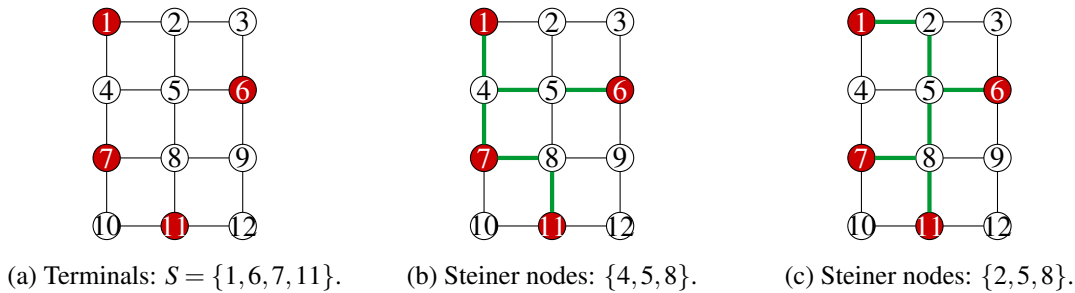


Figure 5: Solutions to the Steiner tree problem on the 12-qubit grid G .

Computing Steiner trees is NP-hard and the related decision problem is NP-complete [13]. There are a number of heuristic algorithms that compute approximate Steiner trees [20, 7, 12]. There is a trade-off between the size of the approximate Steiner tree and the algorithm's runtime, so the choice of algorithm is determined by its application. Here, we create an approximate Steiner tree by building a minimum spanning tree over the terminals (we use Prim's algorithm [8]), with the weights being the distance between the terminals. Whenever a terminal is added to the spanning tree, its entire path to the tree is added. The weights calculated in the next step involve the paths between the tree up until then and terminals not yet added to the spanning tree. For this, we use Floyd-Warshall's algorithm [8] to calculate the shortest path between all qubits.

2.3 Synthesizing CNOT circuits for specific topologies

Given the parity matrix of a CNOT circuit, we want to synthesize an equivalent CNOT circuit such that all CNOTs are allowed according to a connectivity graph. From Section 2.1, we know that every CNOT corresponds to a row operation in the parity matrix. Moreover, we can use Gaussian elimination to turn the parity matrix into the identity matrix. The process of adding rows are called *elimination*. If we keep track of which row operations are performed during the Gaussian elimination process, we obtain a CNOT circuit that is *semantically equivalent*, which means that the parity matrix of the original circuit is equal to that of the synthesized circuit. Hence, these two circuits have the same input-output behaviour.

For the extracted CNOTs to adhere to the given connectivity constraints, we need to adjust our method to allow only row operations that correspond to the connected vertices in the topology. There are several methods to do this [14, 17, 19]. Our algorithm is based on the algorithm *RowCol* [27], which reduces the input parity matrix into the identity matrix by eliminating a column and a row at each round of execution.

It starts by selecting a qubit (i.e., a vertex in the connectivity graph). This determines the *pivot column*, which is the column to be eliminated. To ensure the updated topology stays connected, the selected qubit must correspond to a non-cutting vertex. This means we can select qubits in an arbitrary order as long as the vertex removal does not disconnect the graph. Then we build Steiner trees to find the shortest paths over which the row additions are performed. As a result, the eliminations send the pivot column and a corresponding row to the basis vectors, after which they are removed from the parity matrix. Accordingly, the selected qubit is no longer needed and the vertex is removed from the graph.

Next, the algorithm starts on a smaller instance of the problem. In what follows, we explain the column and row eliminations in more details.

To eliminate a column, we identify all 1s in the column. Then a Steiner tree T is built with the diagonal as the root and the rows with a 1 as terminals. These rows should be added together so that we end up with an column equal to a basis vector. Due to the connectivity constraints, we use the Steiner nodes to "move" the 1s to the terminals. This is done by traversing T from the bottom up. When a Steiner node is reached, we add its child to itself, so the corresponding row in the pivot column is equal to 1. Then, T is traversed again, during which each row is added to its child in the Steiner tree. As a result, only the root row will be 1, while other rows will be 0 in the pivot column. Thus, the pivot column is eliminated. This procedure is also described by Algorithm 4 in Appendix A.

A row is eliminated in a similar manner. For brevity, we call it a *pivot row*. Compared to the column elimination, it is less straightforward to find which row operations to perform in order to eliminate the pivot row. According to [27], they are determined by solving a system of linear equations defined by the parity matrix. We can once more build a Steiner tree T' with the diagonal as root and the rows added the pivot row as terminals. Then we traverse T' top down from the root, adding every Steiner node to its parent. Next, we traverse T' bottom up and add every node to its parent. As a result, the rows corresponding to the Steiner nodes are added twice to their parent. They do not participate in the sum since the row addition is performed modulo 2. Moreover, every terminal is added together and propagated to the root after the bottom-up traversal. This procedure is described as by Algorithm 5 in Appendix A.

2.4 Dynamic qubit mapping with permutation matrices

The state-of-the-art CNOT circuit synthesis eliminates the parity matrix to an identity matrix, which is essentially a qubit map where logical qubit i is stored in the physical qubit register i . This means that the exact synthesis preserves the circuit's semantics and moves the logical qubits back to their original registers. However, to route CNOTs in a topologically-constrained quantum circuit, it might not be necessary to stick to the parity matrix faithfully. Moreover, restoring the logical values in each physical qubit register adds extra CNOTs to the synthesis results. For example, in Figure 2, the synthesized circuit (Figure 2.d) have less CNOTs if we allow dynamic qubit maps (Figure 2.c). Additionally, if the CNOT synthesis is done as part of a slice-and-build approach where each subcircuit is synthesized locally [10], the cost of keeping the logical qubits in the same physical qubit registers grows linearly with the number of slices.

Recall that in the parity matrix, the i th column stores the parity term being output from the i th qubit of the CNOT circuit, $1 \leq i \leq n$, where n is the number of qubits in the circuit. In other words, the order of the columns corresponds to the different physical qubit registers where these parity terms are stored. If we change which parity term ends up on each physical qubit register, we can equivalently synthesize a CNOT circuit where the parity matrix has its columns reordered with respect to the original parity matrix. In the case of routing CNOTs, this is exactly possible. Reordering the columns of the identity matrix corresponds to reading the logical qubits from different quantum registers that are defined by the new column order. Since reading the circuit output from different registers is a classical operation, it can be considered free when running a quantum circuit.

In this work, we remove the restriction on qubit maps in CNOT circuit synthesis and propose an algorithm that eliminates the parity matrix to the *permutation matrix*, which is an identity matrix with reordered columns. Accordingly, a parity matrix M that is a permutation matrix can be seen as a *qubit map* where the logical qubit i is stored in the physical qubit register j iff $M_{i,j} = 1$.

However, it is not trivial to determine an optimal qubit remap. In [14], a generic algorithm *Steiner-Gauss* was proposed to find a better qubit map, but it is unfortunately unscalable. In the next section, we explain how to do this in a scalable way.

2.5 Reverse traversal strategy

Reverse Traversal Strategy (RT) [15] leverages the reversibility of quantum circuits. We define a reverse circuit to be the circuit that is the original circuit in reverse. Since CNOTs are self-inverse, this is equivalent to the inverse circuit. Reverse Traversal iteratively improves the initial qubit mapping using any routing procedure that optimizes the circuit and remaps the qubits. It does this by using the output qubit mapping as input qubit mapping for the reverse circuit and reapplying the routing procedure. Because the routing procedure optimizes the circuit in the process, we might find a smaller circuit than what we started out with. Then, we can repeat this processes of routing the reverse-reverse circuit to obtain a new output mapping and possibly find a better circuit. We refer the reader to the original paper [15] for a more detailed explanation with pictures.

This technique can only be used in a routing method that remaps the qubits. Therefore, RT could not be used for Steiner-tree based methods until now.

3 The PermRowCol Algorithm

Here, we introduce the algorithm *PermRowCol*. It is a Steiner-tree based synthesis algorithm that dynamically remaps the logical qubits to physical registers while routing CNOTs. To this end, we build on the algorithm *RowCol* [27] described in Section 2.3. This algorithm is executed iteratively on the input parity matrix until it is eliminated to an identity. At each round, *RowCol* picks a new logical qubit and eliminates the corresponding row and column such that they can be removed from the problem.

Our adjustment is described by algorithm 1, where we lift the restrictions for the row and column to be removed such that they don't necessarily intersect at the diagonal. Specifically, we pick the logical qubit corresponding to the row (i.e., the pivot row), and a column (i.e., the pivot column) to be the new register for that logical qubit. Then, we can eliminate both the pivot column and row through a sequence of row operations such that they become basis vectors. In the meantime, *PermRowCol* uses subroutines described in Appendix A, whose behaviour is articulated below.

Given a parity matrix M to synthesize over a connectivity graph G , the vertices of G correspond to the numbering of rows in M . Before the synthesis, the original qubit map indicates that the logical qubit i is stored in the physical qubit register i , which corresponds to the vertex i .

At the start of each round, pick the logical qubit i that we want to remove from the problem. The only constraint is that it needs to be non-cutting for G . Among the non-cutting qubits, we use Algorithm 2 to determine which row to eliminate. For our purpose, a simple heuristic is used: selecting the row with the lowest hamming weight in M . This gives us the pivot row i .

Next, we use Algorithm 3 to choose the pivot column to eliminate. Suppose column j is picked for row i . This means logical qubit i will be stored in the physical qubit register j after this round of elimination. Any column should work as long as it has a 1 on row i and does not have a qubit assigned to it yet. Among all the candidate columns, here we choose the column with the lowest hamming weight.

Note that in principle, we can choose any arbitrary row and column, as long as the vertex corresponding to the pivot row does not disconnect G after it is removed. Moreover, the pivot row and column should not have been picked before. This means Algorithms 2 and 3 could be replaced by other heuristics

so long as the above constraints are satisfied. We discuss the implications of these choices in Section 5 and a possible improvement in Section 6.

With the pivot row and column, we can add rows together such that the pivot row and column only have a 1 at their intersection in M , and 0's everywhere else. This means that they are eliminated. More precisely, in Algorithm 4, we start with the pivot column and gather all its rows that are non-zero. Let it be S . Then, build a Steiner tree with the pivot row as the root and the nodes in S as terminals. Starting at the leaves, for each Steiner node in the tree, add its child to it. By construction, the Steiner nodes correspond to the rows that are zero in the pivot column. Thus, this method will make all Steiner nodes equal to 1. Next, traverse the tree again from the leaves to the root and add every parent to its child. This results in a matrix with all zeros in the pivot column except for the pivot row.

Afterwards, Algorithm 5 carries out a similar elimination process for the pivot row. First, we need to find which rows to add together such that the entire pivot row is filled with 0's except for the pivot column. This could be done by solving a system of linear equations defined by M . Let the set of rows to be added be T . Then, we can construct a new Steiner tree with the pivot row as the root and the rows in T as terminals. Again we traverse the tree twice: once top-down while adding Steiner nodes to their parents, and once bottom-up while adding all nodes to their parents. As a result, all rows in T are added to the pivot row, and thus it is turned into a basis vector.

After column j and row i are eliminated to basis vectors, they are removed from the parity matrix. Accordingly, vertex i is removed from the connectivity graph. We can update the output qubit map to indicate that logical qubit i is now stored in the physical qubit register j after this round of elimination. As discussed in Section 2.3, Algorithms 4 and 5 are adapted from the algorithm *RowCol* in [27].

In summary, Algorithm 1 constructs Steiner trees based on the rows of M and generates CNOT based up row operations. Once a row in M becomes an identity row (i.e. only contains a single 1), this row is eliminated and we can remove the corresponding vertex from the connectivity graph. Then we restart the algorithm on a smaller problem, until the updated graph has no vertex. This ensures the termination of the algorithm.

The time complexity of Algorithm 1-*PermRowCol* depends on the choice of heuristics in Algorithms 2 and 3, as well as the choice of the (approximate) Steiner tree algorithm when eliminating a row and column in Algorithms 4 and 5. The remaining time complexity is dominated by calculating the non-cutting vertices. Thus, given N qubits and a connectivity graph with E edges, the time complexity for *PermRowCol* is as follows.

$$\begin{aligned} O(\text{PermRowCol}) &= O(N)(O(\text{NonCuttingVertices}) + O(\text{ChooseRow}) + O(\text{ChooseColumn}) + O(\text{EliminateColumn}) + O(\text{EliminateRow})) \\ &= O(N)(O(\text{NonCuttingVertices}) + O(\text{ChooseRow}) + O(\text{ChooseColumn}) + 2 * O(\text{Steinertree})) \end{aligned}$$

The suggested *ChooseRow* and *ChooseColumn* heuristics both have time complexity $O(N^2)$, but these can be replaced by other heuristics with improved complexities. In the meantime, the complexity of the Steiner tree algorithm depends heavily on the choice of the algorithm. Building Steiner trees is NP-hard but the topologies being benchmarked are sparse enough that approximate methods perform well. The algorithm we use is based on Prim's and Floyd-Warshall's algorithms and it has time complexity $O(N^3)$. Thus, the time complexity for our specific implementation of *PermRowCol* is

$$O(\text{PermRowCol}) = O(N)(O(N^2 + EN) + 2 * O(N^2) + 2 * O(N^3)) = O(N^4).$$

Algorithm 1: PermRowCol

```

Input : Parity matrix  $M$  and topology  $G(V_G, E_G)$  with labels corresponding to the rows of  $M$ 
Output: CNOT circuit  $C$  and output qubit map  $P$ 
 $P \leftarrow [-1 \dots -1]$ ; /*  $|V_G|$  times */
 $C \leftarrow$  New empty circuit;
while  $|V_G| > 1$  do
  // Find non-cutting qubits  $V_s$  that can (still) be removed  $G$ .
   $V_s \leftarrow$  NonCuttingVertices( $G$ );
   $r \leftarrow$  ChooseRow( $V_s, M$ );
  // Choose a physical qubit register to map  $r$  to.
   $c \leftarrow$  ChooseColumn( $M, r, [i : i \in [1 \dots |P|]$  where  $P[i] = -1]$ );
   $Nodes \leftarrow [i : i \in V_G$  where  $M_{i,c} = 1]$ ;
   $C.add(\text{EliminateColumn}(M, G, r, Nodes))$ ;
  // Reduce the row if it is not yet eliminated.
  if  $\sum_{j \in [1 \dots |P|]} M_{r,j} > 1$  then
     $A \leftarrow M$  without row  $r$  and without column  $c$ ;
     $B \leftarrow M[r]$  without column  $c$ ;
     $X \leftarrow A^{-1}B$ ; /* Find rows to add to eliminate row  $r$  */
     $Nodes \leftarrow [i : i \in V_G$  where  $i = r$  or  $X[Index(i)] = 1]$ ;
     $C.add(\text{EliminateRow}(M, G, r, Nodes))$ ;
  end
  // Update the output qubit map.
   $P[c] \leftarrow r$ ;
   $G \leftarrow$  subgraph of  $G$  with vertex  $r$  and connecting edges removed;
end
// The loop ends with 1 qubit in  $G$ .
// Update the map for the last output qubit.
 $i \leftarrow [i : i \in [1 \dots |P|]$  where  $P[i] = -1]$ ; /* Find index with -1 */
 $P[i] \leftarrow V_G[0]$ ;
return  $C, P$ 

```

4 Benchmarking Results

We benchmark our algorithm against *Steiner-Gauss* [14] and *RowCol* [27] to demonstrate the advantage of dynamically remapping qubits during synthesis, both with and without *Reverse Traversal* (RT). To do this, we generate CNOT circuits with q qubits and d CNOTs that are sampled uniformly at random. The final dataset consists of 100 such CNOT circuits per (q, d) -pair. The implementation of our proposed algorithm, the benchmark algorithms, the CNOT circuit generation script, unit tests, as well as the dataset of random CNOT circuits can be found on GitHub¹. The specific script that ran our experiments can also be found there².

The number of CNOTs in the routed circuit versus the number of CNOTs in the original circuit are plotted in Figures 6, 7, 8 and 10. The blue line $x = y$ serves as the baseline to compare the routing

¹<https://github.com/Aerylia/pyzx/tree/rowcol>

²<https://github.com/Aerylia/pyzx/blob/rowcol/demos/PermRowCol%20results.ipynb>

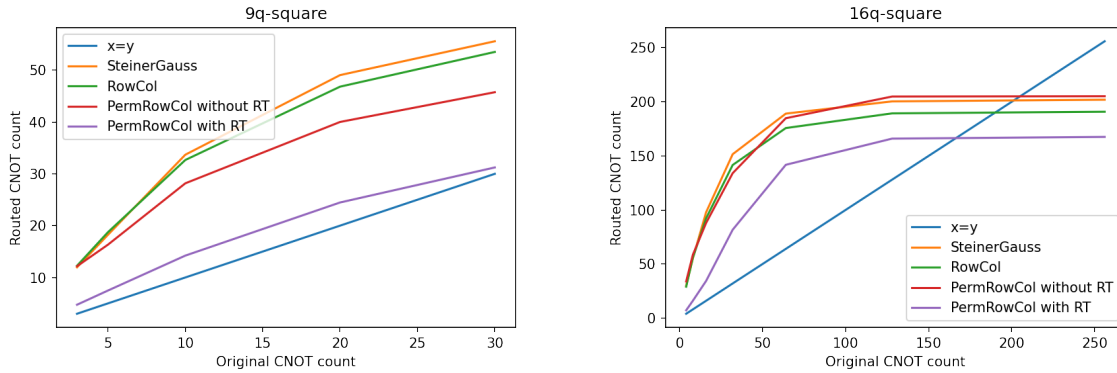


Figure 6: Compare the number of CNOTs generated by *Steiner-Gauss* [14] (orange), *RowCol* [27] (green), and *PermRowCol* without *Reverse Traversal* (RT) (red) and with RT (purple) for different fictitious square grid topologies: 9-qubit square grid (left) and 16-qubit square grid (right). The blue line $x = y$ serves as the baseline to compare the routing overhead of different algorithms. If a point is above the blue line, the routed circuit requires more CNOTs than the original circuit. If a point is below the blue line, then otherwise.

overhead of different algorithms. If a point is above the blue line, the routed circuit requires more CNOTs than the original circuit. This is expected in practice, but we want as few CNOTs as possible. On the other hand, when the original CNOT circuit has many CNOTs, it is possible for some algorithm to synthesize a circuit with fewer CNOTs than that of the original circuit. This implies that the original circuit contains redundant CNOTs which are removed by the algorithm. This happens because after a certain amount of CNOTs, the parity matrix representing the circuit becomes a random matrix and synthesizing a random parity matrix requires a constant amount of CNOTs, as discussed in [19].

Figure 6 compares the performance of the three algorithms on two fictitious square grid topologies, whereas Figure 7 compares their performance on three real device topologies. The corresponding connectivity graphs are shown in Figure 9. Overall, the proposed *PermRowCol* algorithm outperforms the other algorithms when the CNOT count is small. For a large number of CNOTs, it depends on the topology whether the proposed algorithm still outperforms the others. Possible reasons for this are discussed in Section 5.

When the algorithm *PermRowCol* is combined with the *Reverse Traversal* (RT) strategy, it outperforms the other algorithms on all architectures, as shown by the purple line in Figures 6 and 7. Note that because algorithms *SteinerGauss* and *RowCol* does not remap the qubits, RT would have no effect on them.

5 Discussion and Conclusion

In this paper, we propose the algorithm *PermRowCol* that synthesizes CNOT circuits from a parity matrix while respecting the connectivity constraints of a quantum computer. It dynamically remaps logical qubits to different physical qubit registers. This technique is designed for the global optimization of quantum circuits. Therefore, our technique can add improvements that cannot be found with local optimization methods. Our work is based on the observation that allowing dynamic qubit maps during the circuit synthesis may reduce the output CNOT count. We provide a recipe to construct the improved al-

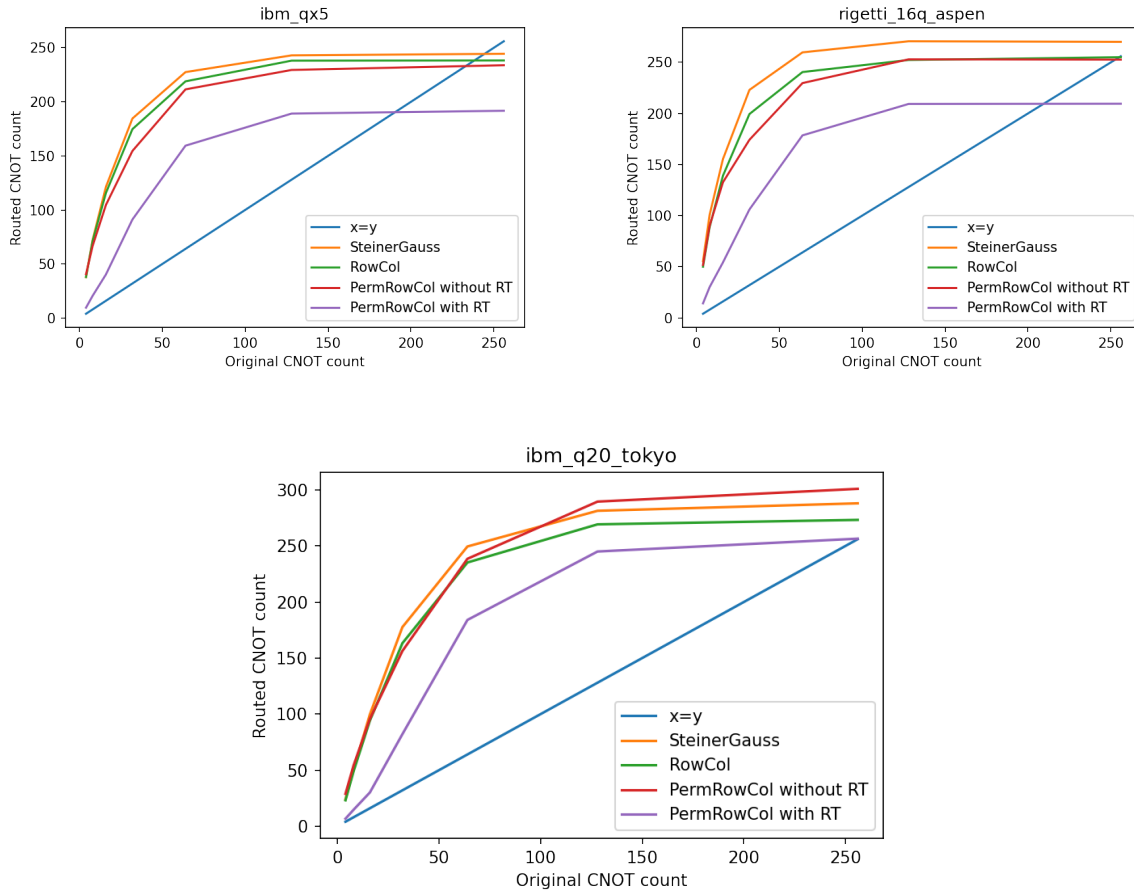


Figure 7: Compare the number of CNOTs generated by *Steiner-Gauss* [14] (orange), *RowCol* [27] (green), and *PermRowCol* without *Reverse Traversal* (RT) (red) and with RT (purple) for different real topologies: 16-qubit IBM QX5 (left), 16-qubit Rigetti Aspen (right), and 20-qubit IBM Tokyo (middle). The blue line $x = y$ serves as the baseline to compare the routing overhead of different algorithms. If a point is above the blue line, the routed circuit requires more CNOTs than the original circuit. If a point is below the blue line, then otherwise.

gorithm whose performance has been confirmed by the benchmarking results. We show that in some but not all cases, the qubit remap results in a smaller CNOT overhead compared to the other state-of-the-art algorithms. Moreover, adding the *Reverse Traversal* (RT) technique improves the results noticeably.

Looking ahead, many problems remain open. For example, in some cases, the *PermRowCol* performs worse than the original *RowCol*, which differs from the *PermRowCol* because we pick a different row and column to eliminate rather than the ones intersecting at the diagonal. This flexibility results in the remap of qubits. If the *PermRowCol* produces more CNOT overhead than that of *RowCol*, we may have picked an inefficient remap. Thus, the simple heuristics used in our algorithm may not be optimal when synthesizing a random parity matrix. This makes sense because our heuristic relies on the number of 1s in each row and column; but in a random parity matrix, the number of 1s in each row and column is approximately the same. Therefore, the *PermRowCol* with our choice of heuristic may decide to eliminate a random row and column that requires more CNOTs later in the synthesis process.

Besides, there seems to be some connections between the underlying topologies and different algorithms’ performance. In Figure 9, the 16-qubit square grid machine, the IBM QX5 machine, and the Rigetti Aspen machine all consist of 16 qubits, but have distinct connectivity constraints. In Figure 6, to synthesize large CNOT circuits (≥ 200 CNOTs) given the 16-qubit square grid (Figure 9.c), *PermRowCol* without RT generates the most CNOTs. For a circuit of arbitrary size and for all topologies in Figure 9, *PermRowCol* with RT generates the least CNOTs (Figure 7). In the meantime, Table 1 shows that the topologies of the real devices are sparser than their fictitious counterparts. This means that when removing random non-cutting vertices from the connectivity graph, our options are much more limited when working with real devices. Conversely, removing random non-cutting vertices from a square grid will restrict our options less because the vertices are connected through more paths. This makes it less likely to pick an inefficient option. However, when synthesizing circuits without any connectivity constraints, Figure 8 shows that *PermRowCol* outperforms other algorithms even when it is not combined with RT. It seems that picking an inefficient qubit allocation is less detrimental when CNOTs are allowed between arbitrary pairs of qubits.

Qubit Count	Topology		Avg. Graph Distance	Average Degree
16	Fictitious devices	Square	2.5	4
		Fully-Connected	1	15
	Real devices	Rigetti Aspen	3.25	2.25
		IBM QX5	3.125	2.75

Table 1: Different topologies in Figure 9 have distinct average graph distance and average vertex degree. The topologies of real devices (i.e., Rigetti Aspen and IBM QX5) have greater average graph distance and smaller average vertex degree than those of fictitious devices (i.e., Square and Fully-Connected). In our example, the topologies of real devices are sparser than their fictitious counterparts.

Based upon the above observations, we expect that the heuristics for picking the pivot row and column can be further improved. Since we need to pick the pivot row and column in an optimal order, this is a combinatorial search space. For completeness, we have implemented an A* algorithm [21] for the choice of pivots that tries all possible pivots following Dijkstra’s shortest path algorithm. These results and their technical details can be found in Appendix E. Since this method scales exponentially with respect to the number of qubits, we do not think this is a sensible approach. This is why it is not in the main body of this paper. The A* algorithm is only added to indicate the effect of a “perfect” heuristic.

In Appendix E, we show that the A* algorithm performs marginally worse than RT on its own on 5-qubit devices. This seems to indicate that it is equally important for *PermRowCol* to have a good initial qubit mapping as having a good heuristic. This effect is diminished when *PermRowCol* is applied on multiple CNOT slices in a circuit. Because the initial qubit mapping cascades throughout the repeated applications of *PermRowCol* that uses the heuristic for every slice.

Additionally, it is likely that the performance of alternative heuristics depends heavily on the given topology and parity matrix. Therefore, we encourage the reader to think about what heuristics would work well for their specific use case.

In conclusion, we need a better method to determine the dynamic qubit maps than the heuristic we have used in this paper. This is also why we have not yet compared the performance of *PermRowCol* with those in the established quantum compilers such as Qiskit, TKET, and SABRE. We will discuss possible directions for improvements in the next section.

6 Future Work

Our *PermRowCol* algorithm is shown to be promising when combined with Reverse Traversal (RT). In Section 6.1, we discuss ways to improve the *PermRowCol*. In Section 6.2, we illustrate how to leverage the *PermRowCol* to synthesize an arbitrary quantum circuit.

6.1 Improve the PermRowCol algorithm

In Section 5, we conclude that a better heuristic is needed for choosing the row and column to eliminate. Moreover, the quality of the heuristics may depend on the given topology and the original CNOT circuit. Beyond this, to achieve the full-stack quantum circuit compilation, it is important to extend our method by accounting for different gate error rates and the difficulty to couple two qubits. When constructing a Steiner tree, we can define a non-uniform weight function based on the quality of the gates acting on each qubit and the prevalent error model in the chosen architecture. This insight may inspire some interesting error-mitigation techniques. Moreover, due to the dynamic qubit maps in the *PermRowCol*, we might redesign the algorithm to take into account the quality of the single qubit gates that are executed after the CNOT circuit. Under such restrictions, some qubits cannot be mapped to certain registers. These kinds of selection rules can then be added to the *PermRowCol* by changing the heuristic for choosing which column to eliminate.

Additionally, the *PermRowCol* can be improved by adding a blockwise elimination method [19]. In [27], in addition to the algorithm *RowCol*, the *size-block elimination* (SBE) was introduced. It uses a similar strategy as did in [19] but with Steiner trees and Gray codes. Eliminating blocks of rows and columns might not be very efficient on sparse graphs, but it might be interesting as an alternative to the unconstrained Gaussian elimination tasks where a permutation matrix is accepted as a valid solution (e.g. ZX-diagram extraction [5]).

6.2 Extension PermRowCol to synthesize arbitrary quantum circuits

To achieve the universality for quantum computing, we need to work with unitaries beyond just CNOT gate. Therefore, a natural next step for us is to extend the *PermRowCol* algorithm such that we can route and map qubits in any quantum circuit. There are two potential approaches: (1) synthesize the full circuit from a flexible representation; (2) cut the circuit into pieces so that we can synthesize each subcircuit separately and glue the synthesized pieces back together. In what follows, we discuss these strategies by building upon our *PermRowCol* algorithm to work with more general quantum circuits until we end up with the set of universal quantum circuits.

Since the *PermRowCol* can only synthesize circuits over CNOT, we start by extending it to work with the class of circuits over CNOT and R_z rotations. These circuit can be characterized by *phase polynomials*, which are described by the set of parity terms where each R_z occurs, along with a parity matrix describing the output parity terms of the quantum circuit [3, 2]. This notion is also known the *sum-over-paths*. Various Steiner-tree-based methods have been proposed to synthesize the parity term of each R_z gate [17, 11, 10, 25]. The remaining parity matrix can then be synthesized by *PermRowCol*. To extend the phase polynomials to arbitrary quantum circuits, we need to add the H gate which these methods cannot synthesize. To this end, we can cut the circuit into subcircuits at the locality of H gates, and each subcircuit is composed of CNOT and R_z gates. Then our problem is reduced to synthesizing the phase polynomial of each subcircuit and then glue them back together. This method is known as the *slice-and-build*, and it is proposed in [10]. *PermRowCol* can make a difference because the dynamic qubit

maps allows this algorithm to move the H gates so that they could act on different but more convenient quantum registers.

Moreover, the presence of a NOT gate is compatible with the current characterization because a NOT gate on some physical qubit introduces an 1 modulo 2 to the corresponding parity term. Thus, we can further extend the above method to work with circuits over NOT, CNOT, and R_z rotations by adding an extra row to the parity matrix to represent the participation of a NOT gate. Then we can proceed as before using the phase-polynomial network synthesis and the slice-and-build to synthesize a circuit over NOT, CNOT, and R_z rotations [10]. This is much simpler than first synthesizing the phase polynomial without the NOT gates, replacing each NOT by $HR_z(\pi)H$, and then synthesizing the latter phase polynomial. Accordingly, we have generalized *PermRowCol* to synthesize the Clifford+T circuits, which is a family of circuits that are well-suited for universal quantum computation.

Furthermore, it is worth investigating how to combine *PermRowCol* with a generalized notion of the sum-over-paths: *Pauli exponentials* [9]. Like phase polynomials, the Pauli exponential keeps track of the parity terms where each R_z rotation occurs. This is done by using *Pauli strings* over I, X, Y, Z rather than using the binary strings. In [9], an algorithm is proposed to extract a circuit from the Pauli exponential form. This is done by adding Clifford gates to the circuit until the Pauli exponential is reduced to a phase polynomial. Although the algorithm is not architecture-aware, it is possible to adjust the algorithm such that it only generates CNOTs that are allowed by the target topology, and the phase polynomial can be synthesized using the methods described above. Additionally, the algorithm was created for synthesizing particular quantum chemistry circuits, but it can be argued that the Pauli exponentials should serve as an important primitive for quantum computation in general [16].

Lastly, we can use *PermRowCol* in the extraction of ZX diagrams [5]. ZX calculus is universal for quantum computation, so any quantum circuit can be expressed as a ZX diagram. Then, we can make the diagram into a normal form from which to extract an optimized circuit. The Gaussian elimination is used in the extraction procedure, and it can be replaced by the *PermRowCol*. Even though the extraction procedure doesn't take any topologies into account, it may be interesting to use the *PermRowCol* with a fully-connected graph. In ZX-calculus, only connectivity matters, so reordering the outputs of the extracted circuit is equivalent to bending wires and therefore free. Thus, it is better to end up with crossing wires than to extract CNOTs. In Figure 8, we show that the *PermRowCol* results in less CNOTs compared to the Gaussian elimination and the *RowCol*. Therefore, the *PermRowCol* could be beneficial to improve the overhead of quantum circuit synthesis.

Acknowledgements

The authors would like to thank Jukka K. Nurminen, Douwe van Gijn, and Dustin Meijer for proof-reading. They also wish to thank Matthew Amy, Neil J. Ross, and John van de Wetering for helpful feedback on choosing the paper's title.

References

- [1] Gernot Alber, Thomas Beth, Michał Horodecki, Paweł Horodecki, Ryszard Horodecki, Martin Rötteler, Harald Weinfurter, Reinhard Werner, Anton Zeilinger, Thomas Beth et al. (2001): *Quantum algorithms: Applicable algebra and quantum physics*. *Quantum information: an introduction to basic theoretical concepts and experiments*, pp. 96–150, doi:10.1007/3-540-44678-8_4.
- [2] Matthew Amy, Parsiad Azimzadeh & Michele Mosca (2018): *On the controlled-NOT complexity of controlled-NOT-phase circuits*. *Quantum Science and Technology* 4(1), doi:10.1088/2058-9565/aad8ca.

- [3] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953.
- [4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell et al. (2019): *Quantum supremacy using a programmable superconducting processor*. *Nature* 574(7779), pp. 505–510, doi:10.1038/s41586-019-1666-5.
- [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, doi:10.22331/q-2021-03-25-421.
- [6] Timothée Goubault de Brugière, Marc Baboulin, Benoît Valiron, Simon Martiel & Cyril Allouche (2020): *Quantum CNOT circuits synthesis for NISQ architectures using the syndrome decoding problem*. In: *Quantum CNOT circuits synthesis for NISQ architectures using the syndrome decoding problem*, Springer, pp. 189–205, doi:10.1007/978-3-030-52482-1_11.
- [7] Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoß & Laura Sanità (2013): *Steiner tree approximation via iterative randomized rounding*. *Journal of the ACM (JACM)* 60(1), doi:10.1145/2432622.2432628.
- [8] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest & Clifford Stein (2022): *Introduction to algorithms*. MIT press.
- [9] Alexander Cowtan, Will Simmons & Ross Duncan (2020): *A generic compilation strategy for the unitary coupled cluster ansatz*. *arXiv preprint*. arXiv:2007.10515, doi:10.48550/arXiv.2007.10515.
- [10] Vlad Gheorghiu, Jiaxin Huang, Sarah Meng Li, Michele Mosca & Priyanka Mukhopadhyay (2022): *Reducing the CNOT count for Clifford+ T circuits on NISQ architectures*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, doi:10.1109/TCAD.2022.3213210.
- [11] Arianne Meijer-van de Griend & Ross Duncan (2020): *Architecture-aware synthesis of phase polynomials for NISQ devices*. *arXiv preprint*. arXiv:2004.06052, doi:10.48550/arXiv.2004.06052.
- [12] Frank K Hwang & Dana S Richards (1992): *Steiner tree problems*. *Networks* 22(1), doi:10.1007/0-306-48332-7_489.
- [13] Richard M Karp (1972): *Reducibility among combinatorial problems*. In: *Complexity of computer computations*, Springer, pp. 85–103, doi:10.1007/978-1-4684-2001-2_9.
- [14] Aleks Kissinger & Arianne Meijer van de Griend (2020): *CNOT circuit extraction for topologically-constrained quantum memories*. *Quantum Information and Computation* 20(7-8). arXiv:1904.00633, doi:10.48550/arXiv.1904.00633.
- [15] Gushu Li, Yufei Ding & Yuan Xie (2019): *Tackling the qubit mapping problem for NISQ-era quantum devices*. In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1001–1014. arXiv:1809.02573, doi:10.1145/3297858.3304023.
- [16] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding & Yuan Xie (2022): *Paulihedral: a generalized block-wise compiler optimization framework for Quantum simulation kernels*. In: *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 554–569, doi:10.1145/3503222.3507715.
- [17] Beatrice Nash, Vlad Gheorghiu & Michele Mosca (2020): *Quantum circuit optimizations for NISQ architectures*. *Quantum Science and Technology* 5(2). arXiv:1904.01972, doi:10.1088/2058-9565/ab79b1.
- [18] Michael A Nielsen & Isaac L Chuang (2001): *Quantum computation and quantum information*. *Phys. Today* 54(2), p. 60, doi:10.1017/CBO9780511976667.
- [19] Ketan N Patel, Igor L Markov & John P Hayes (2004): *Optimal synthesis of linear reversible circuits*. *Quantum Information & Computation* 8(3), doi:10.48550/arXiv.quant-ph/0302002.
- [20] Gabriel Robins & Alexander Zelikovsky (2005): *Tighter bounds for graph Steiner tree approximation*. *SIAM Journal on Discrete Mathematics* 19(1), doi:10.1137/S0895480101393155.
- [21] Stuart Russell & Peter Norvig (2010): *Artificial Intelligence: A Modern Approach*, 3 edition. Prentice Hall.

- [22] Vivek V Shende, Aditya K Prasad, Igor L Markov & John P Hayes (2003): *Synthesis of reversible logic circuits*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(6), pp. 710–722, doi:10.1109/TCAD.2003.811448.
- [23] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *$t|ket\rangle$: a retargetable compiler for NISQ devices*. *Quantum Science and Technology* 6(1). arXiv:2003.10611, doi:10.1088/2058-9565/ab8e92.
- [24] Roberto Stassi, Mauro Cirio & Franco Nori (2020): *Scalable quantum computer with superconducting circuits in the ultrastrong coupling regime*. *npj Quantum Information* 6(1). arXiv:1910.14478, doi:10.1038/s41534-020-00294-x.
- [25] Vivien Vandaele, Simon Martiel & Timothée Goubault de Brugière (2022): *Phase polynomials synthesis algorithms for NISQ architectures and beyond*. *Quantum Science and Technology*. arXiv:2104.00934, doi:10.1088/2058-9565/ac5a0e.
- [26] Robert Wille, Rod Van Meter & Yehuda Naveh (2019): *IBM’s Qiskit tool chain: Working with and developing for real quantum computers*. In: *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 1234–1240, doi:10.23919/DATE.2019.8715261.
- [27] Bujiao Wu, Xiaoyu He, Shuai Yang, Lifu Shou, Guojing Tian, Jialin Zhang & Xiaoming Sun (2023): *Optimization of CNOT circuits on limited-connectivity architecture*. *Physical Review Research* 5(1), p. 013065, doi:10.1103/PhysRevResearch.5.013065.
- [28] Xiangzhen Zhou, Yuan Feng & Sanjiang Li (2022): *A Monte Carlo Tree Search Framework for Quantum Circuit Transformation*. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*. arXiv:2008.09331, doi:10.1145/3514239.
- [29] Qingling Zhu, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong et al. (2022): *Quantum computational advantage via 60-qubit 24-cycle random circuit sampling*. *Science bulletin* 67(3), pp. 240–245, doi:10.1016/j.scib.2021.10.017.

A Subroutines

For reference, this section contains the pseudo-code of all subroutines used in Algorithm 1.

Algorithm 2: ChooseRow: Subroutine for choosing which row to eliminate.

Input : Parity matrix M , candidate vertices V_s that correspond to rows in M
Output: The row to eliminate: the pivot row
 // Pick the row with the least 1s in M .
 $row \leftarrow \operatorname{argmin}_{v \in V_s} (\sum M[v]);$
return *pivot row*

Algorithm 3: ChooseColumn: Subroutine for choosing which column to eliminate.

Input : Parity matrix M , candidate columns in M , the pivot row
Output: The column to eliminate: the pivot column
 // For columns with an 1 in the pivot row, pick the one with the least 1s.
 $row \leftarrow \operatorname{argmin}_{c \in \text{ColsToEliminate}} (\sum M[:,c] \text{ if } M[\text{row}][c] = 1 \text{ else } |\text{ColsToEliminate}|);$
return *pivot column*

Algorithm 4: EliminateColumn: Subroutine for eliminating a column. Here, *SteinerTree* is a routine which creates a Steiner tree over graph G with root $root$ and nodes $Terminals$. *BottomUpTraversal* performs a post-order traversal on the given tree and returns the edges rather than the vertices.

Input : Parity matrix M , graph G representing the connectivity constraints, $root$ vertex, $Terminals$ to build the Steiner tree over

Output: List of CNOTs C to add to the circuit.

```

C ← [];
Tree ← SteinerTree(G, root, Terminals);
for edge ∈ BottomUpTraversal(Tree) do
    if M[edge[0]][col] = 0 then
        C.add(CNOT(edge[0], edge[1]));          /* Make Steiner nodes into 1s */
        M[edge[0]] ← M[edge[0]] + M[edge[1]] mod 2;
    end
end
for edge ∈ BottomUpTraversal(Tree) do
    C.add(CNOT(edge[1], edge[0]));          /* Make the column into identity */
    M[edge[1]] ← M[edge[0]] + M[edge[1]] mod 2;
end
return C

```

Algorithm 5: EliminateRow: Subroutine for eliminating a row. Here, *SteinerTree* is a routine which creates a Steiner tree over graph G with root $root$ and nodes $Terminals$. *TopDownTraversal* performs a pre-order traversal on the given tree and returns the edges rather than the vertices. Similarly, *BottomUpTraversal* performs a post-order traversal on the tree and returns the edges when traversed for the second time.

Input : Parity matrix M , graph G representing the connectivity constraints, $root$ vertex, $Terminals$ to build the Steiner tree over

Output: List of CNOTs C to add to the circuit.

```

C ← [];
Tree ← SteinerTree(G, root, Terminals);
for edge ∈ TopDownTraversal(Tree) do
    if edge[1] ∉ Nodes then
        C.add(CNOT(edge[0], edge[1]));
        M[edge[0]] ← M[edge[0]] + M[edge[1]] mod 2;
    end
end
for edge ∈ BottomUpTraversal(Tree) do
    C.add(CNOT(edge[0], edge[1]));
    M[edge[0]] ← M[edge[0]] + M[edge[1]] mod 2;
end
return C

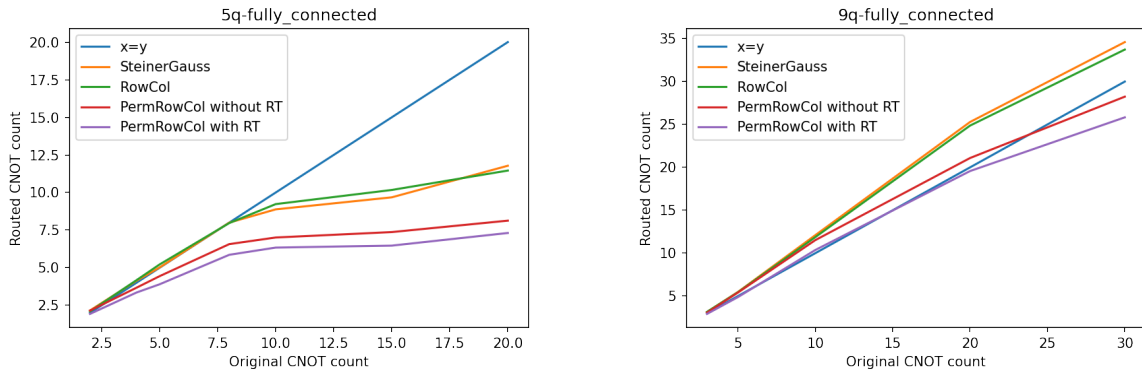
```

B Unconstrained performance

As additional information, Figure 8 compares the algorithm performance in the unconstrained case, i.e. synthesizing a CNOT circuit given a fully connected topology. Although this use case does not include the routing of CNOTs, there are two use cases for which this is interesting. Both of these cases need to allow reallocation of qubits to make use of *PermRowCol*. Restricting *PermRowCol* to a fixed allocation is equivalent to the *RowCol* algorithm.

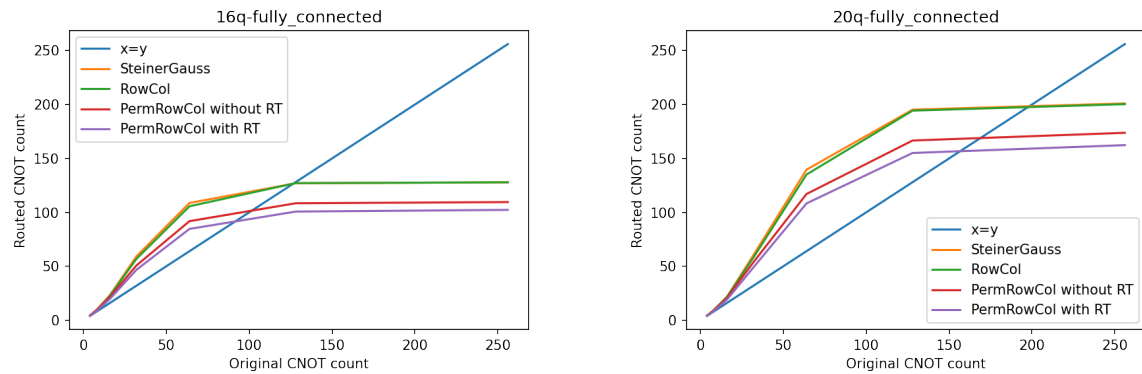
First of all, we can use *PermRowCol* without topological restrictions in case we have a circuit with many CNOTs (with respect to the number of qubits) to optimize the number CNOTs.

Secondly, *PermRowCol* can be used in cases when the CNOT circuit is not known beforehand and needs to be synthesized from a given parity matrix. For example, this can be done as part of the *GraySynth* [2] algorithm that is used in the *PauliSynth* [9] algorithm for generating UCCSD circuits in quantum chemistry. Alternatively, it can be used in place of Gaussian elimination as part of ZX-diagram extraction [5]. As we have already discussed in Section 6.2.



(a) Algorithms' performance for the 5-qubit fully connected graph.

(b) Algorithms' performance for the 9-qubit fully connected graph.



(c) Algorithms' performance for the 16-qubit fully connected graph.

(d) Algorithms' performance for the 20-qubit fully connected graph.

Figure 8: These figures show the number of CNOTs generated by *Steiner-Gauss* [14] (orange), *RowCol* [27] (green), and *PermRowCol* (proposed, red) for the unconstrained case, i.e. a complete graph of 5, 9, 16, and 20 qubits where every pair of distinct vertices is connected by a unique edge. The blue $x = y$ -line is used to infer the CNOT overhead.

C Topologies of real devices

We show in Figure 9 the different topologies for the real quantum computers that we used for the connectivity constraints in our experiments.

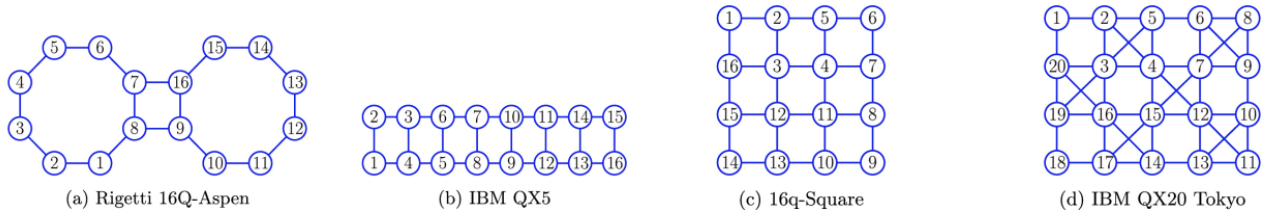


Figure 9: Topologies of existing quantum computers that we use for testing our algorithm. Images are taken from [6].

D Results in table form

For completeness, we show the results from Figure 6 and 7 in the form of Table 2.

CNOT	Topology	<i>Steiner-Gauss</i>	<i>RowCol</i>	<i>PermRowCol</i>	<i>PermRowCol+RT</i>
3	9q-square	11.96 (298.67 %)	12.26 (308.67 %)	12.17 (305.67 %)	4.74 (58.00 %)
5	9q-square	18.23 (264.60 %)	18.74 (274.80 %)	16.33 (226.60 %)	7.48 (49.60 %)
10	9q-square	33.69 (236.90 %)	32.67 (226.70 %)	28.17 (181.70 %)	14.22 (42.20 %)
20	9q-square	49.05 (145.25 %)	46.82 (134.10 %)	40.01 (100.05 %)	24.47 (22.35 %)
30	9q-square	55.58 (85.27 %)	53.52 (78.40 %)	45.75 (52.50 %)	31.23 (4.10 %)
4	16q-square	29.90 (647.50 %)	29.02 (625.50 %)	33.86 (746.50 %)	7.21 (80.25 %)
8	16q-square	55.14 (589.25 %)	55.70 (596.25 %)	58.49 (631.13 %)	15.96 (99.50 %)
16	16q-square	97.85 (511.56 %)	92.24 (476.50 %)	87.95 (449.69 %)	34.34 (114.63 %)
32	16q-square	151.54 (373.56 %)	141.57 (342.41 %)	133.99 (318.72 %)	81.68 (155.25 %)
64	16q-square	189.09 (195.45 %)	175.74 (174.59 %)	184.84 (188.81 %)	141.75 (121.48 %)
128	16q-square	200.40 (56.56 %)	189.32 (47.91 %)	204.89 (60.07 %)	165.97 (29.66 %)
256	16q-square	201.95 (-21.11 %)	190.83(-25.46 %)	205.16(-19.86 %)	167.55(-34.55 %)
4	rigetti 16q aspen	55.59 (1289.75 %)	50.00 (1150.00 %)	51.77 (1194.25 %)	14.17 (254.25 %)
8	rigetti 16q aspen	100.85(1160.63 %)	88.78 (1009.75 %)	90.12 (1026.50 %)	30.13 (276.63 %)
16	rigetti 16q aspen	155.25 (870.31 %)	138.99 (768.69 %)	132.58 (728.63 %)	54.15 (238.44 %)
32	rigetti 16q aspen	223.03 (596.97 %)	199.41 (523.16 %)	174.21 (444.41 %)	106.04 (231.38 %)
64	rigetti 16q aspen	259.77 (305.89 %)	240.51 (275.80 %)	229.71 (258.92 %)	178.55 (178.98 %)
128	rigetti 16q aspen	270.64 (111.44 %)	252.33 (97.13 %)	252.95 (97.62 %)	209.31 (63.52 %)
256	rigetti 16q aspen	270.07 (5.50 %)	255.06 (-0.37 %)	252.69 (-1.29 %)	209.52(-18.16 %)
4	ibm qx5	37.87 (846.75 %)	37.96 (849.00 %)	40.54 (913.50 %)	9.62 (140.50 %)
8	ibm qx5	72.34 (804.25 %)	71.43 (792.88 %)	66.81 (735.13 %)	20.62 (157.75 %)
16	ibm qx5	121.33 (658.31 %)	115.63 (622.69 %)	104.60 (553.75 %)	40.31 (151.94 %)
32	ibm qx5	184.57 (476.78 %)	174.74 (446.06 %)	154.56 (383.00 %)	91.17 (184.91 %)
64	ibm qx5	227.48 (255.44 %)	218.93 (242.08 %)	211.51 (230.48 %)	159.43 (149.11 %)
128	ibm qx5	242.96 (89.81 %)	238.07 (85.99 %)	229.47 (79.27 %)	189.13 (47.76 %)
256	ibm qx5	244.43 (-4.52 %)	238.25 (-6.93 %)	233.83 (-8.66 %)	191.73(-25.11 %)
4	ibm q20 tokyo	24.04 (501.00 %)	23.14 (478.50 %)	28.88 (622.00 %)	6.71 (67.75 %)
8	ibm q20 tokyo	50.58 (532.25 %)	49.09 (513.63 %)	54.29 (578.63 %)	14.72 (84.00 %)
16	ibm q20 tokyo	99.70 (523.13 %)	94.17 (488.56 %)	95.66 (497.88 %)	30.08 (88.00 %)
32	ibm q20 tokyo	177.58 (454.94 %)	163.22 (410.06 %)	156.28 (388.38 %)	82.09 (156.53 %)
64	ibm q20 tokyo	249.51 (289.86 %)	235.23 (267.55 %)	238.52 (272.69 %)	183.99 (187.48 %)
128	ibm q20 tokyo	281.34 (119.80 %)	269.25 (110.35 %)	289.53 (126.20 %)	245.02 (91.42 %)
256	ibm q20 tokyo	288.01 (12.50 %)	273.23 (6.73 %)	300.92 (17.55 %)	256.48 (0.19 %)

Table 2: This table shows the performance of *Steiner-Gauss* [14, 17], *RowCol* [27], *PermRowCol* (proposed), and *PermRowCol* with *Reverse Traversal strategy* (proposed) for different topologies. The shown numbers represent the average CNOT count over 100 circuits and the average CNOT overhead with respect to the original circuit in brackets behind it.

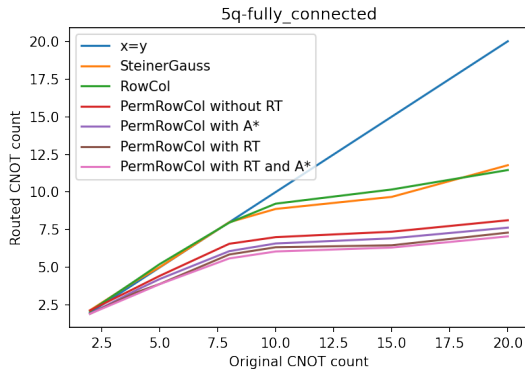
E A* results

In this appendix, we show the results for using the A* algorithm [21] for choosing the row and column in the iterations of *PermRowCol*. We do this for different 5-qubit topologies because the overhead of A* becomes too long for larger topologies.

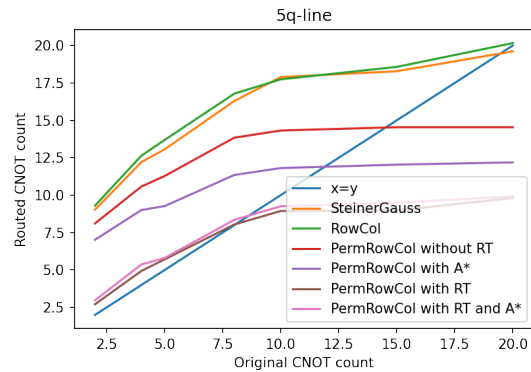
We added the A* algorithm into *PermRowCol* using a priority queue. Initially, we push the original problem into the queue with priority equal to 0. Then, while the queue is not empty, we remove an instance from the queue, reduce the matrix for each combination of chosen row and column and push the resulting smaller problem to the queue with the size of the circuit until now as priority. We continue this process until a solution has been found. By construction, the resulting solution will be the smallest circuit that can be found with *PermRowCol* regardless of the choice in ChooseRow and ChooseColumn heuristics, but the complexity is exponential in the number of qubits.

To reduce the runtime of the algorithm, we restrict the number of new problem instances created at each iteration by a parameter called *choiceWidth*. Instead of expanding each possible combination of row and column, we only expand the top *choiceWidth* options where we rank the options using the original ChooseRow and ChooseColumn heuristics. Where the ChooseRow has priority. Additionally, we limit the size of the queue such that problem instances low in the queue are removed from memory since they probably don't need to be expanded before a solution is found. For these results, we used *choiceWidth*=4 and *max_size*=10. Resulting in an algorithm with time complexity $O(\text{choiceWidth}^{O(\text{PermRowCol})} = O(4^{n^4})$.

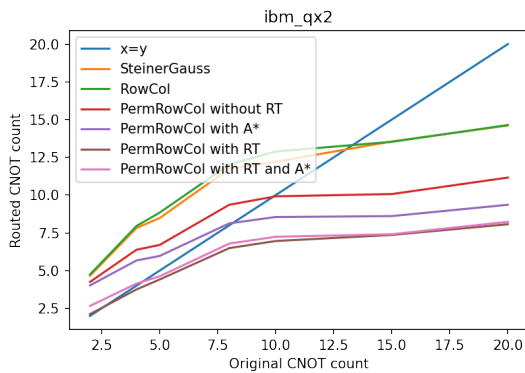
The results are shown in Figure 10 where we see that A* does not perform better than than the Reverse Traversal (RT) strategy, even when combining the two algorithms.



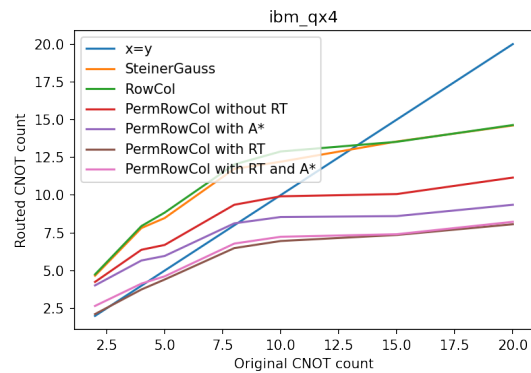
(a) Algorithms' performance for the 5-qubit fully connected graph.



(b) Algorithms' performance for the 5-qubit line graph.



(c) Algorithms' performance for the 5-qubit IBM QX2 device.

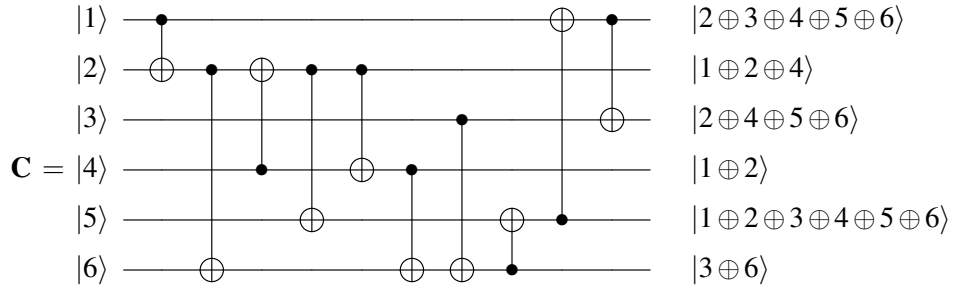


(d) Algorithms' performance for the 5-qubit IBM QX4 device.

Figure 10: These figures show the number of CNOTs generated by *Steiner-Gauss* [14] (orange), *Row-Col* [27] (green), and different variants of *PermRowCol* (proposed) on different 5-qubit topologies. The different variants of *PermRowCol* are the original (red), with A* algorithm (purple), with Reverse Traversal (RT) (brown), and with both A* and RT (pink). The blue $x = y$ -line is used to infer the CNOT overhead.

F Example execution of *PermRowCol*

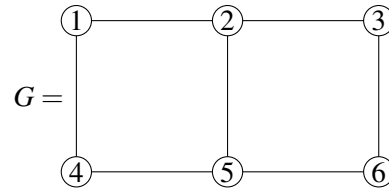
In the section, we execute algorithm *PermRowCol* with inputs in Figure 11. We start with a parity matrix \mathbf{A} to synthesize over the topology graph G , and reduce the problem to eliminating \mathbf{A} to a permutation matrix. The labelling of vertices in G corresponds to the numbering of rows in \mathbf{A} .



(a) A CNOT circuit C composed of 6 qubits.

$$\mathbf{A} = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

(b) The parity matrix of C .



(c) The 6-qubit square grid G .

Figure 11: The CNOT circuit C in figure (a) can be exactly represented by the 6×6 parity matrix A in figure (b). Under the constrained topology G in figure (c), the Steiner-tree based algorithm *PermRowCol* accounts for G and re-synthesizes C .

F.1 Notations

We declare notations that will be used in Appendices F.2 and F.3.

According to algorithm *PermRowCol*, the chosen logical qubit corresponds to a row in A . Its index is denoted by r . The chosen new physical register for r corresponds to a column in A . Its index is denoted by c . V_s is a set of non-cutting vertices of G under which A is synthesized. S is the set of terminal nodes corresponding the indices of all non-zero entries in a row or column. $R(i, j)$ denotes a row operation on A such that row i is added to row j , while row i remains unchanged.

The output qubit allocation Table 3 keeps track of the row and column being selected at each elimination step.

Logical qubit/ r	1	2	3	4	5	6
Physical register/ c						

Table 3: The output qubit allocation table

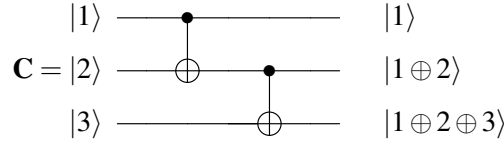
F.2 Parity matrix for a CNOT circuit

Consider a CNOT circuit composed of n qubits, where $CNOT(c, t)$ has control c and target t . Based on the specification in Section 2.1, $CNOT(c, t)$ corresponds to a row operation $R(t, c)$ such that row t is

added to row c , while row t remains unchanged.

In addition to the parity matrix defined in this paper, we note an alternative characterization for CNOT circuits [19, 17, 10]. Let's call it as the *alt-parity matrix* for a CNOT circuit. Similar to the parity matrix characterization, a square matrix is constructed to record the output parities of the circuit. Each row represents a parity term, and each column represents the input qubit. By construction, given a CNOT circuit, its alt-parity matrix is the transpose of its parity matrix. Accordingly, $CNOT(c,t)$ corresponds to a row operation $R(c,t)$ such that row c is added to row t , while row c remains unchanged. However, the synthesis procedure constructs the circuit in reverse.

For example, in Figure 12, we compare the differences between the two notations and the corresponding CNOT synthesis algorithms. In Figure 12.c, we see that the column-wise representation of the parities (Figure 12.b) results in taking row operations that correspond to the CNOTs in order, while adding the target to the control: $R(2,1)R(3,2) \sim CNOT(1,2)CNOT(2,3)$. Alternatively, in Figure 12.e, we see that the row-wise representation of the parities (Figure 12.d) results in taking row operations that correspond to the CNOTs in reverse order, while adding the control to the target: $R(2,3)R(1,2) \sim CNOT(1,2)CNOT(2,3)$.



(a) A CNOT circuit \mathbf{C} composed of 3 qubits and 2 CNOTs. The labels of input qubits are denoted on the left of the circuit. The output parities are denoted on the right of the circuit.

$$\mathbf{A} = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{array}{ccc} 1' & 2' & 3' \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

(b) \mathbf{A} is the parity matrix of \mathbf{C} . Each column represents an output parity term and each row represents an input qubit.

$$\mathbf{A} = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{array}{ccc} 1' & 2' & 3' \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \xrightarrow{R(2,1)} \mathbf{A}' = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{array}{ccc} 1' & 2' & 3' \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \xrightarrow{R(3,2)} I = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{array}{ccc} 1' & 2' & 3' \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

(c) Given \mathbf{A} , CNOT(1,2) corresponds to the row operation $R(2,1)$ and CNOT(2,3) corresponds to the row operation $R(3,2)$.

$$\mathbf{B} = \begin{array}{c} 1' \\ 2' \\ 3' \end{array} \begin{array}{ccc} 1 & 2 & 3 \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$

(d) \mathbf{B} is the alt-parity matrix of \mathbf{C} . Each row represents an output parity term and each column represents an input qubit.

$$\mathbf{B} = \begin{array}{c} 1' \\ 2' \\ 3' \end{array} \begin{array}{ccc} 1 & 2 & 3 \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array} \xrightarrow{R(2,3)} \mathbf{B}' = \begin{array}{c} 1' \\ 2' \\ 3' \end{array} \begin{array}{ccc} 1 & 2 & 3 \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \xrightarrow{R(1,2)} I = \begin{array}{c} 1' \\ 2' \\ 3' \end{array} \begin{array}{ccc} 1 & 2 & 3 \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

(e) Given \mathbf{B} , CNOT(2,3) corresponds to the row operation $R(2,3)$ and CNOT(1,2) corresponds to the row operation $R(1,2)$.

Figure 12: Circuit \mathbf{C} in (a) is composed of CNOT(1,2) and CNOT(2,3). Its output parities are described by the parity matrix \mathbf{A} in (b), or equivalently, by the alt-parity matrix \mathbf{B} in (d). $\mathbf{B}^\top = \mathbf{A}$.

F.3 PermRowCol walkthrough

Elimination step 1 Before the first elimination step, the parity matrix **A** and the constrained topology *G* are shown in Figure 11.

Choose the row and column to eliminate: The set of non-cutting vertices of *G* is $V_s = \{1, 2, 3, 4, 5, 6\}$. Then $r = 1$ and $c = 4$ since

$$\begin{array}{c}
 \begin{array}{cccccccc}
 & 1' & 2' & 3' & 4' & 5' & 6' & \text{Sum} & \text{Row} \\
 1 & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} & 3 & \checkmark \\
 2 & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} & 5 & \\
 3 & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} & 3 & \\
 4 & \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} & 4 & \\
 5 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} & 3 & \\
 6 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & 4 &
 \end{array}
 & \Rightarrow &
 \begin{array}{cccccccc}
 & 1' & 2' & 3' & 4' & 5' & 6' & \\
 1 & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} & & & & & \\
 2 & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} & & & & & \\
 3 & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} & & & & & \\
 4 & \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 5 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 6 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & & & & & \\
 \text{Sum} & \backslash & 3 & \backslash & 2 & 6 & \backslash \\
 \text{Column} & & & & \checkmark & &
 \end{array}
 \end{array}$$

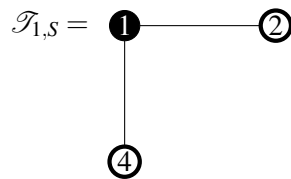
Eliminate the chosen row and column: We start by eliminating column 4' to e_1^\top , then $S = \{1, 2\}$. The Steiner tree $\mathcal{T}_{1,S}$ has root 1 and a set of terminals *S*:



Thus, algorithm *PermRowCol* assigns CNOT(2, 1). It follows that

$$\mathbf{A} = \begin{array}{c}
 \begin{array}{cccccccc}
 & 1' & 2' & 3' & 4' & 5' & 6' & \\
 1 & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} & & & & & \\
 2 & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} & & & & & \\
 3 & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} & & & & & \\
 4 & \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 5 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 6 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & & & & &
 \end{array}
 \xrightarrow{R(1,2)}
 \begin{array}{cccccccc}
 & 1' & 2' & 3' & 4' & 5' & 6' & \\
 1 & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} & & & & & \\
 2 & \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} & & & & & \\
 3 & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} & & & & & \\
 4 & \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 5 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} & & & & & \\
 6 & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & & & & &
 \end{array}
 \end{array} \tag{1}$$

Next, we eliminate row 1 to e_4 . From solving the system of linear equations while emitting column 4', row 1 is formed by rows 2 and 4. Thus $S = \{1, 2, 4\}$. The Steiner tree $\mathcal{T}_{1,S}$ has root 1 and a set of terminals *S*:



Thus, algorithm *PermRowCol* assigns CNOT(1,2)CNOT(1,4). It follows that

$$(1) = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & \xrightarrow{R(2,1)} & \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & \end{matrix} \quad (2)$$

$$\xrightarrow{R(4,1)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & \end{matrix} \quad (3)$$

Update the output qubit allocation: The output qubit allocation after eliminating row 1 and column 4' is updated in Table 4.

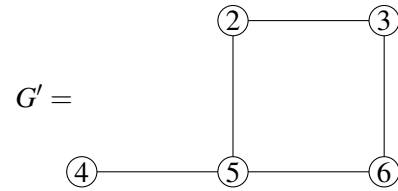
Logical qubit/r	1	2	3	4	5	6
Physical register/c	4					

Table 4: Logical qubit 1 is stored in the physical register 4.

Elimination step 2 After the first elimination step, the parity matrix A' and the constrained topology G' are shown in Figure 13.

$$A' = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} & \end{matrix}$$

(a) The updated parity matrix A' .



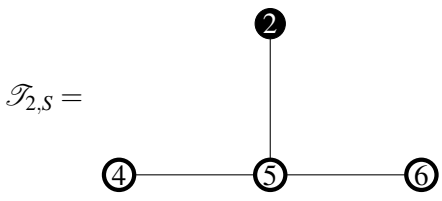
(b) The 5-qubit grid G' .

Figure 13: After elimination step 1, *PermRowCol* eliminates the parity matrix A to A' in (a) under the constrained topology G . Accordingly, G is reduced to G' in (b).

Choose the row and column to eliminate: The set of non-cutting vertices of G' is $V_s = \{2, 3, 4, 6\}$. Then $r = 2$ and $c = 3$ since

$$\mathbf{A}' = \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{cccccc} 1' & 2' & 3' & 4' & 5' & 6' \\ \left(\begin{array}{cccccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{array} \right) \end{array} \begin{array}{c} \text{Sum} \\ \backslash \\ 2 \\ 3 \\ 4 \\ \backslash \\ 4 \end{array} \begin{array}{c} \text{Row} \\ \\ \\ \checkmark \\ \\ \end{array} \Rightarrow \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ \text{Sum} \\ \text{Column} \end{array} \begin{array}{cccccc} 1' & 2' & 3' & 4' & 5' & 6' \\ \left(\begin{array}{cccccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{array} \right) \end{array} \begin{array}{c} \backslash \\ \backslash \\ \backslash \\ \backslash \\ \backslash \\ \backslash \\ 5 \\ \backslash \\ 4 \\ \backslash \\ \backslash \\ \backslash \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \checkmark \\ \\ \\ \\ \end{array}$$

Eliminate the chosen row and column: We start by eliminating column 3' to e_2^I , then $S = \{2, 4, 5, 6\}$.
 The Steiner tree $\mathcal{T}_{2,S}$ has root 2 and a set of terminals S :



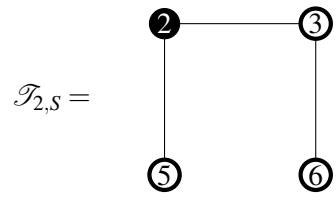
Thus, algorithm *PermRowCol* assigns CNOT(4,5)CNOT(6,5)CNOT(5,2). It follows that

$$\mathbf{A}' = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow{R(5,4)} \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{matrix} \quad (4)$$

$$\xrightarrow{R(5,6)} \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (5)$$

$$\xrightarrow{R(2,5)} \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (6)$$

Next, we eliminate row 2 to e_3 . From solving the system of linear equations while emitting columns 3' and 4', row 2 is formed by rows 3, 5, and 6. Thus $S = \{2, 3, 5, 6\}$. The Steiner tree $\mathcal{T}_{2,S}$ has root 2 and a set of terminals S :



Thus, algorithm *PermRowCol* assigns CNOT(3,6)CNOT(2,3)CNOT(2,5). It follows that

$$(6) = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \xrightarrow{R(6,3)} & \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (7)$$

$$\xrightarrow{R(3,2)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (8)$$

$$\xrightarrow{R(5,2)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (9)$$

Update the output qubit allocation: The output qubit allocation after eliminating row 2 and column 3' is updated in Table 5.

Logical qubit/r	1	2	3	4	5	6
Physical register/c	4	3				

Table 5: Logical qubit 2 is stored in the physical register 3.

Elimination step 3 After the second elimination step, the parity matrix A'' and the constrained topology G'' are shown in Figure 14.

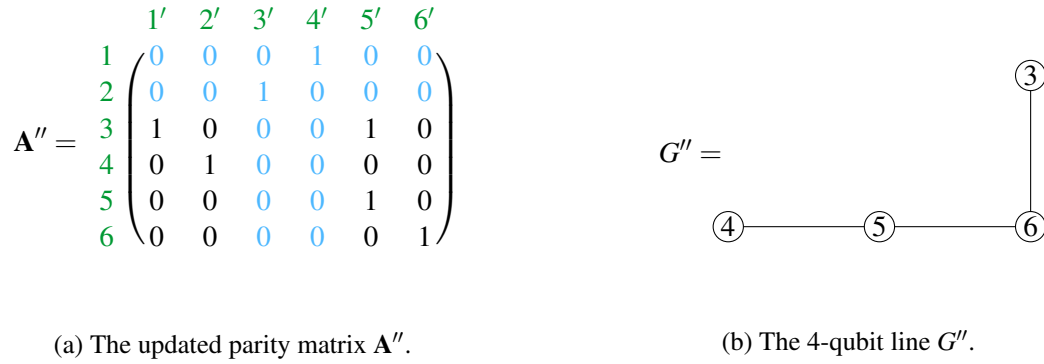


Figure 14: After elimination step 2, *PermRowCol* eliminates the parity matrix \mathbf{A}' to \mathbf{A}'' in (a) under the constrained topology G' . Accordingly, G' is reduced to G'' in (b).

Choose the row and column to eliminate: The set of non-cutting vertices of G'' is $V_s = \{3, 4, 5, 6\}$. Then $r = 4$ and $c = 2$ since

$$\mathbf{A}'' = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} & \text{Sum} & \text{Row} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} \backslash \\ \backslash \\ 2 \\ 1 \\ 1 \\ 1 \end{matrix} & \begin{matrix} \\ \\ \\ \checkmark \\ \\ \end{matrix} \end{matrix} \Rightarrow \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \text{Sum} & \begin{matrix} \backslash \\ \backslash \\ \backslash \\ \backslash \\ \backslash \end{matrix} & \begin{matrix} \\ \\ \\ \\ \checkmark \end{matrix} \\ \text{Column} & & \end{matrix}$$

Eliminate the chosen row and column: In fact, column 2' and row 4 is e_4^T and e_2 respectively, this step is complete.

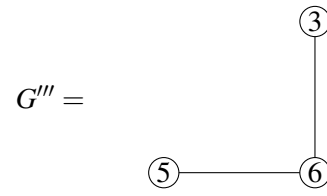
Update the output qubit allocation: The output qubit allocation after eliminating row 4 and column 2' is updated in Table 6.

Logical qubit/r	1	2	3	4	5	6
Physical register/c	4	3		2		

Table 6: Logical qubit 4 is stored in the physical register 2.

Elimination step 4 After the third elimination step, the parity matrix \mathbf{A}''' and the constrained topology G''' are shown in Figure 15.

$$\mathbf{A}''' = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}$$



(a) The updated parity matrix \mathbf{A}''' .

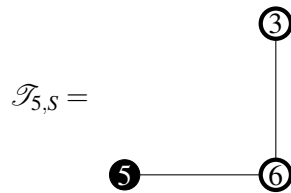
(b) The 3-qubit line G''' .

Figure 15: After elimination step 3, *PermRowCol* eliminates the parity matrix \mathbf{A}'' to \mathbf{A}''' in (a) under the constrained topology G'' . Accordingly, G'' is reduced to G''' in (b).

Choose the row and column to eliminate: The set of non-cutting vertices of G''' is $V_s = \{3, 5\}$.
Then $r = c = 5$ since

$$\mathbf{A}''' = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} & \text{Sum} & \text{Row} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} \backslash \\ \backslash \\ 2 \\ \backslash \\ 1 \\ \backslash \end{matrix} & \begin{matrix} \\ \\ \\ \checkmark \\ \\ \end{matrix} \end{matrix} \Rightarrow \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \text{Sum} & \begin{matrix} \backslash \\ \backslash \\ \backslash \\ \backslash \\ 2 \\ \backslash \end{matrix} & \text{Column} & \begin{matrix} \\ \\ \\ \\ \checkmark \\ \end{matrix} \end{matrix}$$

Eliminate the chosen row and column: We start by eliminating column 5' to e_5^I , then $S = \{3, 5\}$.
The Steiner tree $\mathcal{T}_{5,S}$ has root 5 and a set of terminals S :



Thus, algorithm *PermRowCol* assigns $\text{CNOT}(6,3)\text{CNOT}(3,6)\text{CNOT}(6,5)$. It follows that

$$\mathbf{A}''' = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \xrightarrow{R(3,6)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix} \quad (10)$$

$$\xrightarrow{R(6,3)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix} \quad (11)$$

$$\xrightarrow{R(5,6)} \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (12)$$

Since row 5 is in fact e_5 , this step is complete.

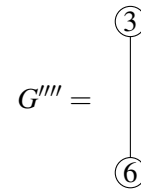
Update the output qubit allocation: The output qubit allocation after eliminating row 5 and column 5' is updated in Table 7.

Logical qubit/r	1	2	3	4	5	6
Physical register/c	4	3		2	5	

Table 7: Logical qubit 5 is stored in the physical register 5.

Elimination step 5 After the fourth elimination step, the parity matrix \mathbf{A}'''' and the constrained topology G'''' are shown in Figure 16.

$$\mathbf{A}'''' = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}$$



(a) The updated parity matrix \mathbf{A}'''' .

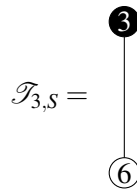
(b) The 2-qubit line G'''' .

Figure 16: After elimination step 4, *PermRowCol* eliminates the parity matrix \mathbf{A}''' to \mathbf{A}'''' in (a) under the constrained topology G''' . Accordingly, G''' is reduced to G'''' in (b).

Choose the row and column to eliminate: The set of non-cutting vertices of G'''' is $V_s = \{3, 6\}$. Then $r = 3$ and $c = 6$ since

$$\mathbf{A}'''' = \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' & \text{Sum} & \text{Row} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} \backslash \\ \backslash \\ 1 \\ \backslash \\ \backslash \\ 2 \end{matrix} & \checkmark & \Rightarrow & \begin{matrix} & 1' & 2' & 3' & 4' & 5' & 6' \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} \backslash \\ \backslash \\ \backslash \\ \backslash \\ \backslash \\ 1 \end{matrix} & \text{Sum} & \text{Column} \\ & & & & & & & & \checkmark \end{matrix}$$

Eliminate the chosen row and column: We start by eliminating column 6' to e_3^T , then $S = \{3, 6\}$. The Steiner tree $\mathcal{T}_{3,S}$ has root 3 and a set of terminals S :



Thus, algorithm *PermRowCol* assigns $\text{CNOT}(6, 3)$. It follows that

$$\mathbf{A}''' = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \xrightarrow{R(3,6)} \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix} \quad (13)$$

Since row 3 is in fact e_6 , this step is complete.

Update the output qubit allocation: The output qubit allocation after eliminating row 3 and column 6' is updated in Table 8.

Logical qubit/r	1	2	3	4	5	6
Physical register/c	4	3	6	2	5	

Table 8: Logical qubit 3 is stored in the physical register 6.

F.4 Output from *PermRowCol*

After the fifth elimination step, *PermRowCol* terminates as there is precisely one vertex left in the constrained topology. The parity matrix \mathbf{A} is reduced to a permutation P . Accordingly, the final output qubit allocation is shown below.

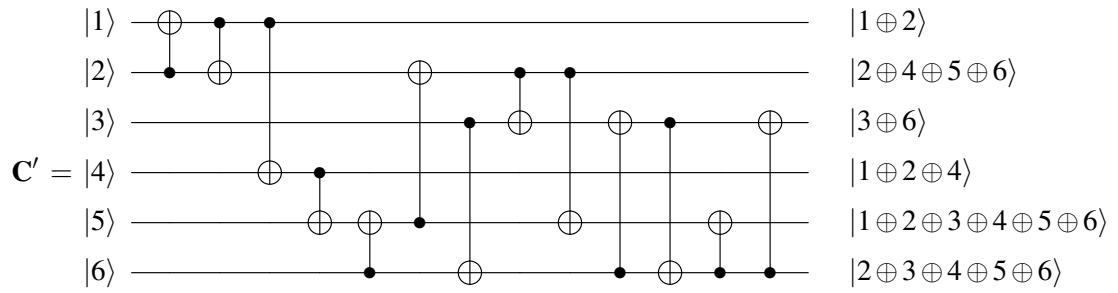
$$P = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

Logical qubit/r	1	2	3	4	5	6
Physical register/c	4	3	6	2	5	1

Table 9: Qubit allocation after resynthesizing the CNOT circuit \mathbf{C} under the constrained topology G .

F.5 Examination of the output validity

By concatenating CNOTs produced from each elimination step, the re-synthesized circuit \mathbf{C}' and the corresponding parity matrix \mathbf{M} is shown in Figure 17. With the input parity matrix \mathbf{A} and the permutation matrix P output by *PermRowCol*, we have $\mathbf{M}P = \mathbf{A}$. This corresponds to the qubit allocation after resynthesizing the CNOT circuit \mathbf{C} over G , as described by Table 9. Hence, our re-synthesized circuit \mathbf{C}' is equivalent to circuit \mathbf{C} up to column permutation specified by P .



(a) The re-synthesized CNOT circuit C' after running *PermRowCol* with input parity matrix and constrained topology defined in Figure 11.

$$\mathbf{M} = \begin{matrix} & \begin{matrix} 1' & 2' & 3' & 4' & 5' & 6' \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

(b) \mathbf{M} is the parity matrix of C' .

Figure 17: The circuit C' in (a) can be exactly represented by the 6×6 parity matrix \mathbf{M} in (b).

Quantum de Finetti Theorems as Categorical Limits, and Limits of State Spaces of C*-algebras

Sam Staton Ned Summers

Department of Computer Science,
University of Oxford
Oxford, United Kingdom

De Finetti theorems tell us that if we expect the likelihood of outcomes to be independent of their order, then these sequences of outcomes could be equivalently generated by drawing an experiment at random from a distribution, and repeating it over and over. In particular, the quantum de Finetti theorem says that exchangeable sequences of quantum states are always represented by distributions over a single state produced over and over. The main result of this paper is that this quantum de Finetti construction has a universal property as a categorical limit. This allows us to pass canonically between categorical treatments of finite dimensional quantum theory and the infinite dimensional. The treatment here is through understanding properties of (co)limits with respect to the contravariant functor which takes a C*-algebra describing a physical system to its convex, compact space of states, and through discussion of the Radon probability monad. We also show that the same categorical analysis also justifies a continuous de Finetti theorem for classical probability.

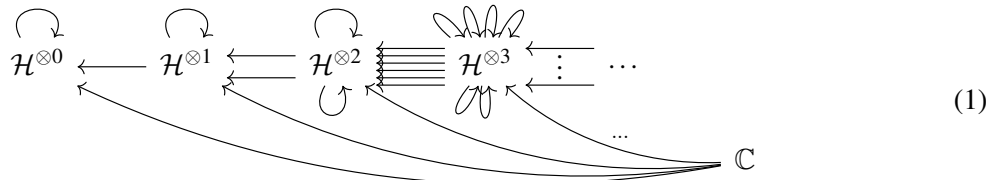
1 Introduction

The quantum analogue of de Finetti’s theorem [4, 14, 13, 24] explains that a “belief about a quantum state” has a more elementary description as an exchangeable sequences of quantum states. The point of this paper is to show that this de Finetti theorem can be phrased in categorical terms. Thus we connect this theorem, which is a fundamental theorem of quantum Bayesianism (e.g. [8]), with categorical and compositional approaches to axiomatizations and reconstructions of quantum theory (e.g. [6, 11, 15, 22, 23, 25, 26]).

Let \mathcal{H} be a Hilbert space (e.g. \mathbb{C}^2). An *sequence* of states on \mathcal{H} is a collection of quantum states on tensor powers of \mathcal{H} :

a state ρ_0 for $\mathcal{H}^{\otimes 0} = \mathbb{C}$, a state ρ_1 for $\mathcal{H}^{\otimes 1} = \mathcal{H}$, a state ρ_2 for $\mathcal{H}^{\otimes 2} = \mathcal{H} \otimes \mathcal{H}$, a state ρ_3 for $\mathcal{H}^{\otimes 3}$, ...

For example, if $\mathcal{H} = \mathbb{C}^2$, then a sequence of states on \mathcal{H} is a sequence of density matrices in $\mathbb{C}^{(2^n)^2}$. A sequence is *exchangeable* if each state commutes with reindexing, e.g. $\rho_2 = \rho_2 \circ \text{swap}$ for the swap map $\mathcal{H} \otimes \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{H}$, and if taking the partial trace of ρ_m over any $m - n$ indices gives ρ_n , for $n \leq m$. We can phrase this in categorical terms by recalling that a state is a quantum channel $\mathbb{C} \rightarrow \mathcal{H}$ (i.e. a CPTP map between the corresponding spaces of density matrices), and so an exchangeable sequence is a commuting cone of quantum channels



Our categorical statement of the quantum de Finetti theorem (Theorem 4.3, dgm. (5)) is about a limit for this diagram, i.e. a universal exchangeable sequence of quantum channels. This follows recent categorical treatments of the classical case [7, 16].

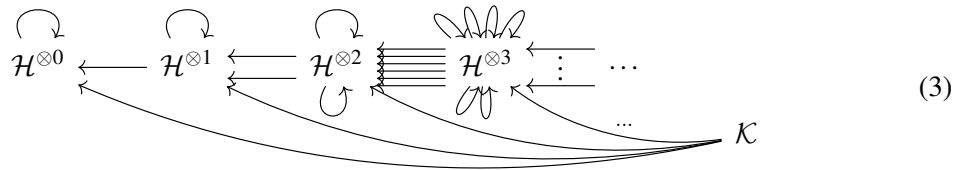
To give this categorical statement precisely, we make three steps.

1. Firstly, we extend our set up to allow for channels that have both quantum and classical information. Formally, this is done by recalling that the dual category $(\mathbf{C}_{\text{CPU}}^*)^{\text{op}}$ of \mathbf{C}^* -algebras and completely positive unital maps fully embeds the category of quantum channels, but also fully embeds a good deal of classical probability, in terms of Radon probability kernels between compact Hausdorff spaces.

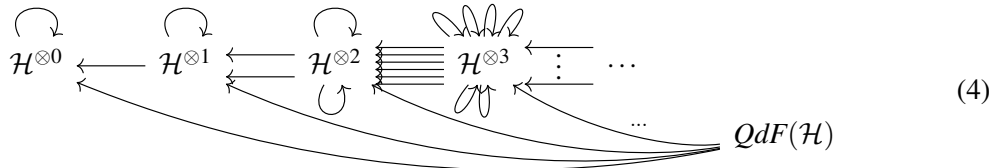
$$(\text{quantum channels}) \hookrightarrow (\mathbf{C}_{\text{CPU}}^*)^{\text{op}} \longleftarrow (\text{classical probability kernels}) \quad (2)$$

This move is important. It turns out that there is no Hilbert space that is the limit of diagram 1. Instead, some classical probability is necessary.

2. Then, rather than look only at exchangeable sequences of states $\mathbb{C} \rightarrow \mathcal{H}^{\otimes n}$, we look more generally at *parameterized* exchangeable sequences, i.e. sequences of channels $\mathcal{K} \rightarrow \mathcal{H}^{\otimes n}$, incorporating both classical and quantum randomness.

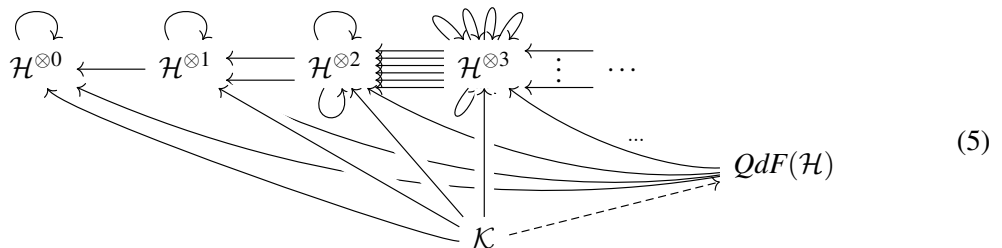


3. **Theorem 4.3** (paraphrased): There is a \mathbf{C}^* -algebra $QdF(\mathcal{H})$ and a cone



which is limiting in the category $(\mathbf{C}_{\text{CPU}}^*)^{\text{op}}$.

This universal property says that any cone factors uniquely through $QdF(\mathcal{H})$.



In other words, to give a cone, i.e. a parameterised exchangeable sequence of states in \mathcal{H} , is equivalent to giving a channel of quantum and classical information to $QdF(\mathcal{H})$.

The starting point for our proof of Theorem 4.3 is Størmer’s quantum de Finetti result [24]. Inspired by Størmer’s result, we take the candidate limiting cone $QdF(\mathcal{H})$ to be the \mathbf{C}^* -algebra corresponding to the space of classical distributions on *all* states of \mathcal{H} . Although Størmer’s work is not phrased in categorical terms at all, it follows from that result that there is a unique mediating morphism in diagram (5) in the case where $\mathcal{K} = \mathbb{C}$. To show that this is indeed is a categorical limit, we follow the following steps.

- We note that the candidate limiting cone $QdF(\mathcal{H})$ is classical, that is to say, it lies on the right hand, classical side of (2), even though the diagram itself typically lies on the left hand, quantum side of (2). So we can consider the categorical limit and diagram (5) in the category of \mathbf{C}^* -algebras and *positive* unital maps since positive maps into and out of commutative \mathbf{C}^* -algebras are necessarily *completely* positive.
- The category of \mathbf{C}^* -algebras and positive unital maps dually embeds into a category of compact spaces with convex structure, by regarding their states (§2, Thm. 2.18; [9]). This category of convex compact spaces is categorically well-behaved, since it is the category of algebras for a monad on another well-behaved category. In this larger category we are able to use standard monadicity results to translate ‘pointwise’ limiting structure to categorical limiting structure (§3, Theorem 3.2).

In particular, we can then show that diagram (4) is a categorical limit (§4, Theorem 4.3).

In this way, we can understand “belief about a quantum state” in categorical terms. This opens the door to using categorical diagrammatic notation, which we illustrate in Section 5.

2 Preliminaries

We recall rudiments of probability theory (§2.1) and \mathbf{C}^* -algebras (§2.3,2.6). In this context we recall classical and quantum de Finetti theorems (§2.2,2.7) and the \mathbf{C}^* -algebraic treatment of probability (§2.4,2.5).

2.1 Rudiments of Probability Theory

We begin by recalling some measure-theoretic probability theory.

Definition 2.1 (Probability Measure). For a set X , a σ -algebra for X is a collection of subsets of X , $\Sigma_X \subseteq \mathcal{P}(X)$, which contains X and is closed under countable unions and complements.

A *measurable space* is a pair (X, Σ_X) of a set X and a σ -algebra Σ_X on X .

In what follows, we are almost exclusively concerned with measures on topological spaces. The *Borel σ -algebra* $Borel(X)$ on a *topological space* X is the smallest σ -algebra generated by the open sets of X . Additionally, when we refer to finite or countable sets, we consider them as measurable spaces with the σ -algebra of all possible subsets.

A *measurable function* between measurable spaces (X, Σ_X) and (Y, Σ_Y) is a function $f: X \rightarrow Y$ such that for any $S \in \Sigma_Y$, we have $f^{-1}(S) \in \Sigma_X$. Continuous functions between topological spaces $(X, Borel(X)) \rightarrow (Y, Borel(Y))$ are measurable, though not all measurable functions are continuous.

A *probability measure on a measurable space* (X, Σ_X) is a function $\mu: \Sigma_X \rightarrow [0, 1]$ such that $\mu(X) = 1$ and for a disjoint countable collection of sets $\{U_i\}_{i \in \mathbb{N}} \subset \Sigma_X$, $\mu(\bigcup_{i \in \mathbb{N}} U_i) = \sum_{i \in \mathbb{N}} \mu(U_i)$. Given a probability measure $\mu: \Sigma_X \rightarrow [0, 1]$ and a measurable function $f: X \rightarrow Y$, the *pushforward of μ by f* , a probability measure on Y , is denoted $f_*\mu$ and given by $f_*\mu(S) := \mu(f^{-1}(S))$ for measurable $S \in \Sigma_Y$.

All our topological spaces will be compact, Hausdorff spaces, so we will only use measures that behave well with the compactness of the space. Let X be a compact Hausdorff space. A *Radon probability*

measure on X , $\mu: \text{Borel}(X) \rightarrow [0, 1]$, is a probability measure on the Borel σ -algebra which is *inner regular*: for any measurable $S \subseteq X$, $\mu(S) = \sup_{K \subseteq S} \mu(K)$ where K varies over all compact subsets. For any continuous function $f: X \rightarrow \mathbb{C}$, a Radon probability measure μ induces an integral $\int_{x \in X} f(x) d\mu \in \mathbb{C}$, so that we can regard μ as a map from the set (in fact, space) of continuous functions on X to \mathbb{C} (see Thm. 2.13, §2.5).

2.2 Kolmogorov Extension Theorem and Hewitt-Savage de Finetti Theorem

Kolmogorov’s extension theorem connects measures on infinite product spaces with measures on finite truncations. Recall that for a set of topological spaces $\{X_i\}_{i \in I}$, their product has underlying set $\prod_{i \in I} X_i$ and has topology generated by the cylinder sets $V[U_{i_1}, \dots, U_{i_n}] = \{(x_i)_{i \in I} \in \prod_{i \in I} X_i \mid x_{i_k} \in U_{i_k} \text{ for } 1 \leq k \leq n\}$, varying over all finite subsets $\{i_1, \dots, i_n\} \subset I$ and open sets $U_{i_k} \subset X_{i_k}$.

Theorem 2.2 (Kolmogorov Extension Theorem (e.g. [21])). *Let X be a compact Hausdorff space. Let $X^{\mathbb{N}}$ be the countable product of copies of X . For each finite $N \subset \mathbb{N}$, let μ_N be a Radon probability measure on X^N with the product topology, with the condition that if we take finite subsets $M \subset N \subset \mathbb{N}$, μ_M is the pushforward of μ_N by the projection $X^N \rightarrow X^M$. Then there exists a unique probability measure μ on $X^{\mathbb{N}}$ such that, for any finite $N \subset \mathbb{N}$, μ_N is the pushforward of μ by the projection $X^{\mathbb{N}} \rightarrow X^N$. Further, this measure is itself Radon.*

From here we can now define an exchangeable measure.

Definition 2.3 (Exchangeable Measure). Let X be a measurable space. Let μ be a measure on $X^{\mathbb{N}}$. For each permutation $\sigma: \mathbb{N} \rightarrow \mathbb{N}$, there is an isomorphism

$$\eta_\sigma: X^{\mathbb{N}} \rightarrow X^{\mathbb{N}} \quad \eta_\sigma(x_1, x_2, x_3, \dots) = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, x_{\sigma^{-1}(3)}, \dots).$$

μ is called *exchangeable* if, for every permutation $\sigma: \mathbb{N} \rightarrow \mathbb{N}$ which fixes all but a finite number of elements, we have that $(\eta_\sigma)_* \mu = \mu$.

Let X be a compact Hausdorff space. We define $\mathcal{R}(X)$ to be the set of all Radon probability measures on the Borel σ -algebra on X , made into a compact Hausdorff space with the topology generated by the open sets $\{\mu \in \mathcal{R}(X) : \int_{x \in X} f(x) d\mu \in U\}$ for $U \subseteq \mathbb{C}$ open, $f: X \rightarrow \mathbb{C}$ continuous.

Given a Radon measure μ on X , by Kolmogorov extension, there is a Radon measure $\tilde{\mu}$ on $X^{\mathbb{N}}$ defined on the basis of cylinder set opens by $\tilde{\mu}(V[U_{n_1}, \dots, U_{n_k}]) = \prod_{i \in \mathbb{N}} \mu(U_{n_i})$ for $U_{n_i} \subset X$ open. This is because for all $n \in \mathbb{N}$ there is a unique Radon measure on X^n , denoted by μ^n , which has $\mu^n(U_1 \times \dots \times U_n) = \prod_{i \in \mathbb{N}} \mu(U_i)$ for $U_i \in \text{Borel}(X)$.

Theorem 2.4 (Hewitt-Savage de Finetti Theorem [12]). *Let $\mu \in \mathcal{R}(X^{\mathbb{N}})$ be an exchangeable Radon probability measure on the countably infinite product of copies of X , with the Borel σ -algebra. Then there exists a Radon probability measure ν on $\mathcal{R}(X)$ (i.e. $\nu \in \mathcal{R}(\mathcal{R}(X))$) such that, for all measurable $U \subseteq X^{\mathbb{N}}$,*

$$\mu(U) = \int_{p \in \mathcal{R}(X)} \tilde{p}(U) d\nu.$$

Example 2.5. Supposing that we are modelling coin flips, so $X = \{H, T\}$, then $\mathcal{R}(X) \cong [0, 1]$, and the de Finetti result says that an exchangeable distribution on sequences of H and T can only come from picking from bag of coins with bias distributed according to some distribution on $[0, 1]$ and then flipping the coin you have picked over and over forever.

2.3 Rudiments of C^* -algebras and Gelfand duality

Definition 2.6 (C^* -Algebra). An algebra V (over \mathbb{C}) is a vector space V over \mathbb{C} equipped with a binary operation of *multiplication*, $\cdot : V \times V \rightarrow V$, which is bilinear. If this multiplication is commutative, then V is a *commutative algebra*. We will assume that all algebras are *unital*, i.e. have a multiplicative unit.

A *Banach algebra* is an algebra V equipped with a norm $\|\cdot\|$ such that V is complete with respect to $\|\cdot\|$ and for all $x, y \in V$, $\|x \cdot y\| \leq \|x\|\|y\|$. A *$*$ -algebra* is an algebra V that is equipped with an involution: a function $(-)^* : V \rightarrow V$ that is self-inverse, a multiplication antihomomorphism (i.e. it reverses multiplication) and is conjugate linear.

A bounded linear map between $*$ -algebras which preserves multiplication, the unit and involution is called a *$*$ -homomorphism*. A C^* -algebra \mathcal{A} is a Banach $*$ -algebra such that for all $x \in \mathcal{A}$, $\|x^*x\| = \|x\|^2$.

We write $\mathbf{C}_{\text{Mult}}^*$ for the category which has as its objects C^* -algebras and $*$ -homomorphisms as its morphisms. It has as a full subcategory $\mathbf{cC}_{\text{Mult}}^*$ of commutative C^* -algebras.

Example 2.7. For any Hilbert space \mathcal{H} over \mathbb{C} , we denote the space of all bounded linear operators $\phi : \mathcal{H} \rightarrow \mathcal{H}$ by $\mathcal{B}(\mathcal{H})$. $\mathcal{B}(\mathcal{H})$ is the prototypical example of a C^* -algebra. The generally non-commutative multiplication is given by composition of operators, the unit is the identity map and involution is taking the adjoint of a map. The norm is the operator norm.

Theorem 2.8 (e.g. [17], C.12). *Every C^* -algebra is isomorphic to a sub-algebra of $\mathcal{B}(\mathcal{H})$ for some \mathcal{H} .*

Example 2.9 (Commutative C^* -algebras). One important example of a C^* -algebra is the space $C(X) = \{\psi : X \rightarrow \mathbb{C} \mid \psi \text{ is continuous}\}$ for some compact Hausdorff space X , equipped with the (topological) supremum norm: $\|f\| = \sup_{x \in X} |f(x)| < \infty$. It is an algebra with multiplication and involution defined pointwise, and is commutative. The unit is the constant map to 1.

This extends to a duality between commutative C^* -algebras and the category \mathbf{CH} of compact Hausdorff spaces and continuous maps.

Theorem 2.10 (Gelfand Duality). *The functor $C(-) = \mathbf{Top}(-, \mathbb{C}) : \mathbf{CH} \rightarrow (\mathbf{cC}_{\text{Mult}}^*)^{\text{op}}$, which acts on morphisms by $C(f : X \rightarrow Y) : \phi \mapsto \phi \circ f$, is an equivalence of categories.*

2.4 Positivity, Probabilistic Gelfand Duality, and the Radon Monad

Definition 2.11 (Positivity in C^* -algebras). For a C^* -algebra \mathcal{A} , an element $x \in \mathcal{A}$ is called *positive* if there is some $y \in \mathcal{A}$ such that $x = y^*y$.

In \mathbb{C} , these are exactly the elements of the non-negative real line $\mathbb{R}_{0 \leq}$. In $\mathcal{B}(H)$, for some Hilbert space H , these are exactly the operators $\phi : H \rightarrow H$ such that for all $v \in H$, $\langle v \mid \phi v \rangle \geq 0$. In $C(X)$, for some compact Hausdorff space X , these are the functions whose images lie exclusively in $\mathbb{R}_{0 \leq}$.

A linear map between C^* -algebras, $f : \mathcal{A}_1 \rightarrow \mathcal{A}_2$, is called *positive* if, for all $x \in \mathcal{A}_1$, $f(x^*x) \geq 0$. In other words, it takes positive elements of \mathcal{A}_1 to positive elements of \mathcal{A}_2 . If both the domain and codomain have a unit, the map is called *unital* if it takes the unit of the domain to the unit of the codomain.

We will refer to the category of C^* -algebras with positive, unital maps between them as $\mathbf{C}_{\text{Pos}}^*$. It has $\mathbf{C}_{\text{Mult}}^*$ as a subcategory.

Jacobs and Furber [9] extended Gelfand duality stochastically: the addition of positive maps which are not multiplicative (i.e. not $*$ -homomorphisms) is equivalent to adding stochastic maps to \mathbf{CH} .

Definition 2.12 (Radon Monad, e.g. [9]). Let X be a compact Hausdorff space, and let $\mathcal{R}(X)$ be the space of all Radon probability measures on X as for Thm.2.4 above. With this topology, $\mathcal{R}(X)$ is both compact and Hausdorff.

We regard \mathcal{R} as a monad on the category \mathbf{CH} of compact Hausdorff spaces and continuous maps. The functor part is given by pushforward: let $f : X \rightarrow Y$ be a morphism in \mathbf{CH} ; then $\mathcal{R}(f)(\mu) := f_*(\mu)$. The unit of the monad takes $x \in X$ to the Dirac measure $\delta_x \in \mathcal{R}(X)$, the distribution supported entirely at x . The multiplication is a form of marginalization, or averaging:

$$\text{mult} : \mathcal{R}^2(X) \rightarrow \mathcal{R}(X) \quad \text{mult}(\phi)(U) := \int_{\mu \in \mathcal{R}(X)} \mu(U) d\phi(\mu)$$

Theorem 2.13 (Probabilistic Gelfand Duality, [9]). *The functor $C(-) : \mathcal{Kl}(\mathcal{R}) \rightarrow (\mathbf{cC}_{\text{Pos}}^*)^{\text{op}}$, which acts on morphisms by $C(f : X \rightarrow \mathcal{R}(Y)) : \phi \mapsto \int \phi df(-)$, is an equivalence of categories between the Kleisli category of the Radon monad and the opposite of the category of commutative \mathbf{C}^* -algebras and positive unital maps.*

2.5 States, State Spaces, and Convex Spaces

A particularly important class of positive, unital maps are states:

Definition 2.14. Positive, unital maps from a \mathbf{C}^* -algebra \mathcal{A} to \mathbb{C} are called *states on \mathcal{A}* .

Under the probabilistic Gelfand duality (Thm. 2.13), for any compact Hausdorff space X we have the correspondence (as sets)

$$\mathbf{C}_{\text{Pos}}^*(C(X), \mathbb{C}) \cong \mathcal{Kl}(\mathcal{R})(\{*\}, X) = \mathbf{CH}(\{*\}, \mathcal{R}(X)) \cong \mathcal{R}(X).$$

In fact, all of the objects in this correspondence have structures as convex, compact, Hausdorff spaces and are isomorphic as such. Thus it is meaningful to consider states on \mathbf{C}^* -algebras as a generalisation of classical probability distributions.

Example 2.15 (Density Matrices are States on a \mathbf{C}^* -algebra). Density matrices in quantum theory are given by operators $\rho \in \mathcal{B}(\mathcal{H})$, for a finite dimensional Hilbert space \mathcal{H} , with $\text{Tr}(\rho) = 1$ such that for all $v \in \mathcal{H}$, $\langle v | \rho v \rangle \geq 0$. For each such ρ , we may define a linear map $s_\rho : \mathcal{B}(\mathcal{H}) \rightarrow \mathbb{C}$ by $s_\rho(a) = \text{Tr}(\rho a)$. The trace condition says this map is unital. Further, in the finite dimensional case, where we can form an eigenvalue decomposition $\rho = \sum_{i \in I} p_i |v_i\rangle\langle v_i|$, then

$$s_\rho(a^*a) = \text{Tr}(\rho a^*a) = \sum_i p_i \langle v_i | a^* a v_i \rangle = \sum_i p_i \langle a v_i | a v_i \rangle = \sum_i p_i \|a v_i\|^2 \geq 0.$$

So s_ρ is a state on $\mathcal{B}(\mathcal{H})$. In fact, in this finite dimensional case, all states on $\mathcal{B}(\mathcal{H})$ are of this form.

So we can see \mathbf{C}^* -algebra states generalise both classical and quantum probability.

Definition 2.16 (State Space of a \mathbf{C}^* -algebra). Let \mathcal{A} be a \mathbf{C}^* -algebra. *The state space of \mathcal{A}* , denoted by $S(\mathcal{A}) := \mathbf{C}_{\text{Pos}}^*(\mathcal{A}, \mathbb{C})$, is the set of all states on \mathcal{A} equipped with the coarsest topology such that for all $a \in \mathcal{A}$, the evaluation function $\text{ev}_a : S(\mathcal{A}) \rightarrow \mathbb{C}$ which takes $\rho \mapsto \rho(a)$ is continuous. The topology is generated by the sets $\text{ev}_a^{-1}(\Omega)$ for $a \in \mathcal{A}$ and $\Omega \subset \mathbb{C}$ open. $S(\mathcal{A})$ has the additional properties of always being Hausdorff and compact. Moreover, $S(-)$ extends to a functor $(\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \rightarrow \mathbf{CH}$ via $S(f) = - \circ f$.

In fact $S(\mathcal{A})$ also has an obvious convex structure where the convex combination $\lambda \rho_1 + (1 - \lambda) \rho_2$ for $\lambda \in [0, 1]$ and $\rho_1, \rho_2 \in S(\mathcal{A})$ is evaluated pointwise using the addition of \mathbb{C} . We now recall how this convex structure is functorial.

Definition 2.17. For our purposes, a *compact convex space* is a pair (V, X) where V is a convex, compact subset of a locally convex, Hausdorff topological vector space X . The category \mathbf{ConvCH} has as objects compact convex spaces (V, X) , and morphisms are affine, continuous maps between the convex subsets V (only).

Theorem 2.18 ([9]). *The state space functor $S: (\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \rightarrow \mathbf{CH}$ factors through \mathbf{ConvCH} . Moreover the resulting functor $S: (\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \rightarrow \mathbf{ConvCH}$ is full and faithful.*

In other words, $(\mathbf{C}_{\text{Pos}}^*)^{\text{op}}$ is isomorphic to a full subcategory of \mathbf{ConvCH} (characterized in [1]).

Theorem 2.19 ([9]). *The category of Eilenberg-Moore algebras of the Radon monad, $\mathcal{E}m(\mathcal{R})$ (Def. 2.12), is equivalent to the category \mathbf{ConvCH} (Def. 2.17).*

This realises the forgetful functor $U: \mathbf{ConvCH} \rightarrow \mathbf{CH}$ as the right adjoint of the monadic adjunction $\mathcal{R}' \dashv U$ for $\mathcal{R}': \mathbf{CH} \rightarrow \mathbf{ConvCH}$ taking a space X to the convex, compact Hausdorff space $\mathcal{R}(X)$.

Note then that all the objects of $\mathcal{E}m(\mathcal{R})$ are either spaces of probability measures (the free algebras, equivalently objects of $\mathcal{K}l(\mathcal{R})$, Thm. 2.13), or they are (category-theoretical) “quotients” of these spaces (the non-free algebras). The transition from classical to quantum probability has as a crucial part the fact that some convex mixtures of outcomes to an experiment are equivalent (for example, different decompositions of mixed states into pure states).

2.6 Tensor products and complete positivity

Recall that the tensor product of Hilbert spaces, $\mathcal{H}_1 \otimes_H \mathcal{H}_2$, is the completion of the algebraic tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ under the inner product norm.

Definition 2.20 (Spatial Tensor Product of \mathbf{C}^* -algebras). Let \mathcal{A}_1 and \mathcal{A}_2 be \mathbf{C}^* -algebras, with representations $\pi_i: \mathcal{A}_i \rightarrow \mathcal{B}(\mathcal{H}_i)$ (Thm. 2.8). Then the map $\pi_1 \otimes \pi_2: \mathcal{A}_1 \otimes \mathcal{A}_2 \rightarrow \mathcal{B}(\mathcal{H}_1 \otimes_H \mathcal{H}_2)$ given by $(\pi_1 \otimes \pi_2)(A_1 \otimes A_2) = \pi_1(A_1) \otimes \pi_2(A_2)$ is a faithful representation of the vector space $\mathcal{A}_1 \otimes \mathcal{A}_2$ and thus we can use it to give a norm to that space, $\|a\|_* = \|(\pi_1 \otimes \pi_2)(a)\|$, which is independent of the choice of representations (π_i) . The spatial tensor product $\mathcal{A}_1 \otimes_{\min} \mathcal{A}_2$ is the completion with respect to this norm.

There are other tensor products definable on \mathbf{C}^* -algebras, though this is the smallest. If a \mathbf{C}^* -algebra \mathcal{A} is finite dimensional or commutative, then all possible \mathbf{C}^* -norms on $\mathcal{A} \otimes \mathcal{B}$ are equivalent.

Not all positive maps are physical, and now that we have defined a tensor product on \mathbf{C}^* -algebras, we are able to define those that are.

Definition 2.21 (Completely Positive). A linear map between \mathbf{C}^* -algebras $\phi: \mathcal{A} \rightarrow \mathcal{B}$ is called *completely positive* if, for all $n \in \mathbb{N}$, the map $\phi \otimes 1_n: \mathcal{A} \otimes \mathcal{B}(\mathbb{C}^n) \rightarrow \mathcal{B} \otimes \mathcal{B}(\mathbb{C}^n)$ is positive.

All positive maps to or from commutative spaces are completely positive.

Example 2.22. Let \mathcal{H}, \mathcal{K} be finite dimensional Hilbert spaces. Then a completely positive and unital map $\mathcal{B}(\mathcal{H}) \rightarrow \mathcal{B}(\mathcal{K})$ induces a function $S(\mathcal{B}(\mathcal{K})) \rightarrow S(\mathcal{B}(\mathcal{H}))$ between the corresponding spaces of density matrices (Ex. 2.15); these are exactly the quantum channels (e.g. [20]).

As we noticed a few times before, much of the seemingly infinite behaviour in different formulations of the De Finetti theorem is, in fact, the result of behaviour which happens on all possible finite truncations of a process, with requirements of consistency between them.

Given a \mathbf{C}^* -algebra \mathcal{A} , we can define $\mathcal{A}^{\otimes n} := \underbrace{\mathcal{A} \otimes_{\min} \dots \otimes_{\min} \mathcal{A}}_{n \text{ times}}$. For $n \leq m$, there is an isometric *-homomorphism embedding of \mathbf{C}^* -algebras

$$\iota_{nm}: \mathcal{A}^{\otimes n} \rightarrow \mathcal{A}^{\otimes m} \quad \iota_{nm} \left(\bigotimes_{i=1}^n A_i \right) := \bigotimes_{i=1}^n A_i \otimes \bigotimes_{i=n+1}^m 1_{\mathcal{A}}.$$

Definition 2.23 (Infinite Spatial Tensor Product). The (countably) infinite spatial tensor product of \mathcal{A} is defined as the colimit as Banach spaces (equivalently, in $\mathbf{C}_{\text{Pos}}^*$) of the ω -shaped diagram, for $\omega =$

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow \dots$, which has the objects $\mathcal{A}^{\otimes n}$ for all $n \in \mathbb{N}$ and the morphisms ι_{nm} for all $n \leq m$. We will denote it by $\mathcal{A}^{\otimes \infty}$.

This comes equipped with embeddings, again isometric $*$ -homomorphisms, $\psi_n : \mathcal{A}^{\otimes n} \rightarrow \mathcal{A}^{\otimes \infty}$ which we may intuitively imagine as taking $\bigotimes_{i=1}^n A_i \mapsto \bigotimes_{i=1}^n A_i \otimes \bigotimes_{i=n+1}^{\infty} 1_{\mathcal{A}}$ since for all $n \leq m$, $\psi_n = \psi_m \circ \iota_{nm}$.

$$\begin{array}{ccccccc}
 \mathcal{A}^{\otimes 1} & \xrightarrow{\iota_{12}} & \mathcal{A}^{\otimes 2} & \xrightarrow{\dots \iota_{2n}} & \mathcal{A}^{\otimes n} & \xrightarrow{\dots \iota_{nm}} & \mathcal{A}^{\otimes m} & \dots \\
 & \searrow \psi_1 & & \searrow \psi_2 & \downarrow \psi_n & & \swarrow \psi_m & \\
 & & & & \mathcal{A}^{\otimes \infty} & & &
 \end{array} \tag{6}$$

2.7 The Quantum Kolmogorov Extension and de Finetti Theorems

Theorem 2.24 (Quantum Kolmogorov Extension Theorem [10]). *Let $\{\rho_n\}_{n \in \mathbb{N}}$ be a sequence with $\rho_n \in S(\mathcal{A}^{\otimes n})$ such that, for all $n \leq m$, $\rho_n = \rho_m \circ \iota_{nm}$. There is a unique state $\rho \in S(\mathcal{A}^{\otimes \infty})$ such that $\rho_n = \rho \circ \psi_n$ for all $n \in \mathbb{N}$ (with ψ_n as in (6)).*

Definition 2.25 (Exchangeable State). Given a permutation $\sigma \in \mathcal{S}_n$, we may define an associated map permuting the spaces of the n-fold tensor product $\mathcal{A}^{\otimes n}$

$$\eta_{\sigma} : \mathcal{A}^{\otimes n} \rightarrow \mathcal{A}^{\otimes n} \quad \eta_{\sigma}(a_1 \otimes \dots \otimes a_n) := a_{\sigma^{-1}(1)} \otimes \dots \otimes a_{\sigma^{-1}(n)}$$

A state $\rho_n \in S(\mathcal{A}^{\otimes n})$ is said to be *symmetric* if, for all $\sigma \in \mathcal{S}_n$, $\rho_n = \rho_n \circ \eta_{\sigma}$. A state $\rho : \mathcal{A}^{\otimes \infty} \rightarrow \mathbb{C}$ is said to be *exchangeable* if, for all $n \in \mathbb{N}$, $\rho_n = \rho \circ \psi_n$ is symmetric. Note again that we are only concerned with permutations on a finite number of factors.

The space of exchangeable states of $\mathcal{A}^{\otimes \infty}$, denoted by $\mathcal{I}(\mathcal{A}) := \{\rho \in S(\mathcal{A}^{\otimes \infty}) \mid \rho \text{ is exchangeable}\}$ is convex, compact and Hausdorff, so in **ConvCH**.

Theorem 2.26 (Størmer’s Quantum de Finetti Theorem). *Let \mathcal{A} be a \mathbf{C}^* -algebra. Then there is a bi-continuous, affine bijection $\mathcal{I}(\mathcal{A}) \cong \mathcal{R}(S(\mathcal{A}))$. In other words, the exchangeable state space of $\mathcal{A}^{\otimes \infty}$ is isomorphic to $S(C(S(\mathcal{A}))) \cong \mathcal{R}(S(\mathcal{A}))$ in **ConvCH**.*

Remark. To be explicit about this isomorphism we note that given states $\rho_1 \in S(\mathcal{A}_1), \rho_2 \in S(\mathcal{A}_2)$, we can form a unique state $\rho_1 \otimes \rho_2 \in S(\mathcal{A}_1 \otimes_{\min} \mathcal{A}_2)$ with the property that $\rho_1 \otimes \rho_2(a_1 \otimes a_2) = \rho_1(a_1)\rho_2(a_2)$. Then, for $\rho \in S(\mathcal{A})$, we define

$$\rho^{\otimes n} := \underbrace{\rho \otimes \dots \otimes \rho}_{n \text{ times}} \in S(\mathcal{A}^{\otimes n}).$$

There is a state on $\mathcal{A}^{\otimes \infty}$, $\rho^{\otimes \infty} \in S(\mathcal{A}^{\otimes \infty})$, via theorem 2.24 from the sequence $\{\rho^{\otimes n}\}_{n \in \mathbb{N}}$.

This isomorphism then is given by $-\circ \Phi : S(C(S(\mathcal{A}))) \rightarrow \mathcal{I}(\mathcal{A})$ where $\Phi : \mathcal{A}^{\otimes \infty} \rightarrow C(S(\mathcal{A}))$ is defined as $\Phi(a)(\rho) = \rho^{\otimes \infty}(a)$.

3 (Co)limits and the State Space Functor

We now build on the prior work in Section 2 to characterize the categorical limits in the categories of positive unital maps between \mathbf{C}^* -algebras ($(\mathbf{C}_{\text{Pos}}^*)^{\text{op}}$) and compact convex spaces (**ConvCH**).

Lemma 3.1. *The state space functor $S : (\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \rightarrow \mathbf{ConvCH}$ (Thm. 2.18) preserves and reflects limits.*

That is to say that, given a diagram of \mathbf{C}^* -algebras $\mathcal{A} : \mathcal{J} \rightarrow \mathbf{C}_{\text{Pos}}^*$, a cocone $\{\mathcal{A}_j \rightarrow \mathcal{B}\}_{j \in \mathcal{J}}$ is a colimit if and only if the corresponding cone $\{S(\mathcal{B}) \rightarrow S(\mathcal{A}_j)\}_{j \in \mathcal{J}}$ of the diagram $(\mathcal{J})^{\text{op}} \rightarrow (\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \xrightarrow{S} \mathcal{Em}(\mathcal{R})$ is a limit.

Proof. Since S is full and faithful (Thm. 2.18), it must reflect limits and colimits.

There are monadic forgetful functors from $U: \mathcal{E}m(\mathcal{R}) \rightarrow \mathbf{CH}$ and from $U': \mathbf{CH} \rightarrow \mathbf{Set}$ (Thm. 2.19, [18, VI.9]). They create limits. The composition

$$(\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \xrightarrow{S} \mathcal{E}m(\mathcal{R}) \xrightarrow{U} \mathbf{CH} \xrightarrow{U'} \mathbf{Set}$$

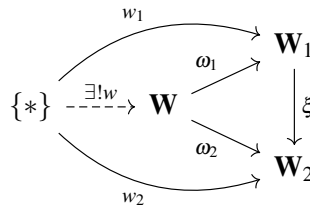
is just the hom-functor $\mathbf{C}_{\text{Pos}}^*(-, \mathbb{C}): (\mathbf{C}_{\text{Pos}}^*)^{\text{op}} \rightarrow \mathbf{Set}$. Thus, supposing a colimit exists in $\mathbf{C}_{\text{Pos}}^*$, it is preserved via $\mathbf{C}_{\text{Pos}}^*(-, \mathbb{C})$ into a limit in \mathbf{Set} , which then creates a limit in \mathbf{CH} and then creates another in $\mathcal{E}m(\mathcal{R}) \cong \mathbf{ConvCH}$. Thus S preserves the original colimit/limit in $(\mathbf{C}_{\text{Pos}}^*)^{\text{op}}$. \square

The de Finetti limits we construct will be a simple result of the way that limits in \mathbf{ConvCH} are constructed *pointwise* in the following way:

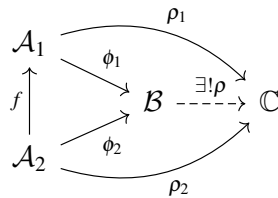
Lemma 3.2 (Pointwise limits are limits in \mathbf{ConvCH}). *Consider a diagram $\mathbf{W}_-: \mathcal{J} \rightarrow \mathbf{ConvCH}$. That is, consider a collection $\{\mathbf{W}_i\}_{i \in \mathcal{J}}$ of compact convex spaces with affine continuous maps $\{\xi_k: \mathbf{W}_i \rightarrow \mathbf{W}_j\}_{k \in \mathcal{J}(i,j)}$. Let \mathbf{W} be a convex, compact Hausdorff space with morphisms $\omega_i: \mathbf{W} \rightarrow \mathbf{W}_i$ satisfying the following properties:*

- $\{\omega_i: \mathbf{W} \rightarrow \mathbf{W}_i\}_{i \in \mathcal{J}}$ is a cone over the diagram: for all $\xi_k: \mathbf{W}_i \rightarrow \mathbf{W}_j$ in the diagram, $\omega_i = \omega_j \circ \xi_k$.
- For any collection of elements $(w_i)_{i \in \mathcal{J}} \in \prod_{i \in \mathcal{J}} \mathbf{W}_i$ which are compatible, in the sense that for all $\xi_k: \mathbf{W}_j \rightarrow \mathbf{W}_i$ in the diagram, $w_i = \xi_k(w_k)$, there is a unique element $w \in \mathbf{W}$ with $w_i = \omega_i(w)$.

Then \mathbf{W} is the limit of the diagram in \mathbf{CH} and \mathbf{ConvCH} .



If the cone comprises state spaces $\mathbf{W}_j = S(\mathcal{A}_j)$ and $\mathbf{W} = S(\mathcal{B})$, rather than general objects of \mathbf{ConvCH} , then the second condition can alternatively be visualised on the level of positive unital maps:



In this way we may talk about limits of state spaces being built *pointwise*. If every set of compatible states corresponds to a limiting state on some other space, then this is naturally a limit, without having to concern ourselves with continuity or affineness of the limiting maps.

Proof. The forgetful functors $U: \mathbf{ConvCH} \rightarrow \mathbf{CH}$ and $U': \mathbf{CH} \rightarrow \mathbf{Set}$ are both monadic (Thm. 2.19, [18, VI.9]) and thus create limits.

Suppose we have a diagram $\mathbf{W}_-: \mathcal{J} \rightarrow \mathbf{ConvCH}$ as above, and a pointwise limit \mathbf{W} with maps $\omega_i: \mathbf{W} \rightarrow \mathbf{W}_i$, where W_i is the underlying set for each \mathbf{W}_i . Then for any set A , a cone of the diagram

of W_i s, that is a collection of maps $\{f_i: A \rightarrow W_i\}_{i \in I}$ gives a unique map $f: A \rightarrow W$ by, for each $a \in A$, letting $f(a)$ be the element of W corresponding to the collection $\{f_i(a) \in W_i\}$ under the pointwise limit property. Thus, W is the limit in **Set** of the diagram

$$\mathcal{J} \xrightarrow{W_-} \mathbf{ConvCH} \xrightarrow{U} \mathbf{CH} \xrightarrow{U'} \mathbf{Set}$$

Since **CH** is monadic over **Set**, then **W**, regarded as a compact Hausdorff space, is created as the limit in **CH** and similarly by monadicity of **ConvCH** over **CH**, it is created as the limit in **ConvCH**. \square

Informally, what this lemma tells us is that non-categorical, state-based treatments of limiting behaviours have been categorical all along. Looking at these things pointwise is fine because the structural properties take care of themselves.

4 A Quantum de Finetti Theorem as a Categorical Limit

We now use the lemmas of Section 3 to recast the quantum Kolmogorov extension theorem (Thm. 2.24) and quantum de Finetti theorem (Thm. 2.26) in a categorical light (Thms. 4.1, 4.3).

Theorem 4.1 (Quantum Kolmogorov Extension Theorem as a Categorical Limit). *Let \mathcal{A} be a \mathbf{C}^* -algebra. The limit of the diagram of state spaces (as compact convex spaces in **ConvCH**)*

$$S(\mathcal{A}) \xleftarrow{-\circ l_{12}} S(\mathcal{A}^{\otimes 2}) \xleftarrow{-\circ l_{23}} S(\mathcal{A}^{\otimes 3}) \xleftarrow{\dots} \dots \tag{7}$$

is $S(\mathcal{A}^{\otimes \infty})$ with the inclusions $- \circ \psi_n: S(\mathcal{A}^{\otimes \infty}) \rightarrow S(\mathcal{A}^{\otimes n})$.

Proof. Since this diagram is exactly the image under S of that which has $\mathcal{A}^{\otimes \infty}$ as its colimit (Def. 2.23), and by Lma. 3.1, S preserves colimits. \square

Lemma 4.2 (Quantum de Finetti Theorem as a Categorical Limit For Positive Maps). *Let \mathcal{A} be a \mathbf{C}^* -algebra. Let \mathbf{I}_{inj} be the category of the finite sets $\{1, \dots, n\}$ for $n \in \mathbb{N}$ and injections between them. We define an \mathbf{I}_{inj} -indexed diagram in $\mathbf{C}_{\text{Pos}}^*$ by taking $\{1, \dots, n\}$ to $\mathcal{A}^{\otimes n}$ and, for $n \leq m$ and an injection $\tau: \{1, \dots, n\} \hookrightarrow \{1, \dots, m\}$, we define*

$$\eta_\tau: \mathcal{A}^{\otimes n} \rightarrow \mathcal{A}^{\otimes m} \tag{8}$$

by taking $A_1 \otimes \dots \otimes A_n$ to the element $B_1 \otimes \dots \otimes B_m$ which has

$$B_j = \begin{cases} A_i & \text{if } j = \tau(i), \\ 1 & \text{otherwise.} \end{cases}$$

The colimit of this diagram in $\mathbf{C}_{\text{Pos}}^*$ is $C(S(\mathcal{A}))$.

In other words, the space of exchangeable sequences of states is a limit of a diagram and is affinely isomorphic to the space of Radon probability measures on $S(\mathcal{A})$.

Proof. Størmer’s proof of Thm. 2.26 [24] gives a bi-continuous, affine bijection $\mathcal{I}(\mathcal{A}) \cong \mathcal{R}(S(\mathcal{A}))$. In other words, the symmetric state space of $\mathcal{A}^{\otimes \infty}$ is isomorphic to $\mathcal{R}(S(\mathcal{A}))$ in **ConvCH**. All that is necessary then is to show that $\mathcal{I}(\mathcal{A}) \subset S(\mathcal{A}^{\otimes \infty})$, with the morphisms $\rho \mapsto \rho \circ \psi_n$ (with ψ_n as in (6)) is the limit of the diagram above. We do this using Lma.3.2.

Via Thm. 2.18, we can regard the diagram $\{\eta_\tau: \mathcal{A}^{\otimes n} \rightarrow \mathcal{A}^{\otimes m}\}$ of C^* -algebras as a diagram of state spaces $\{S(\eta_\tau): S(\mathcal{A}^{\otimes m}) \rightarrow S(\mathcal{A}^{\otimes n})\}$ in **ConvCH**. Since any $\rho \in \mathcal{I}(\mathcal{A})$ is symmetric we get that $\rho \mapsto \rho \circ \psi_n$ is a cone for $\{S(\eta_\tau)\}$.

Any collection of states $\{\rho_n \in S(\mathcal{A}^{\otimes n})\}_{n \in \mathbb{N}}$ which satisfies the diagram $\{S(\eta_\tau)\}$ also satisfies the diagram 7 and thus we get a state on $\mathcal{A}^{\otimes \infty}$, $\rho \in S(\mathcal{A}^{\otimes \infty})$, such that $\rho_n = \rho \circ \psi_n$ for all $n \in \mathbb{N}$. Since for any permutation σ of $\{1, \dots, n\}$, $\rho_n \circ \eta_\sigma = \rho_n$, ρ is symmetric. Thus, $\rho \in \mathcal{I}(\mathcal{A})$ and $\mathcal{I}(\mathcal{A})$ is a pointwise limit, and thus a limit for the diagram $\{S(\eta_\tau)\}$ in **ConvCH**, via Lma. 3.2.

That $C(S(\mathcal{A}))$ is the colimit in the diagram in $\mathbf{C}_{\text{Pos}}^*$ is a corollary of Lma.3.1: S reflects limits. \square

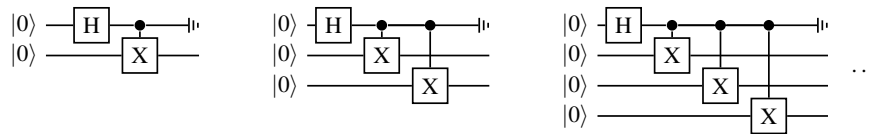
Theorem 4.3 (Quantum de Finetti Theorem as a Categorical Limit For Quantum Channels). *The colimit of the diagram (8) in $\mathbf{C}_{\text{CPU}}^*$ is $C(S(\mathcal{A}))$.*

Proof. Since all the maps in the diagram (8) are $*$ -homomorphisms, in particular completely positive, the diagram factors through $\mathbf{C}_{\text{CPU}}^*$ and, because $C(S(\mathcal{A}))$ is commutative, the colimiting maps are always completely positive. Thus from Lma. 4.2, $C(S(\mathcal{A}))$ is also the colimit in $\mathbf{C}_{\text{CPU}}^*$. \square

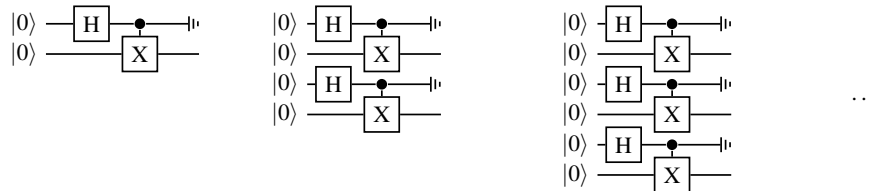
5 Illustrations

To illustrate the de Finetti construction, we focus temporarily on states of a qubit (i.e. $\mathcal{A} = \mathcal{B}(\mathbb{C}^2)$). Our categorical quantum de Finetti theorem gives a universal property to the infinite dimensional space of all states ($C(S(\mathcal{A}))$), in terms of finite dimensional spaces ($\mathbb{C}^2, \mathbb{C}^4$ etc..). We can thus use conventional methods from finite dimensional categorical/diagrammatic quantum mechanics to analyze $C(S(\mathcal{A}))$.

Since every state in $\mathcal{B}(\mathbb{C}^{2^n})$ corresponds to a quantum circuit with n output qubits, we can describe a sequence of states by giving a sequence of circuits. For example, the following sequence of circuits describes a sequence of qubit states that is exchangeable (in that postcomposition with any permutation or discarding of the qubit wires respects the sequence). For familiarity, we use a quantum circuit notation, but any diagrammatic notation with discarding could be used (e.g. [3]).



This exchangeable sequence corresponds to a belief about a qubit state: that the state is pure, and is either $|0\rangle$ (with probability 0.5) or $|1\rangle$ (with probability 0.5). The following sequence is also exchangeable:

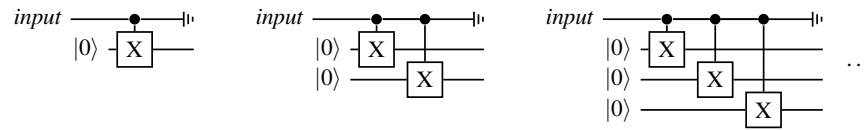


But this sequence corresponds to a different belief about a qubit state: that the state is definitely the mixed state with density matrix $\frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. In each case, the overall sequence of states is equivalent to the process of drawing a state at random from a classical distribution and then repeating it.

There are many other exchangeable sequences of qubit states. For example, there is one corresponding to the belief that a qubit is in a pure state somewhere on the equator of the Bloch sphere, $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi} |1\rangle)$, with ϕ uniformly distributed in $[0, 2\pi]$. There is also one corresponding to the belief that a qubit is in a totally unknown state, i.e. density matrix $\frac{r^{\frac{1}{2}}}{2} \begin{pmatrix} 1+z & \sqrt{1-z^2}e^{-i\theta} \\ \sqrt{1-z^2}e^{i\theta} & 1-z \end{pmatrix}$ with r, z, θ uniformly distributed in $[0, 1], [-1, 1]$ and $[0, 2\pi]$ respectively.

Our categorical version of the quantum de Finetti theorem allows us to also consider sequences with parameters. Since every completely positive unital map $\mathcal{B}(\mathbb{C}^{2^n}) \rightarrow \mathcal{B}(\mathbb{C}^{2^m})$ corresponds to a quantum circuit with n output qubits and m input qubits, we can describe a cone with apex $\mathcal{B}(\mathbb{C}^{2^m})$ by giving a sequence of circuits with m inputs.

For example, the following sequence of circuits describes a cone with apex $\mathcal{B}(\mathbb{C}^2)$.



This corresponds to a belief that the quantum state is pure, and either $|0\rangle$ or $|1\rangle$, with the probability determined by a standard basis measurement of the input state. Indeed, the categorical quantum de Finetti theorem (Thm. 4.3) says that every exchangeable sequence of circuits is equivalent to a sequence where each circuit first measures all the input qubits, resulting in random classical data, and then generates a quantum state depending on this classical outcome.

6 Aside on the Hewitt-Savage de Finetti Theorem as a Categorical Limit

As an aside, we note that de Finetti theorem for classical probability (Thm. 2.4) now also arises as a categorical limit. We express it first in the category **ConvCH** of compact convex spaces, but then state it in terms of the Kleisli category of the Radon monad to show the similarity with the previous result in [16].

Theorem 6.1 (Categorical Hewitt-Savage De Finetti Theorem). *Let X be a compact Hausdorff space. Consider the diagram $(\mathbf{I}_{\text{inj}})^{\text{op}} \rightarrow \mathbf{ConvCH}$ which takes $\{1, \dots, n\}$ to $\mathcal{R}(X^n)$ and an injective function $\tau: \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ to $\zeta_\tau: \mathcal{R}(X^m) \rightarrow \mathcal{R}(X^n)$, defined by $\zeta_\tau(\mu)(A) = \mu(\tilde{A}_\tau)$ for*

$$\tilde{A}_\tau := \{(x_1, \dots, x_m) \in X^m \mid (x_{\tau(1)}, x_{\tau(2)}, \dots, x_{\tau(n)}) \in A\}.$$

The limit of this diagram is the space of exchangeable measures on $X^{\mathbb{N}}$, and is isomorphic to $\mathcal{R}(\mathcal{R}(X))$, where the maps $\mathcal{R}(\mathcal{R}(X)) \rightarrow \mathcal{R}(X^n)$ take a measure Φ on $\mathcal{R}(X)$ to the measure

$$A \in \text{Borel}(X^n) \mapsto \int_{\mu \in \mathcal{R}(X)} \underbrace{\mu \times \dots \times \mu(A)}_{n \text{ times}} d\Phi.$$

Proof. This follows from instantiating Theorem 4.3 with the commutative \mathbf{C}^* -algebra $C(X)$ and noting that $C(X) \otimes_{\min} C(Y) \cong C(X \times Y)$. □

To emphasise the connection with [16], we write $X \rightsquigarrow Y$ for a Kleisli morphism $X \rightarrow \mathcal{R}(Y)$.

Corollary 6.2 (Categorical de Finetti Theorem in $\mathcal{Kl}(\mathcal{R})$). *For some $X \in \mathbf{CH}$, consider the diagram $(\mathbf{I}_{\text{inj}})^{\text{op}} \rightarrow \mathcal{Kl}(\mathcal{R})$ into the Kleisli category of the Radon monad, which takes $\{1, \dots, n\}$ to X^n and each injection $\tau: \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ to the Kleisli-map $\eta_\tau: X^m \rightsquigarrow X^n$ given by*

$$\eta_\tau(x_1, \dots, x_m) = \delta_{(x_{\tau(1)}, x_{\tau(2)}, \dots, x_{\tau(n)})}$$

This diagram has limit $\mathcal{R}(X)$ in $\mathcal{Kl}(\mathcal{R})$, with maps $\text{iid}_n: \mathcal{R}(X) \rightsquigarrow X^n$ given, for measurable $A \subset X^n$, by

$$\text{iid}_n(\mu)(A) = \underbrace{(\mu \times \dots \times \mu)}_{n \text{ times}}(A).$$

Proof. The limit in theorem 6.1 is reflected into $\mathcal{Kl}(\mathcal{R})$, as this category is a full subcategory of $\mathcal{E}m(\mathcal{R}) \cong \mathbf{ConvCH}$ and the limit itself is a free algebra. \square

The notation iid_n arises because $\underbrace{\mu \times \dots \times \mu}_{n \text{ times}}$ describes independent and identical distributions μ on n copies of X .

7 Concluding remarks

We have shown that the quantum de Finetti theorem amounts to a categorical limit for a diagram $\mathcal{A}^{\otimes -}: (\mathbf{I}_{\text{inj}})^{\text{op}} \rightarrow (\mathbf{C}_{\text{CPU}}^*)^{\text{op}}$ (Theorem 4.3). This puts the quantum de Finetti theorem, a cornerstone of quantum Bayesianism and a starting point for quantum tomography, in the setting of categorical and diagrammatic quantum theory. We have focused on \mathbf{C}^* -algebras, but the set-up is relevant more broadly. Recall that an *affine monoidal category* is a symmetric monoidal category with a terminal unit, and these are argued to be causal models of quantum theory (e.g. [5]). Now, $(\mathbf{I}_{\text{inj}})^{\text{op}}$ is the *free* affine monoidal category on one generator (e.g. [19]), and so $\mathcal{A}^{\otimes -}: (\mathbf{I}_{\text{inj}})^{\text{op}} \rightarrow (\mathbf{C}_{\text{CPU}}^*)^{\text{op}}$ is canonical for \mathcal{A} . So we can consider de Finetti limits in other models of quantum theory, perhaps making a bridge to the test space analysis of [2].

Acknowledgements. One starting point for this work was a discussion at the 2020 Workshop on Categorical Probability and Statistics, and we thank the organizers and participants at that workshop, in particular Robert Furber, Bart Jacobs, Arthur Parzygnat and Robert Spekkens. Thanks too to the Oxford group and anonymous reviewers.

Research supported by AFOSR award number FA9550-21-1-0038; the ERC BLAST grant; and a Royal Society University Research Fellowship.

References

- [1] Erik M. Alfsen & Frederic W. Shultz (2001): *State Spaces of Operator Algebras*, 1 edition. Birkhäuser Boston, Boston, MA, doi:10.1007/978-1-4612-0147-2. Available at <http://link.springer.com/10.1007/978-1-4612-0147-2>.
- [2] Jonathan Barrett & Matthew Leifer (2009): *The de Finetti Theorem for Test Spaces*. *New Journal of Physics* 11, pp. 1–10, doi:10.1088/1367-2630/11/3/033024. arXiv:0712.2265.
- [3] Titouan Carrette, Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2021): *Completeness of Graphical Languages for Mixed State Quantum Mechanics*. *ACM Transactions on Quantum Computing* 2(4), doi:10.1145/3464693.

- [4] Carlton M. Caves, Christopher A. Fuchs & Rüdiger Schack (2002): *Unknown Quantum States: The Quantum de Finetti Representation*. *Journal of Mathematical Physics* 43(9), pp. 4537–4559, doi:10.1063/1.1494475. arXiv:quant-ph/0104088.
- [5] Bob Coecke (2014): *Terminality Implies Non-Signalling*. *Electronic Proceedings in Theoretical Computer Science* 172, pp. 27–35, doi:10.4204/EPTCS.172.3.
- [6] Bob Coecke, Tobias Fritz & Robert W. Spekkens (2016): *A Mathematical Theory of Resources*. *Information and Computation* 250, pp. 59–86, doi:10.1016/j.ic.2016.02.008. arXiv:1409.5531.
- [7] Tobias Fritz, Tomáš Gonda & Paolo Perrone (2021): *De Finetti’s Theorem in Categorical Probability*. *Journal of Stochastic Analysis* 2(4), doi:10.31390/josa.2.4.06. arXiv:2105.02639.
- [8] Christopher A. Fuchs & Rüdiger Schack (2004): *Unknown Quantum States and Operations, a Bayesian View*. In Matteo Paris & Jaroslav Řeháček, editors: *Quantum State Estimation*, 649, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 147–187, doi:10.1007/b98673. arXiv:quant-ph/0404156.
- [9] Robert Furber & Bart Jacobs (2015): *From Kleisli Categories to Commutative C^* -algebras: Probabilistic Gelfand Duality*. *Logical Methods in Computer Science* 11(2), pp. 1–28, doi:10.2168/lmcs-11(2:5)2015.
- [10] Alain Guichardet (1969): *Tensor Products of C^* Algebras Part II. Infinite Tensor Products*. Retrieved September 18 2023 from <https://ncatlab.org/schreiber/files/GuichardetTensorProduct.pdf>.
- [11] Chris Heunen, Aleks Kissinger & Peter Selinger (2014): *Completely Positive Projections and Biproducts*. *Electronic Proceedings in Theoretical Computer Science* 171, pp. 71–83, doi:10.4204/EPTCS.171.7.
- [12] Edwin Hewitt & Leonard J. Savage (1955): *Symmetric Measures on Cartesian Products*. *Transactions of the American Mathematical Society* 80(2), p. 470, doi:10.2307/1992999.
- [13] R. L. Hudson (1981): *Analogs of de Finetti’s theorem and interpretative problems of quantum mechanics*. *Foundations of Physics* 11(9-10), pp. 805–808, doi:10.1007/BF00726951.
- [14] R. L. Hudson & G. R. Moody (1976): *Locally normal symmetric states and an analogue of de Finetti’s theorem*. *Z. Wahrschein. verw. Geb.* 33(4), pp. 343–351, doi:10.1007/BF00534784.
- [15] Mathieu Huot & Sam Staton (2019): *Universal Properties in Quantum Theory*. *Electronic Proceedings in Theoretical Computer Science* 287, pp. 213–223, doi:10.4204/EPTCS.287.12.
- [16] Bart Jacobs & Sam Staton (2020): *De Finetti’s Construction as a Categorical Limit*. In Daniela Petrişan & Jurriaan Rot, editors: *Coalgebraic Methods in Computer Science*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 90–111, doi:10.1007/978-3-030-57201-3_6.
- [17] Klaas Landsman (2017): *Foundations of Quantum Theory*. *Fundamental Theories of Physics* 188, Springer International Publishing, Cham, doi:10.1007/978-3-319-51777-3. Available at <http://link.springer.com/10.1007/978-3-319-51777-3>.
- [18] Saunders Mac Lane (1971): *Categories for the Working Mathematician*. 5, Springer New York, New York, NY, doi:10.1007/978-1-4612-9839-7.
- [19] Octavio Malherbe, Philip Scott & Peter Selinger (2013): *Presheaf Models of Quantum Computation: An Outline*. 7860, pp. 178–194, doi:10.1007/978-3-642-38164-5_13. arXiv:1302.5652.
- [20] Michael A. Nielsen & Isaac L. Chuang (2010): *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge.
- [21] Susumu Okada & Yoshiaki Okazaki (1978): *Projective limit of infinite Radon measures*. *Journal of the Australian Mathematical Society* 25(3), pp. 328–331, doi:10.1017/S1446788700021078.
- [22] Arthur J. Parzygnat (2020): *Inverses, disintegrations, and Bayesian inversion in quantum Markov categories*. doi:10.48550/arXiv.2001.08375.
- [23] John H. Selby, Carlo Maria Scandolo & Bob Coecke (2021): *Reconstructing Quantum Theory from Diagrammatic Postulates*. *Quantum* 5, p. 445, doi:10.22331/q-2021-04-28-445. arXiv:1802.00367.
- [24] Erling Størmer (1969): *Symmetric states of infinite tensor products of C^* -algebras*. *Journal of Functional Analysis* 3(1), pp. 48–68, doi:10.1016/0022-1236(69)90050-0.

- [25] Sean Tull (2018): *A Categorical Reconstruction of Quantum Theory*. *Logical Methods in Computer Science* 16(1), pp. 1–4, doi:10.23638/LMCS-16(1:4)2020. arXiv:1804.02265.
- [26] John van de Wetering (2019): *An Effect-Theoretic Reconstruction of Quantum Theory*. *Compositionality* 1(1), p. 1, doi:10.32408/compositionality-1-1. arXiv:1801.05798.

Annealing Optimisation of Mixed ZX Phase Circuits

Stefano Gogioso
Hashberg Ltd
quantum@hashberg.io

Richie Yeung
Quantinuum Ltd
richie.yeung@cambridgequantum.com

We present a topology-aware optimisation technique for circuits of mixed ZX phase gadgets, based on conjugation by CX gates and simulated annealing.

1 Introduction

In this work, we build upon the Master’s thesis by one of the authors [18] and present a topology-aware optimisation technique for circuits of mixed ZX phase gadgets, based on conjugation by CX gates and simulated annealing. The basic rules on CX conjugation of phase gadgets have previously appeared in the literature [2, 3, 7, 11, 14]—which features other topology-aware techniques—and are used by both the PyZX library [9, 10] and the t|ket> compiler [15, 16]. We test the performance of our optimisation technique on random circuits of mixed ZX phase gadgets. An open-source implementation is made available as part of the Python library pauliopt [6], which is also used to generate the circuit figures in this work.

With NISQ applications in mind, our optimisation target is the number $\text{count}_{\text{CX}}(P)$ of nearest-neighbour (NN) CX gates required to implement a mixed ZX phase circuit P on a given topology, making the following assumptions on compilation:

- Each phase gadget will be compiled to a single-qubit rotation conjugated by trees of CX gates.
- Long-range CX gates in the topology will be compiled to double-ladders of NN CX gates.

In these circumstances, the CX count of an individual phase gadget depends on the mutual distances between its legs. Specifically, it is computed by finding a minimum spanning tree with weights related to the distance of leg qubits in the given topology.

Our optimisation technique is based on the observation that conjugating a given mixed ZX phase circuit P with an arbitrary CX circuit C has the effect of changing the legs of the individual gadgets, without otherwise altering their angles or position within the circuit. The resulting circuit $C^\dagger(P) := C^\dagger \circ P \circ C$ is again a mixed ZX phase circuit, related to the original circuit by:

$$P = C \circ C^\dagger(P) \circ C^\dagger$$

However, the different gadget legs in $C^\dagger(P)$ might result in a lower overall CX count:

$$2 \cdot \text{count}_{\text{CX}}(C) + \text{count}_{\text{CX}}(C^\dagger(P))$$

Finding a CX circuit C_{opt} which (approximately) minimises the overall CX count above results in an optimised implementation of our initial mixed ZX phase circuit P , expressed as conjugation of another mixed ZX phase circuit $C_{opt}^\dagger(P)$ by the CX circuit C_{opt} :

$$P = C_{opt} \circ C_{opt}^\dagger(P) \circ C_{opt}^\dagger$$

We use simulated annealing to explore the space of conjugating CX circuits and find an approximately optimal C_{opt} , starting from the empty CX circuit. We fix a number of layers and explore the space by progressively introducing or removing NN CX gates within those layers, altering the overall CX count in the process. In the early stages of optimisation, the annealing “temperature” t is high and we are free to explore the configuration space, even at the expense of a—hopefully temporary—increase in overall CX count. As the optimisation progresses, the temperature t is lowered and the search becomes progressively more greedy, accepting changes which increase the CX count by Δ with progressively lower probability:

$$\text{Prob}(\text{accept CX count increase } \Delta) = \frac{1}{2^{\Delta/t}}$$

Our technique is applicable to parametric circuits, such as the ansatzes used in quantum machine learning, adiabatic quantum computation [5] and quantum approximate optimisation [4]. Such ansatzes typically consist of a large number of repeating layers, using the same phase gadgets with possibly different parameters. In this context, a key observation about our technique is that a single conjugating CX circuit is sufficient to optimise all layers at once. The number K of layers can then be absorbed into the overall CX count calculation, with a single layer L being optimised:

$$2 \cdot \text{count}_{\text{CX}}(C) + K \cdot \text{count}_{\text{CX}}(C^\dagger(L))$$

As a consequence, the optimisation cost scales with the size of L , regardless of K , making our technique especially suited for application in the aforementioned domains.

2 ZX Phase Gadgets

A *Z phase gadget* on n qubits is the exponential of an imaginary scalar multiple of an element of the Pauli group P_n on n qubits involving only the Z and I Pauli matrices (and with scalar factor $+1$):

$$\exp\left(i\theta \bigotimes_{q=0}^{n-1} G_q\right) \quad \text{where} \quad \begin{cases} G_q = Z & \text{if qubit } q \text{ is a leg} \\ G_q = I & \text{otherwise} \end{cases}$$

The element $\bigotimes_{q=0}^{n-1} G_q \in P_n$ is known as the *generator*, the parameter $\theta \in U(1)$ is known as the *angle*, and the qubits q where $G_q = Z$ are known as the *legs* of the gadget. An *X phase gadget* on n qubits is analogously defined, using the Pauli matrix X in place of Z. Below are two examples of phase gadgets on 3 qubits:

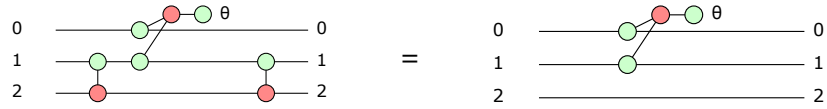
$$\exp\left(i\frac{\pi}{4} Z \otimes I \otimes Z\right) \qquad \exp\left(i\frac{3\pi}{4} X \otimes X \otimes X\right)$$

Phase gadgets can be represented in the ZX calculus by attaching spiders for the gadget basis to each leg and joining them at a “hub”, formed by spiders of alternating colours and including the angle ($\theta = \frac{\pi}{4}$ and $\theta = \frac{3\pi}{4}$ respectively) as a spider phase. Below are the same two examples represented in the ZX calculus (Z phase gadget on the left, X phase gadget on the right):

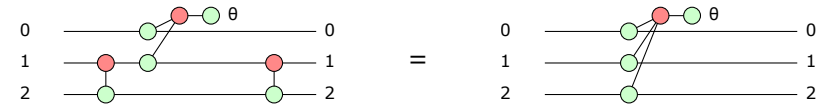


2.1 Conjugation by CX gates

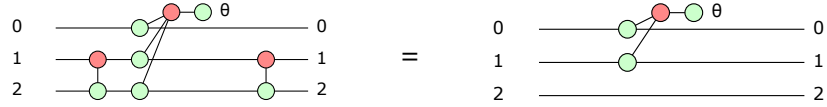
Z and X phase gadgets are particularly well-behaved under conjugation by CX gates: they either keep the same legs, they gain a leg, or they lose a leg, but basis and angle are always left unchanged. Instead of thinking of CX gates in terms of *control* and *target*, we think of them as having a Z qubit (the control) and an X qubit (the target). Z phase gadgets are left unchanged under conjugation if the X qubit is not a leg:



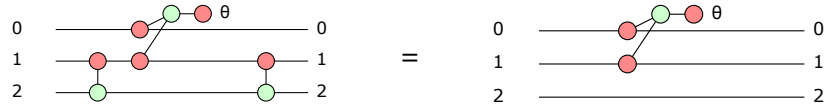
If the X qubit is a leg and the Z qubit is not a leg before conjugation, the Z qubit becomes a leg after conjugation:



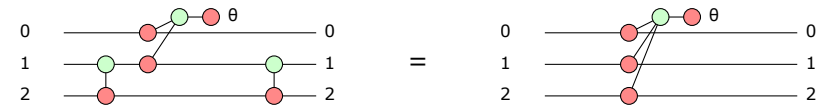
If the X qubit is a leg and the Z qubit is a leg before conjugation, the Z qubit is no longer a leg after conjugation:



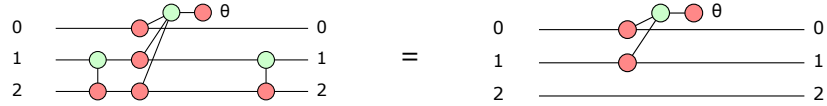
The behaviour of X phase gadgets is obtained by inverting Z and X everywhere. X phase gadgets are left unchanged under conjugation if the Z qubit is not a leg:



If the Z qubit is a leg and the X qubit is not a leg before conjugation, the X qubit becomes a leg after conjugation:



If the Z qubit is a leg and the X qubit is a leg before conjugation, the X qubit is no longer a leg after conjugation:



There are several different ways to obtain the conjugation rules above. A simple one goes through the observation that conjugation commutes with matrix exponentiation:

$$A \left(\exp \left(i\theta \bigotimes_{q=0}^{n-1} G_q \right) \right) A^\dagger = \exp \left(i\theta A \left(\bigotimes_{q=0}^{n-1} G_q \right) A^\dagger \right)$$

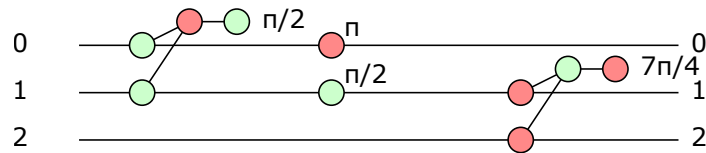
From this, it is enough to look at conjugation by CX for a handful of Pauli group elements:

$$\begin{aligned} \text{cx}_{0,1}(\text{Z} \otimes \text{I})\text{cx}_{0,1} &= \text{Z} \otimes \text{I} & \text{cx}_{0,1}(\text{I} \otimes \text{Z})\text{cx}_{0,1} &= \text{Z} \otimes \text{Z} & \text{cx}_{0,1}(\text{Z} \otimes \text{Z})\text{cx}_{0,1} &= \text{I} \otimes \text{Z} \\ \text{cx}_{0,1}(\text{I} \otimes \text{X})\text{cx}_{0,1} &= \text{I} \otimes \text{X} & \text{cx}_{0,1}(\text{X} \otimes \text{I})\text{cx}_{0,1} &= \text{X} \otimes \text{X} & \text{cx}_{0,1}(\text{X} \otimes \text{X})\text{cx}_{0,1} &= \text{X} \otimes \text{I} \end{aligned}$$

2.2 Mixed ZX Phase circuits

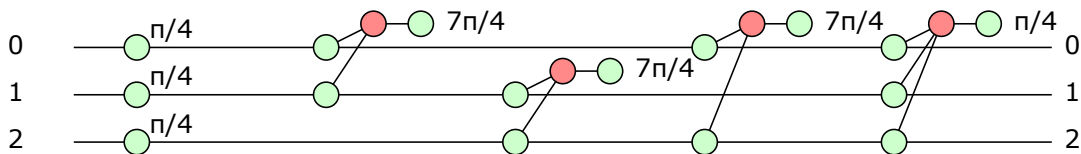
A *mixed ZX phase circuit* is, as the name suggests, a quantum circuit consisting of a mix of Z phase gadgets and X phase gadgets. Examples of mixed ZX phase circuits include the ansatz used by adiabatic quantum computation [5], the quantum approximate optimisation algorithm [4], as well as many quantum machine learning techniques [18]. The `pauliopt` library allows these circuits—called *phase circuits*, for short—to be created directly from a sequence of phase gadgets with given angles and legs:

```
circ = PhaseCircuit(3)
circ >>= Z(pi/2)@{0,1}
circ >>= X(pi)@{0}, Z(pi/2)@{1}
circ >>= X(-pi/4)@{1,2}
```



Utility methods are also made available to translate common gates into phase gadgets:

```
circ = PhaseCircuit(3)
circ.ccz(0,1,2)
```



A generalisation of Euler's decomposition to n qubits (technically, a consequence of Zassenhaus formula [1, 13]) implies that every unitary on n qubits can be expressed using a bounded number of Z and X phase gadgets, meaning that circuits of mixed ZX phase gadgets are universal.

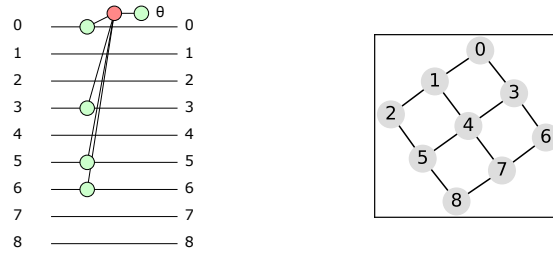
3 Annealing optimisation

3.1 Topology-aware cost of phase gadgets

The optimisation method targets the CX count for phase gadget implementation on a given topology, which we refer to as its *cost*. The cost of a gadget relies on its implementation using a balanced tree of CX gates between its legs [2], individually converted into double ladders of nearest-neighbour (NN) CX gates (without further simplification). The balanced tree of minimum CX count is obtained using Prim's algorithm for minimum spanning trees, using the following weights for any two distinct leg qubits q_i and q_j in the phase gadget:

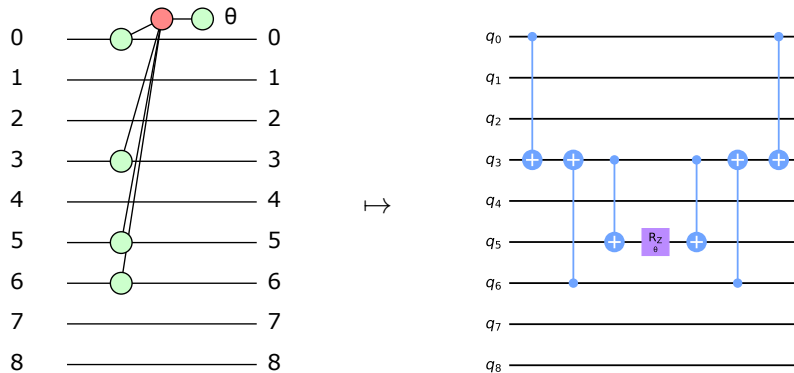
$$w(q_i, q_j) := 4d(q_i, q_j) - 2$$

where $d(q_i, q_j)$ is the graph distance between qubits q_i and q_j in the given topology. For example, consider the following 4-legged gadget on a 3-by-3 grid topology:



The minimum spanning tree implementation for this gadget uses two (adjoint) balanced trees of 3 CX gates each—between qubits $\{0, 3\}$ (NN CX count: 1), between qubits $\{3, 6\}$ (NN CX count: 1), and between qubits $\{3, 5\}$ (NN CX count: 3)—for a total implementation cost of 10 NN CX gates. In this case, it is easy to see that every alternative would have been sub-optimal, because the only common nearest neighbour between qubits 0 and 6 on the grid is qubit 3: connecting both qubits to a qubit at distance 2 from both would have incurred a NN CX count of at least $3 + 3 = 6$, while connecting one qubit to a different nearest neighbour of the other (e.g. connecting 0 and 6 to 1) would have incurred a NN CX count of at least $5 + 1 = 6$, both already exceeding the optimal overall NN CX count of 5. The following snippet produces the phase gadget in Qiskit [17] using the above minimum spanning tree:

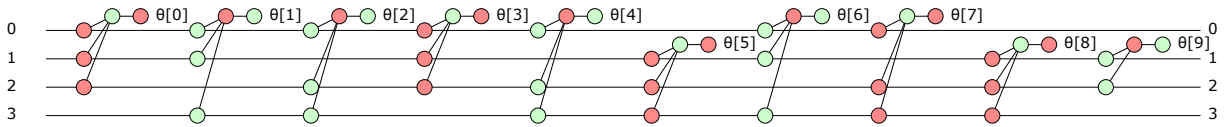
```
from qiskit.circuit import QuantumCircuit, Parameter
cx_tree = QuantumCircuit(4)
cx_tree.cx(0, 2)
cx_tree.cx(1, 2)
cx_tree.cx(2, 3)
gadget = QuantumCircuit(8)
gadget.compose(cx_tree, qubits=(0,6,3,5), inplace=True)
gadget.rz(Parameter("θ"), 5)
gadget.compose(cx_tree.inverse(), qubits=(0,6,3,5), inplace=True)
```



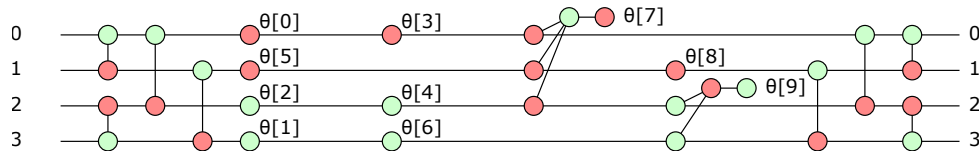
3.2 Conjugation by CX circuits

As discussed above, conjugation of mixed ZX phase circuits by CX circuits preserves the order and angles of the gadgets, but modifies their legs. For adequate choices of nearest-neighbour CX circuits, the

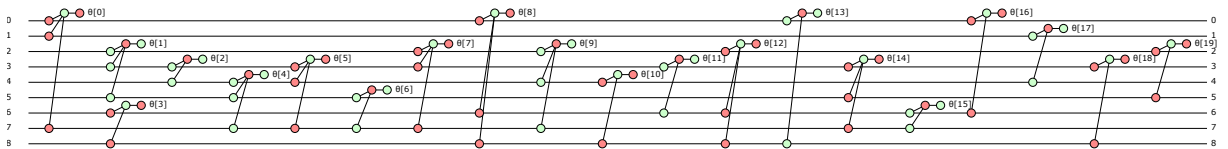
increase in CX count from the conjugating CX gates is counterbalanced by a decrease in implementation cost for the phase gadgets on a given topology, due to leg changes and reduction. The overall 2-qubit gate count can then be used as a cost function for global optimisation methods (such as simulated annealing). Here is example of a 10-gadget circuit on 4 qubits:



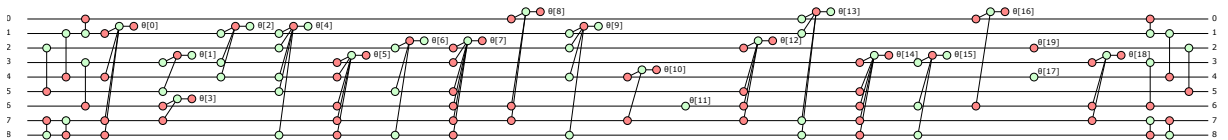
On a cycle topology, the circuit above has a CX count of 42. Below is an equivalent circuit with a CX count of 14, obtained by conjugation from two layers of nearest-neighbour CX gates:



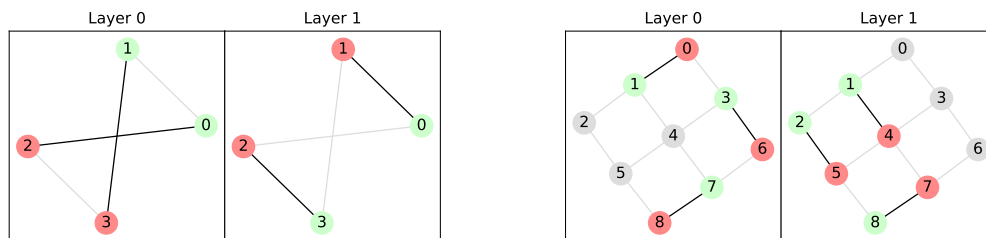
Here is a second example, of a 20-gadget circuit on 9 qubits:



On a 3-by-3 grid topology, the circuit above has a CX count of 142. Below is an equivalent circuit with a CX count of 126, obtained by conjugation from two layers of nearest-neighbour CX gates:



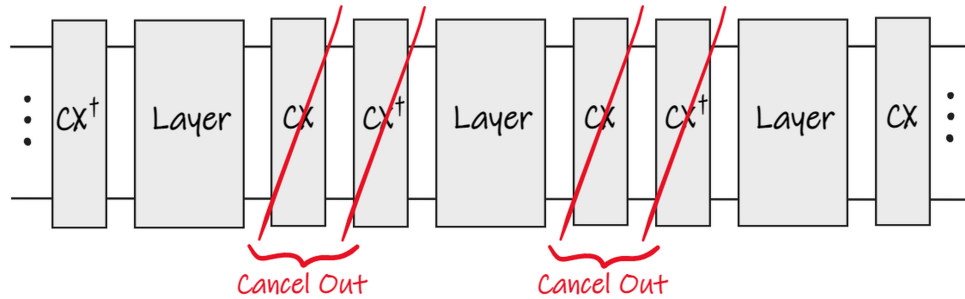
The two layers of conjugating CX gates for each one of the two example above, arranged on the corresponding qubit topologies, are displayed below:



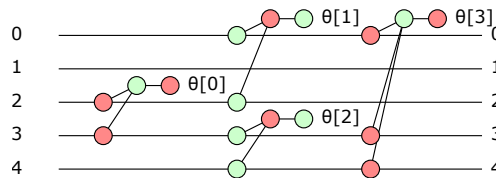
3.3 Repeating layers

A remarkable feature of our optimisation technique concerns its use with circuits composed of repeating layers. For such circuits, it is enough to simplify an individual layer and then repeat it: the conjugating

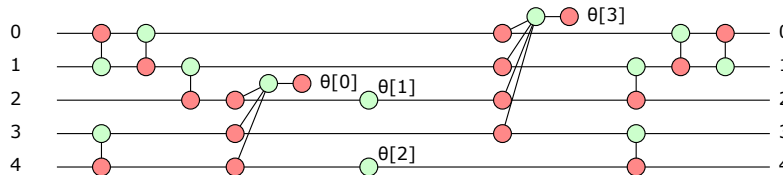
CX gates between repeating layers cancel out in pairs, leaving a single pair of conjugating CX block for the entire circuit.



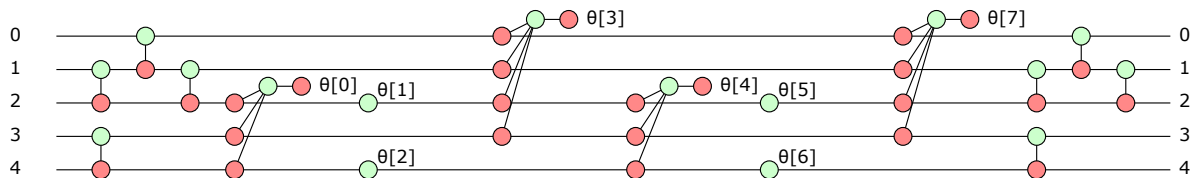
This means that the per-layer CX cost reduction is multiplied by the number of layers, while the CX cost for the conjugating blocks is fixed: as the number of layers grows, the cost of larger conjugating CX blocks is offset by the increased savings on the layers, at no additional computational expense for the optimisation itself. As a concrete example, consider the following 4-gadget layer:



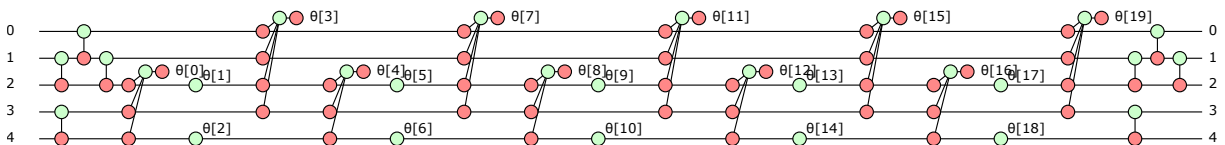
On a line topology with 5 qubits, the initial CX count for this layer is 22. If optimisation is run for a single layer, the CX count is reduced by 18%, from 22 to 18:



With 2 layers, the CX count is reduced by 36%, from 44 to 28:



With 5 layers, the CX count is reduced by 47%, from 110 to 58:



The 8 conjugating CX gates are independent of the number of layers, leaving 10 CX gates per simplified layer from an initial count of 22. In the limit of a large number of layers, the relative CX count reduction for this conjugating CX block approaches $12/22$, or 54.5%.

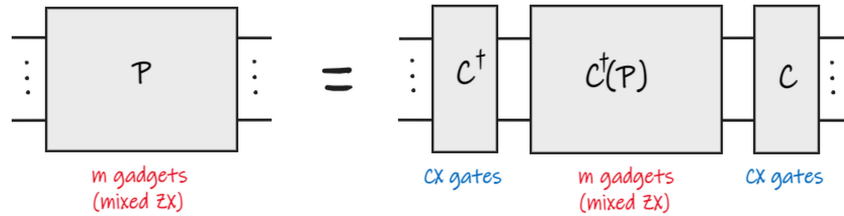
This feature of our optimisation technique makes it particularly interesting for applications in quantum machine learning (QML), adiabatic quantum computation and the quantum approximate optimisation algorithm (QAOA), where circuit ansatz often feature the aforementioned repeating layers structure. In particular, note that the technique is insensitive to the specific angles used in the repeated layers: the angles can be different, and even parametric (as indeed shown in the examples above). This means that an ansatz circuit only need be optimised once, before assigning specific values to its parameter. For a mixed ZX phase circuit formulation of several such ansatz, we refer the reader to [18].

3.4 Configuration space

The configuration space explored by our optimisation algorithm on a given topology consists of all circuits of nearest-neighbour CX gates with a fixed number of layers, which we refer to as the *CX blocks*. At any given point in configuration space, i.e. at any such CX block C , the cost for a phase circuit P is given by sum of:

- the CX count for C^\dagger ;
- the CX counts for the individual gadgets of $C^\dagger(P)$, computed on the given topology;
- the CX count for C ;

where $C^\dagger(P) := C^\dagger \circ P \circ C$ is the phase circuit obtained from P through conjugation by C^\dagger .



For fixed topology and number of layers, we say that a CX block C' is obtained from another CX block C by *flipping* a CX gate $cx_{i,j}$ on a layer l if either:

- the gate $cx_{i,j}$ is present at layer l in C , and C' is obtained from C exactly by removing $cx_{i,j}$ from layer l ;
- there is no gate at layer l in C which is incident to either qubit i or qubit j (or both), and C' is obtained from C exactly by adding $cx_{i,j}$ to layer l .

We write $C \xrightarrow{l;i,j} C'$ to denote this fact; because CX gates are self-inverse, this is equivalent to $C \xleftarrow{l;i,j} C'$. We consider two CX blocks to be nearest neighbours in configuration space if one can be obtained from the other by a single CX gate flip: this endows the configuration space with the structure of an undirected graph, where edges are labelled by the triples $(l;i,j)$ describing the CX gate flips. Our optimisation algorithm performs a random walk on this graph, starting from the empty CX block and attempting to (globally) minimise the cost for a given phase circuit.

3.5 Simulated Annealing

Simulated annealing is a global optimisation method which explores a discrete problem-dependent configuration space \mathcal{X} , in an attempt to find an approximate optimum for a an arbitrary cost function f . At each step of the optimisation, the algorithm sits at some configuration $x \in \mathcal{X}$ and selects a uniformly

random “direction” to explore; that is, it samples a uniformly random $x' \in N(x)$, where $N(x) \subseteq \mathcal{X} \setminus \{x\}$ a bounded set of nearest neighbours of x in configuration space. The algorithm computes $f(x')$ and compares it to $f(x)$: if $f(x') \leq f(x)$, then the algorithm moves to x' deterministically; otherwise, it moves to x' with a probability which decreases exponentially in the difference $\Delta := f(x') - f(x)$:

$$\text{Prob}(x \rightarrow x') = \exp\left(-\frac{\Delta \log(2)}{t}\right)$$

The parameter $t > 0$ is known as the *temperature*: it decreases from an initial value t_0 to a final value $t_N \approx 0$ over N iterations, in a way specified by the “temperature schedule”. The temperature schedule for the annealing determines the probability of accepting changes which increase the cost function: an increase of $t \log_2(1/p)$ has probability p of being accepted. At the start of the annealing, when the temperature is high, the random walk is allowed to explore the configuration space, even at the cost of an increase in cost. As iterations progress and the temperature is lowered, the random walk progressively favours those steps which decrease the cost. Towards the end, the probability of accepting steps which increase the cost becomes vanishingly small, and the algorithm is reduced to a greedy optimisation.

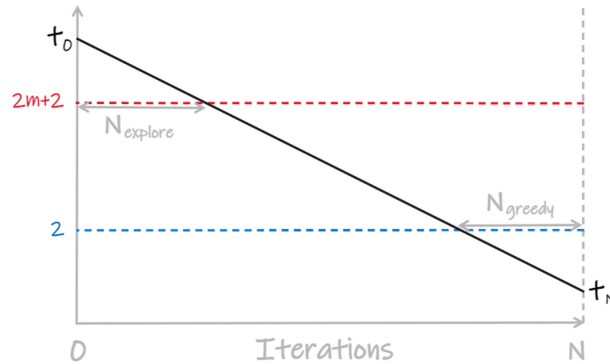
3.6 Simulated Annealing for Mixed ZX Phase Circuit Optimisation

For our problem, the simulated annealing configuration space consists of a fixed number of layers of NN CX gates on the given topology, with nearest neighbours in configuration space determined by CX gate flips in any one of the layers. Given a mixed ZX phase circuit, a topology and a fixed number of layers for the CX blocks, we construct an optimized circuit container; initially, this consists of the phase circuit conjugated by empty CX blocks. To minimise the cost of over the space of CX blocks, we perform a random walk using simulated annealing [8], for a given number of steps/iterations and following a given temperature schedule. At the end of the annealing, the optimized circuit contains a simplified phase circuit conjugated by (typically) non-empty CX blocks. Every iteration of our simulated annealing algorithm proceeds as follows:

1. We obtain the current temperature t from the temperature schedule using the current iteration number (e.g. linearly interpolating from initial temperature to final temperature).
2. We select a random CX gate flip for the current block, i.e. select a random neighbour.
3. We flip the selected CX gate (see Subsection 3.7 below).
4. We calculate the CX count c_{new} for the new circuit and compare it to the previous CX count c_{old} , by computing $\Delta := c_{new} - c_{old}$ and sampling a uniformly random number $r \in [0, 1]$:
 - if $r \leq \exp\left(-\frac{\Delta \log(2)}{t}\right)$, and in particular if $c_{new} \leq c_{old}$, the move is accepted and we go to the next iteration from the new CX block;
 - otherwise, the move is undone—by flipping the selected CX gate again—and we go the next iteration from the old CX block.

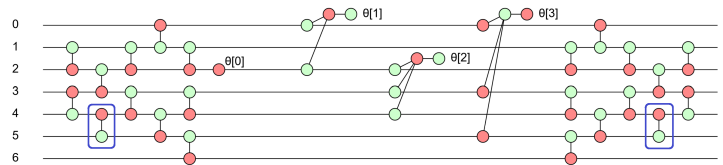
Our selection of temperature schedule depends on the shape of the temperature curve (linear, geometric, reciprocal or logarithmic) and two parameters: the amount of time we want to spend exploring the configuration space at the start of the annealing and the amount of time we want to spend performing greedy optimisation at the end of the annealing. As a convention, we take the temperature value $t = 2m + 2$ (where m is the number of phase gadgets) to denote the end of the exploration phase and the temperature value $t = 2$ to denote the start of the greedy optimisation phase. Because the CX gates are

nearest-neighbour, $2m + 2$ is the maximum possible increase in CX count from a single CX gate flip: 2 CX gates added plus a leg added to all gadgets, at a worst-case CX count increase of 2 per gadget. On the other hand, 2 is the minimum possible (positive) increase: CX count always changes by multiples of 2. This defines the exploration phase as the time where every CX gate flip has a probability higher than 50% of being accepted, and the greedy optimisation phase as the time where every increase in CX count has a probability lower than 50% of being accepted. As an example, below is a sketch of a linear temperature schedule over N iterations:



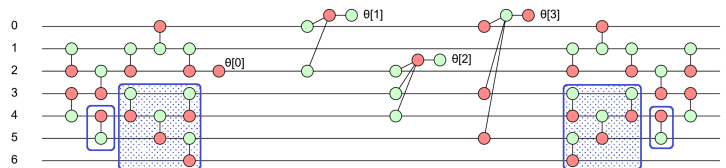
3.7 Flipping CX gates

Because simulated annealing requires thousands of iterations, optimised execution of CX gate flips is paramount. Naively, one could simply maintain the current CX block C and compute the conjugated circuit from scratch at every iteration: this requires conjugating every gadget in the phase circuit by every CX gate in the CX block, even though the CX block itself has only changed by one gate flip. In this work, we adopt a different strategy, where we maintain both the current CX block C and the current conjugated phase circuit $C^\dagger(P)$. Imagine we wish to perform a gate flip $C \xrightarrow{l;i,j} C'$, e.g. performing the flip $C \xrightarrow{3;5,4} C'$ boxed in dark blue below.

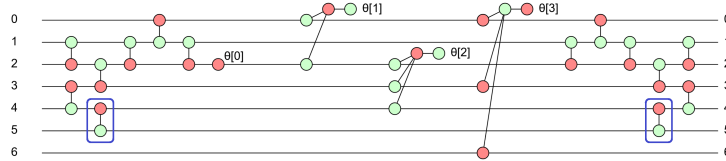


To perform the flip, we go through the following 4 steps:

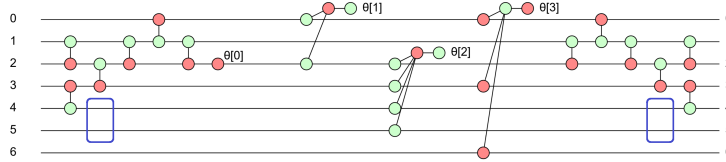
1. Starting from layer $l - 1$ and moving inwards towards layer 0, we create the (partially ordered) set $(l; i, j)_{\downarrow}$ of all CX gates that are strictly in the “past” of gate $cx_{i,j}$ at layer l . These are boxed and highlighted below.



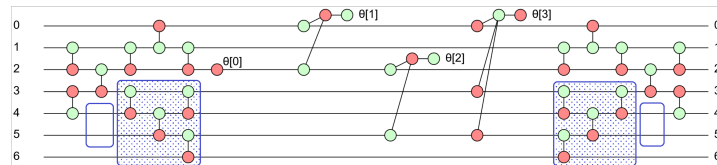
2. We undo all the CX gates in $(l; i, j)_{\downarrow}$, conjugating the phase circuit by each one, working our way outwards from layer 0 (closest to the phase circuit) up to layer $l - 1$ (just below the gate we'll flip).



3. We flip gate $cx_{i,j}$ at layer l , conjugating the phase circuit by $cx_{i,j}$ (adding $cx_{i,j}$ to layer l if it wasn't there, removing it from layer l if it was). Below, we remove $cx_{5,4}$ from layer 3.



4. We redo all the CX gates in $(l; i, j)_{\downarrow}$, conjugating the phase circuit by each one, this time working our way inwards from layer $l - 1$ down to layer 0.



To define the poset $(l; i, j)_{\downarrow}$ of past gates formally, consider the partial order on the gates $(l; x, y)$ of the CX block obtained by transitive-reflexive closure of the following relation:

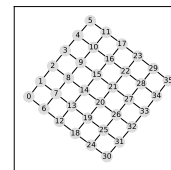
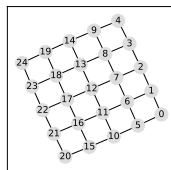
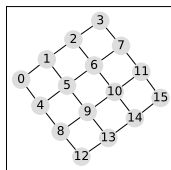
$$(l_1; x_1, y_1) < (l_2; x_2, y_2) \Leftrightarrow l_1 < l_2 \text{ and } \{x_1, y_1\} \cap \{x_2, y_2\} \neq \emptyset$$

The poset $(l; i, j)_{\downarrow}$ consists of all gates strictly less than $(l; i, j)$ in this partial order.

4 Performance Evaluation

4.1 Random mixed ZX phase circuits

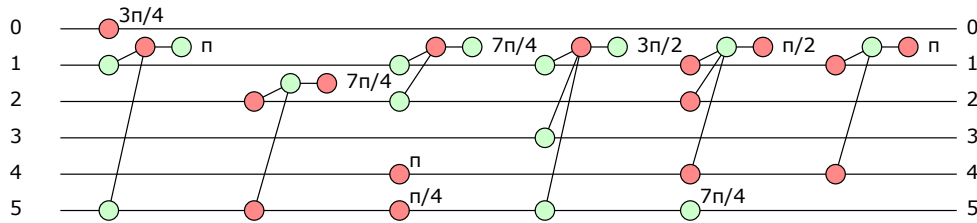
To evaluate the performance of our technique, we test it on random mixed ZX phase circuits with a varying number 2- and 3-legged gadgets, on 16, 25 and 36 qubits arranged in a square grid.



Our random model samples a fixed number of gadgets `num_gadgets` on a fixed number of qubits `num_qubits`, selecting legs in a independent, identically distributed way for each gadget. Legs for a gadget are sampled by first selecting the number of legs uniformly randomly in a given range (from `min_legs` to `max_legs`, both included), and then selecting a uniformly random subset of qubits with the desired cardinality.

`PhaseCircuit.random(num_qubits, num_gadgets, *, min_legs, max_legs)`

Below is an example of a random circuit of 10 gadgets on 6 qubits, with between 1 and 3 legs:



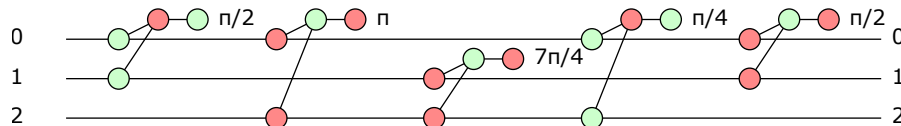
We can also make our random circuits purely parametric, but this doesn't impact our benchmarking.

4.2 Internal representation of mixed ZX phase circuits

The internal representation of phase circuits in our technique is optimised for execution of the the annealing algorithm. A circuit (n qubits, m_z Z gadgets and m_x X gadgets, in any order) is broken down into the following components:

- a pair of binary matrices (an $n \times m_z$ matrix L_z for Z gadgets and an $n \times m_x$ matrix L_x for X gadgets) encoding the positions of the legs;
- a pair of lists (length m_z for Z gadgets and length m_x for X gadgets) mapping the columns of each matrix to the position of the corresponding gadget in the circuit;
- a list (length $m_z + m_x$) of angles for the gadgets (concrete or parametric).

The annealing algorithm operates on gadget legs only—i.e. on the the two matrices—without any alteration to the original gadget order and angles. The algorithm further acts on the columns of each matrix independently, leaving ample scope for caching (implemented) and GPU parallelisation (not yet implemented). As an example, consider the following circuit of $m_z + m_x = 5$ gadgets on $n = 3$ qubits:



The matrices for this circuit are as follows:

$$L_z = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad L_x = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

The lists mapping columns to gadgets are $(0, 3)$ for L_z and $(1, 2, 4)$ for L_x . The list of angles for the gadgets is $(\pi/2, \pi, 7\pi/4, \pi/4, \pi/2)$.

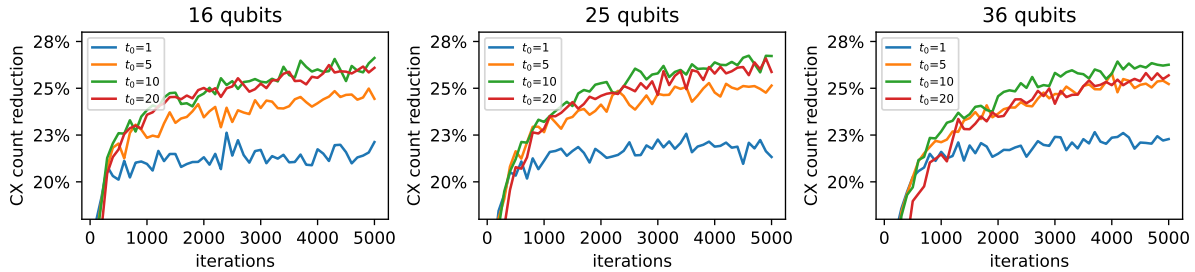
4.3 Benchmarking Results

In all benchmarks below, we proceed as follows:

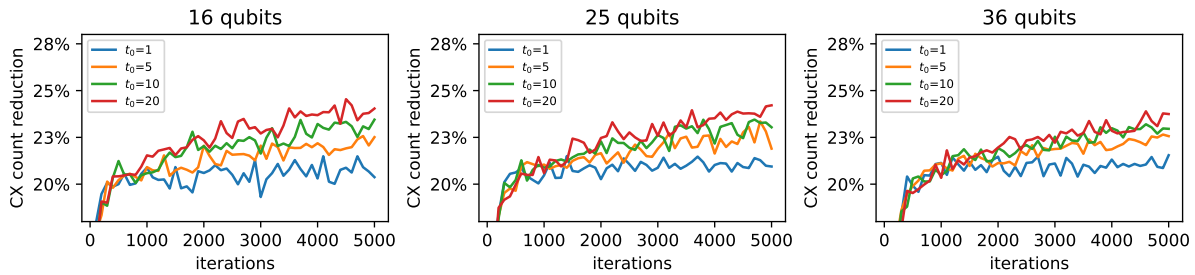
1. We generate 50 random mixed ZX phase circuits, for various combinations of hyper-parameters—number of qubits, number of gadgets per layer, initial temperature, schedule type—randomly sampling between 2-legged and 3-legged gadgets. The number of CX block layers is fixed to 3.

2. We run annealing optimisation with varying number of iterations, from 100 to 5000, in increments of 100. The number of circuit layer repetitions is fixed to 5.
3. We look at the reduction in NN CX count—in average or in distribution—from each original random circuit to the corresponding optimised circuit, relative to the NN CX count of the original circuit. The NN CX count is computed using Prim’s algorithm for minimum spanning trees.

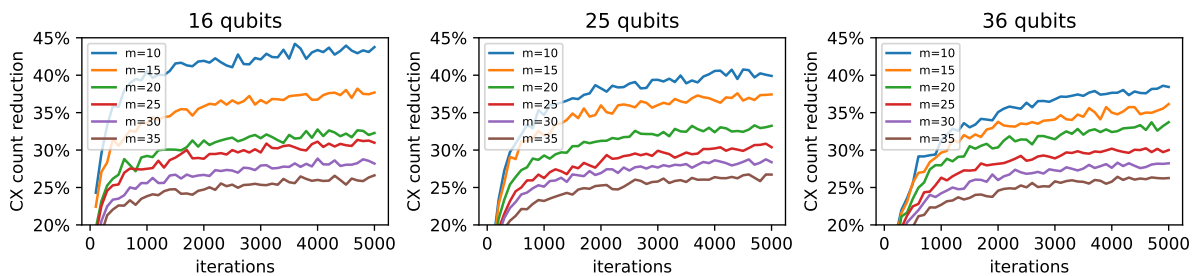
To start with, we look at the performance for four different initial temperatures $t_0 = 1, 5, 10, 20$ and two annealing schedules (linear and geometric). Below is the average CX count reduction over all runs for the linear annealing schedule at the four initial temperatures.



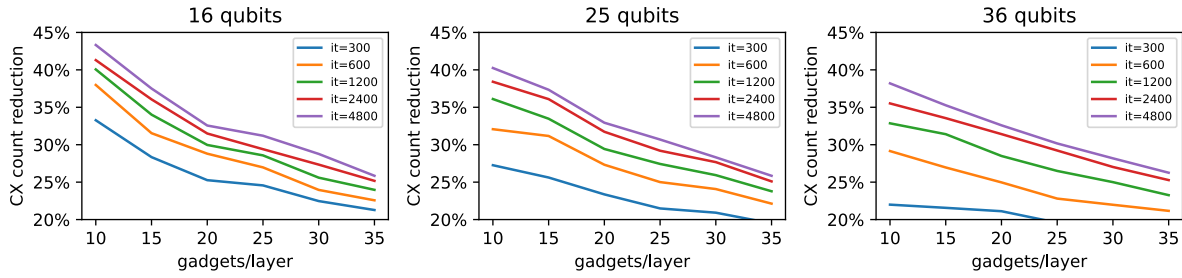
Below is the average CX count reduction over all runs for the geometric annealing schedule at the four initial temperatures. Because of the inferior average performance of other combinations, we restrict our attention to linearly annealed runs with $t_0 = 10$ for the remainder of this section.



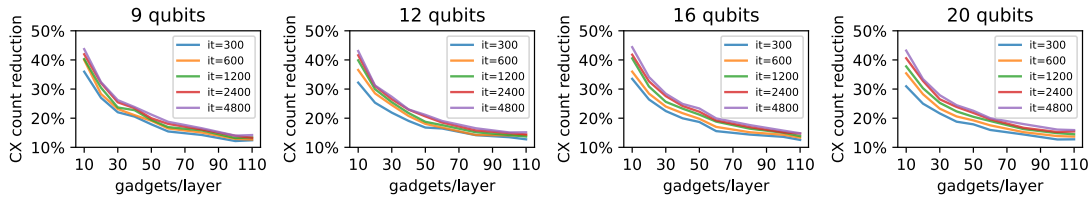
For our optimisation, we always use 5 layer repetitions for the phase circuit and 3 layers for the conjugating CX blocks. Below is the average CX count reduction—as a percentage of the initial CX count—for varying number m of phase gadgets per circuit layer, ranging from 10 to 35 in increments of 5.



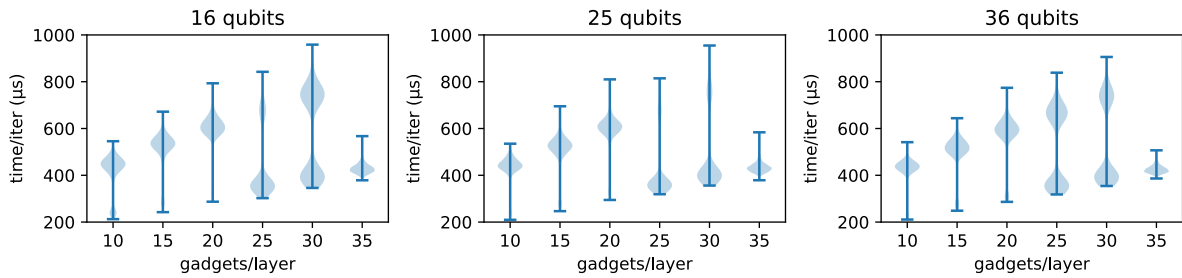
The average CX count reduction degrades as the number of gadgets/layer increases, decreasing by approximately 0.5% per additional gadget on 36 qubits.



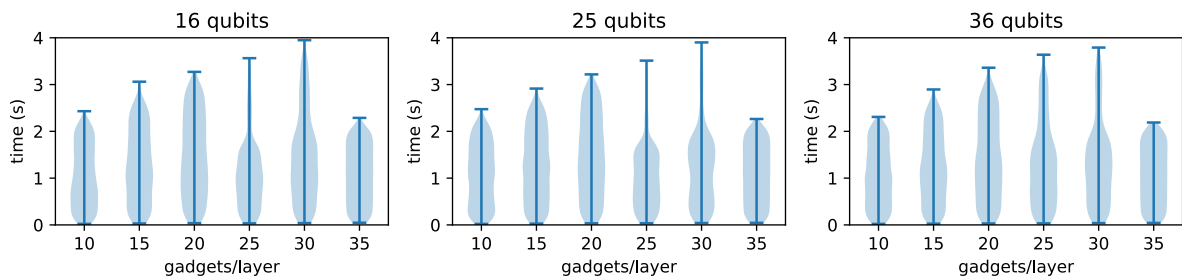
This behaviour is roughly linear when the number of gadgets/layer is small compared to the number of qubits, but approaches a horizontal asymptote as the number of gadgets/layer increases. The following graphs showcase this behaviour over smaller grid topologies (3×3 , 3×4 , 4×4 and 4×5 respectively).



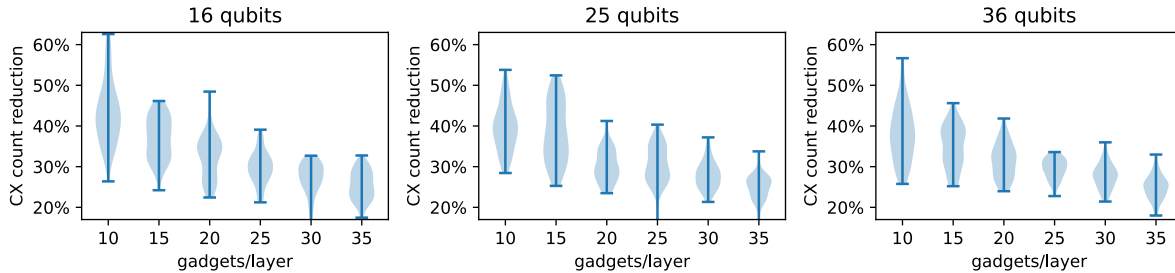
The optimised flipping of CX gates allows us to avoid recomputing the whole conjugated circuit every time. We also implement some basic caching techniques for gadget legs, and use a modified version of Prim’s algorithm to efficiently compute the topologically-aware CX count for each gadget. Our Python implementation is reasonably efficient, taking between $400\mu\text{s}$ and $800\mu\text{s}$ per iteration.



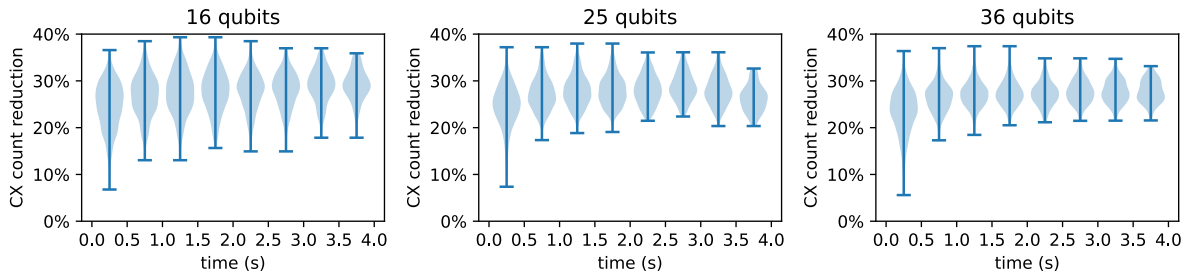
Over the full range of iterations we considered, from 100 to 5000, our experiments never took more than 4s, with the vast majority under 3s.



To evaluate the performance of our approach under strict time constraints, we restrict our attention to the runs that finished under 1s, using almost all the time they were allowed (in the 90th to 100th percentile within runs under 1s for the same number of qubits and gadget/layer). We observe that the average CX count reduction decreases as the number of gadgets/layer increases, and that the distribution narrows.



To further evaluate the effect of time on our optimisation, we restrict our attention to runs at 30 gadgets/layer, partitioned into time bins with 0.5s resolution. The average CX count reduction improves only slightly as more time is allowed, indicating diminishing returns for progressively higher number of iterations. However, we observe a marked improvement of worst-case CX count reduction from anneals under 0.5s to anneals up to 1s, with only marginal improvement after that point.



5 Future Work

Our first avenue for future work concerns the annealing itself: rather than using simulated annealing on classical hardware, we have started working on a formulation of the problem suitable for execution on D-Wave quantum annealers. This would allow our exploration of the space of configurations to efficiently scale to much larger examples, involving hundreds of gadgets and qubits. It would also make our technique a “quantum optimisation of quantum circuits”, which is fun to say. Thanks to QPL 2022 Anonymous Reviewer #1 for pointing us to [12], which provides an Ising formulation for the minimum spanning tree objective.

Our second avenue for future work concerns the scope and method of optimisation: instead of limiting ourselves to mixed phase gadgets and random CX circuits, we have generalised our observation to arbitrary layers of Pauli gadgets using random Clifford circuits. We are in the process of implementing this technique and will report on its performance in future work. We also plan to incorporate commutation between phase gadgets in different bases to further improve performance.

We will benchmark our techniques explicitly against previous literature, using realistic circuit families from quantum chemistry, adiabatic quantum computation, quantum approximate optimisation and quantum machine learning.

Acknowledgements

The authors would like to acknowledge financial support by Hashberg Ltd for the execution of numerical experiments. The authors would like to thank the three anonymous QPL 2022 reviewers for their very helpful comments and suggestions.

References

- [1] Fernando Casas, Ander Murua & Mladen Nadinic (2012): *Efficient computation of the Zassenhaus formula*. *Computer Physics Communications* 183(11), pp. 2386–2391, doi:10.1016/j.cpc.2012.06.006.
- [2] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons & Seyon Sivarajah (2020): *Phase gadget synthesis for shallow circuits*. *EPTCS* 318, doi:10.4204/EPTCS.318.13.
- [3] Alexander Cowtan, Will Simmons & Ross Duncan (2020): *A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz*. *arXiv preprint*, doi:10.48550/arXiv.2007.10515.
- [4] Edward Farhi, Jeffrey Goldstone & Sam Gutmann (2014): *A quantum approximate optimization algorithm*. *arXiv preprint*, doi:10.48550/arXiv.1411.4028.
- [5] Edward Farhi, Jeffrey Goldstone, Sam Gutmann & Michael Sipser (2000): *Quantum computation by adiabatic evolution*. *arXiv preprint quant-ph/0001106*, doi:10.48550/arXiv.quant-ph/0001106.
- [6] Stefano Gogioso & Richie Yeung (2021): *pauliopt Library*. Available at <https://github.com/sg495/pauliopt>.
- [7] Arianne Meijer-van de Griend & Ross Duncan (2020): *Architecture-aware synthesis of phase polynomials for NISQ devices*. *arXiv preprint*, doi:10.48550/arXiv.2004.06052.
- [8] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi (1983): *Optimization by Simulated Annealing*. *Science* 220(4598), pp. 671–680, doi:10.1126/science.220.4598.671.
- [9] Aleks Kissinger & John van de Wetering (2019): *pyzx library*. Available at <https://github.com/Quantomatic/pyzx>.
- [10] Aleks Kissinger & John van de Wetering (2020): *Pyzx: Large scale automated diagrammatic reasoning*. *EPTCS* 318, doi:10.4204/EPTCS.318.14.
- [11] 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.
- [12] Andrew Lucas (2014): *Ising formulations of many NP problems*. *Frontiers in Physics* 2, doi:10.3389/fphy.2014.00005.
- [13] Wilhelm Magnus (1954): *On the exponential solution of differential equations for a linear operator*. *Communications on Pure and Applied Mathematics* 7(4), pp. 649–673, doi:10.1002/cpa.3160070404.
- [14] Beatrice Nash, Vlad Gheorghiu & Michele Mosca (2020): *Quantum circuit optimizations for NISQ architectures*. *Quantum Science and Technology* 5(2), p. 025010, doi:10.1088/2058-9565/ab79b1.
- [15] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2019): *pytket library*. Available at <https://github.com/CQCL/pytket>.

- [16] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *t|ket*: a retargetable compiler for NISQ devices. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92.
- [17] The IBM Qiskit Team & Contributors (2021): *Qiskit: An Open-source Framework for Quantum Computing*, doi:10.5281/zenodo.2573505.
- [18] Richie Yeung (2020): *Diagrammatic Design and Study of Ansätze for Quantum Machine Learning*. doi:10.48550/arXiv.2011.11073.

Finite-dimensional Quantum Observables are the Special Symmetric \dagger -Frobenius Algebras of CP Maps

Stefano Gogioso

Hashberg Ltd, London, UK

Quantum Group, University of Oxford, UK

stefano.gogioso(at)cs.ox.ac.uk

We use *purity*, a principle borrowed from the foundations of quantum information, to show that all special symmetric \dagger -Frobenius algebras in CPM(fHilb)—and, in particular, all classical structures—are *canonical*, i.e. that they arise by doubling of special symmetric \dagger -Frobenius algebras in fHilb.

1 Introduction

The exact correspondence [9] between finite-dimensional C*-algebras and special symmetric \dagger -Frobenius algebras (\dagger -SSFAs) in fHilb—the dagger compact category of finite-dimensional complex Hilbert spaces and linear maps—is a cornerstone result in the categorical treatment of quantum theory [1, 4, 7]. A corresponding characterisation in CPM(fHilb)—the dagger compact category of finite-dimensional Hilbert spaces and completely positive maps [8]—has been an open question for around 10 years [5, 6]—of interest, for example, in the investigation of the robustness of sequentialisable quantum protocols [2, 6].

In this work we use *purity*, a principle borrowed from the foundations of quantum information [3], to answer this question once and for all: the \dagger -SSFAs in CPM(fHilb) are exactly the *canonical* ones, the ones arising by doubling of \dagger -SSFAs in fHilb. This is a notable result in that it allows fundamental notions from quantum theory—notably, those of quantum observable and measurement—to be defined directly in the diagrammatic language of CP maps, without reference to fHilb or the CPM construction, and without relying on the biproduct or convex-linear structure of CPM(fHilb).

2 Purity

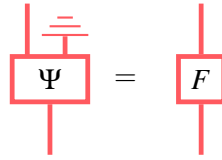
The very definition of morphisms in the category CPM(fHilb) means that a CP map $\Phi : \mathcal{H} \rightarrow \mathcal{K}$ can always be *purified*, i.e. that it can be written in terms of a *pure* map $\Psi : \mathcal{H} \rightarrow \mathcal{K} \otimes \mathcal{E}$ and discarding of an “environment” system \mathcal{E} :

$$\Phi \quad := \quad \begin{array}{c} \text{---} \\ | \\ \boxed{\Psi} \\ | \\ \text{---} \end{array}$$

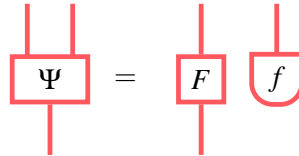
In this work, by a *pure* CP map $\Psi : \mathcal{H} \rightarrow \mathcal{K}$ we mean a CP map in the form $\Psi = \text{CPM}(\psi)$, arising by doubling of a morphism $\psi : \mathcal{H} \rightarrow \mathcal{K}$ in fHilb. This “diagrammatic” notion of purity is connected to the notion of purity used in the foundations of quantum information by the following *purity principle*.

Proposition 1 (Purity Principle).

If the following holds for some pure CP maps $\Psi : \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{E}$ and $F : \mathcal{H} \rightarrow \mathcal{H}$:



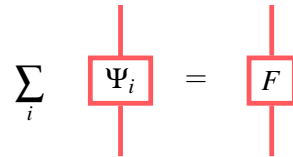
then there is a normalised pure state f on \mathcal{E} such that:



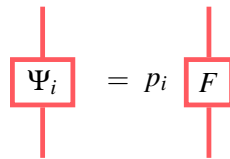
By expanding the discarding map in terms of some orthonormal basis of pure states, one straightforwardly gets an equivalent formulation of the principle in terms of sums.

Proposition 2 (Purity Principle, sums version).

If the following holds for pure CP maps $(\Psi_i : \mathcal{H} \rightarrow \mathcal{H})_i$ and $F : \mathcal{H} \rightarrow \mathcal{H}$:



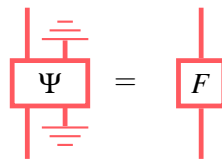
then there are coefficients $p_i \in \mathbb{R}^+$ summing to 1 such that the following holds for all i :



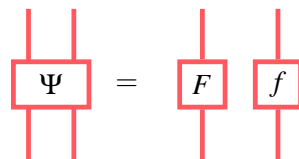
Because CPM (fHilb) is dagger compact, finally, one can also straightforwardly obtain a more general formulation of the principle which replaces the purifying state f with a purifying operator f .

Proposition 3 (Purity Principle, operator version).

If the following holds for some pure CP maps $\Psi : \mathcal{H} \otimes \mathcal{F} \rightarrow \mathcal{H} \otimes \mathcal{E}$ and $F : \mathcal{H} \rightarrow \mathcal{H}$:



then there is a pure CP map $f : \mathcal{F} \rightarrow \mathcal{E}$ such that:



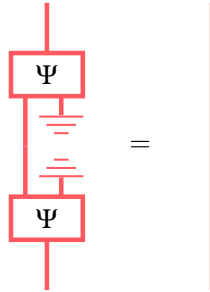
3 Isometries of CP maps

Theorem 4 (Purification of CP isometries).

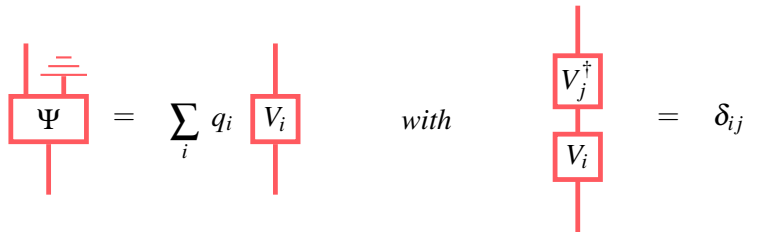
Every isometry Φ in CPM(fHilb) can be written as a \mathbb{R}^+ -linear combination of pure isometries V_i with pairwise orthogonal images.

$$\Phi^\dagger \circ \Phi = \mathbb{1} \quad \Rightarrow \quad \Phi = \sum_i q_i V_i \quad \text{with} \quad V_i^\dagger \circ V_j = \delta_{ij} \mathbb{1}$$

In pictures, this means that if the following equation holds, where Ψ is any purification of Φ :

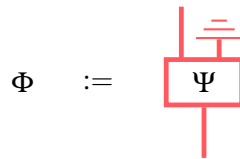


then we must in fact have:

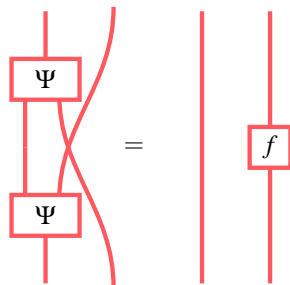


Furthermore, the coefficients satisfy $\sum_i (q_i)^2 = 1$, and they can be chosen to be all non-zero.

Proof. By the existence of purifications, we can obtain $\Phi : \mathcal{H} \rightarrow \mathcal{H}$ by discarding an “environment” system \mathcal{E} from some pure $\Psi : \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{E}$:



By the operator version of the purity principle, the isometry equation $\Phi^\dagger \circ \Phi = \mathbb{1}$ for Φ implies the following equation for Ψ , where f is some pure CP map $\mathcal{E} \rightarrow \mathcal{E}$:



The pure CP map f is self-adjoint, which means that we can find an orthonormal basis of pure states $(\varphi_i)_{i=1}^{\dim \mathcal{E}}$ and associated non-negative real numbers $(q_i)_i$ such that:

$$\begin{array}{c} \varphi_j^\dagger \\ \boxed{f} \\ \varphi_i \end{array} = \delta_{ij} q_i^2$$

for all $i, j = 1, \dots, \dim \mathcal{E}$. For each i with $q_i \neq 0$, define:

$$\boxed{V_i} := \frac{1}{q_i} \begin{array}{c} \varphi_i^\dagger \\ \boxed{\Psi} \end{array}$$

This way we get:

$$\Phi = \begin{array}{c} \equiv \\ \boxed{\Psi} \end{array} = \sum_i \begin{array}{c} \varphi_i^\dagger \\ \boxed{\Psi} \end{array} = \sum_{i \text{ s.t. } q_i \neq 0} q_i \boxed{V_i}$$

Furthermore, the V_i pure maps we just defined are isometries with orthogonal images:

$$\begin{array}{c} \boxed{V_j^\dagger} \\ \boxed{V_i} \end{array} = \frac{1}{q_i q_j} \begin{array}{c} \varphi_j^\dagger \\ \boxed{\Psi} \\ \boxed{\Psi} \\ \varphi_i \end{array} = \frac{1}{q_i q_j} \begin{array}{c} \varphi_j^\dagger \\ \boxed{f} \\ \varphi_i \end{array} = \delta_{ij} \frac{q_i^2}{q_i q_j} = \delta_{ij}$$

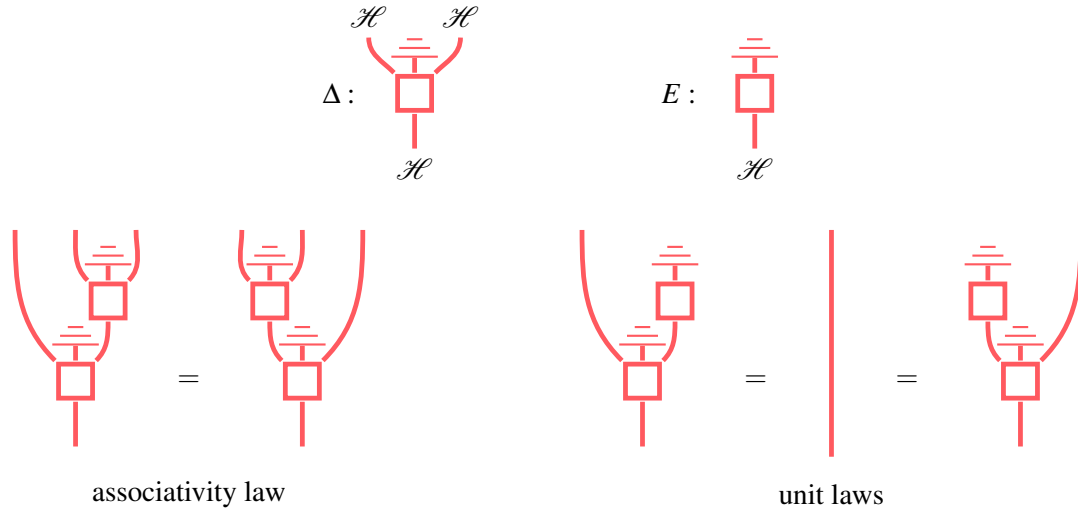
Finally, we have that:

$$\begin{array}{c} | \\ \boxed{\Psi} \\ \equiv \\ \boxed{\Psi} \\ | \end{array} = \sum_{ij} q_i q_j \begin{array}{c} \boxed{V_j^\dagger} \\ \boxed{V_i} \end{array} = \sum_{ij} \delta_{ij} q_i q_j = \sum_i q_i^2$$

so we conclude that $\sum_i (q_i)^2 = 1$. □

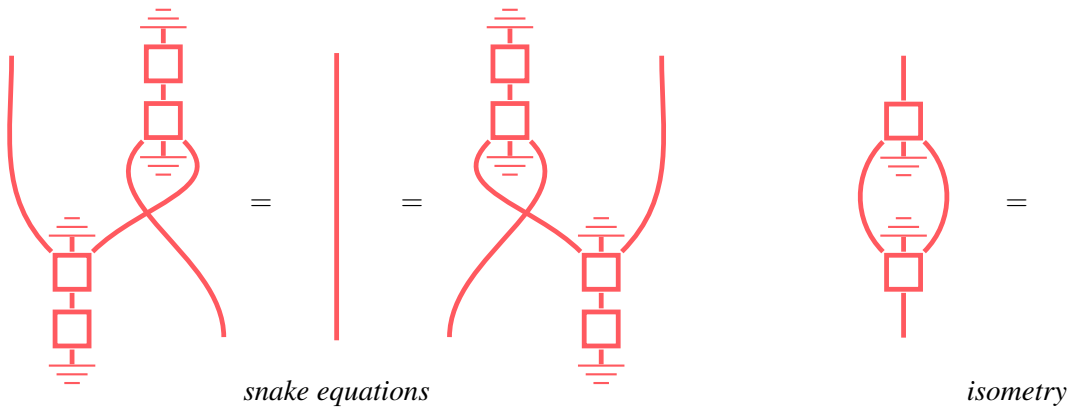
4 Main result

A comonoid (Δ, E) on an object \mathcal{H} of the dagger compact category $\text{CPM}(\text{fHilb})$ is a pair of completely positive maps satisfying the associativity and unit laws:



Theorem 5 (Ophidian isometric CP comonoids are pure).

If (Δ, E) is a comonoid in $\text{CPM}(\text{fHilb})$ which is isometric and satisfies the snake equations, then the CP maps Δ and E are both pure.¹



Proof. By Theorem 4 and the isometry condition, we know that:

$$\begin{array}{c} \text{---} \\ \text{---} \\ \square \\ \text{---} \end{array} = \sum_{i=1}^n q_i \begin{array}{c} \text{---} \\ \text{---} \\ \square \\ \text{---} \\ V_i \end{array}$$

for some pure isometries V_i of fHilb with $V_i^\dagger \circ V_j = \delta_{ij} \mathbb{1}$. The coefficients q_i satisfy $\underline{q} \cdot \underline{q} = \sum_i (q_i)^2 = 1$, and we choose them to be all non-zero. By the existence of purifications, we decompose E as a sum of

¹Associativity is not actually necessary for this result to hold.

non-zero pure effects:

$$\overline{\square} = \sum_k \overline{E_k}$$

By the purity principle and the unit laws for the comonoid, we deduce that there are coefficients $r_{i,k}, l_{i,k} \in \mathbb{R}^+$ such that:

$$\begin{array}{c} \overline{E_k} \\ \downarrow \\ \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = l_{i,k} \quad \begin{array}{c} \overline{E_k} \\ \downarrow \\ \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = r_{i,k}$$

Writing $l_i := \sum_k l_{i,k}$ and $r_i := \sum_k r_{i,k}$, the *unit laws* imply that $\underline{q} \cdot \underline{r} = \sum_i q_i r_i = 1 = \sum_i q_i l_i = \underline{q} \cdot \underline{l}$: it cannot therefore be that for all i, k we have $l_{i,k} = 0$ or that for all i, k we have $r_{i,k} = 0$. Picking some i, k such that $l_{i,k} \neq 0$, we deduce that E is, in fact, a non-zero pure effect:

$$l_{i,k} \overline{\square} = \begin{array}{c} \overline{E_k} \\ \downarrow \\ \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = r_i \overline{E_k}$$

Because E is a pure effect, we will henceforth drop the discarding map in our notation:

$$E : \square$$

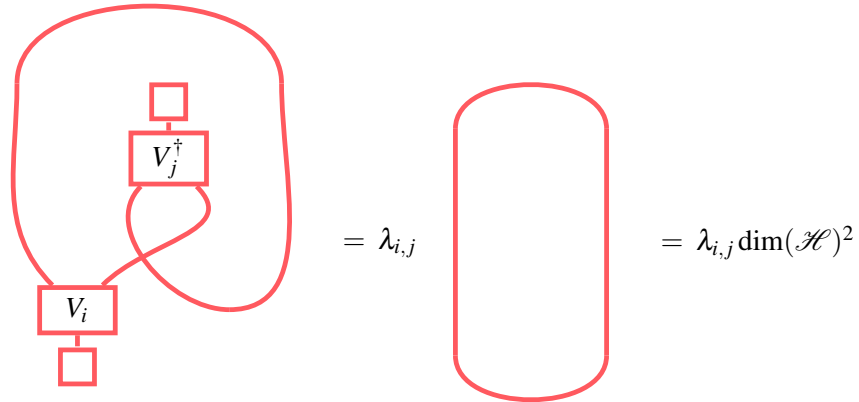
The same trick can then be used to show that $r_i = l_i$ for all i :

$$l_i \square = \begin{array}{c} \square \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = r_i \square$$

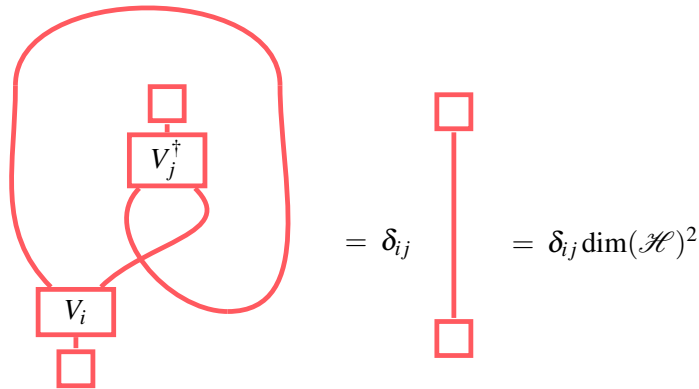
Having established that the counit is pure, we now move on to establish that the comultiplication is pure as well. From the snake equations, the purity principle implies the existence of coefficients $\lambda_{i,j}, \rho_{i,j} \in \mathbb{R}^+$ such that:

$$\begin{array}{c} \square \\ \downarrow \\ V_j^\dagger \\ \downarrow \\ \text{---} \end{array} \begin{array}{c} \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = \lambda_{i,j} \quad \begin{array}{c} \square \\ \downarrow \\ V_j^\dagger \\ \downarrow \\ \text{---} \end{array} \begin{array}{c} \square \\ \downarrow \\ V_i \\ \downarrow \\ \text{---} \end{array} = \rho_{i,j}$$

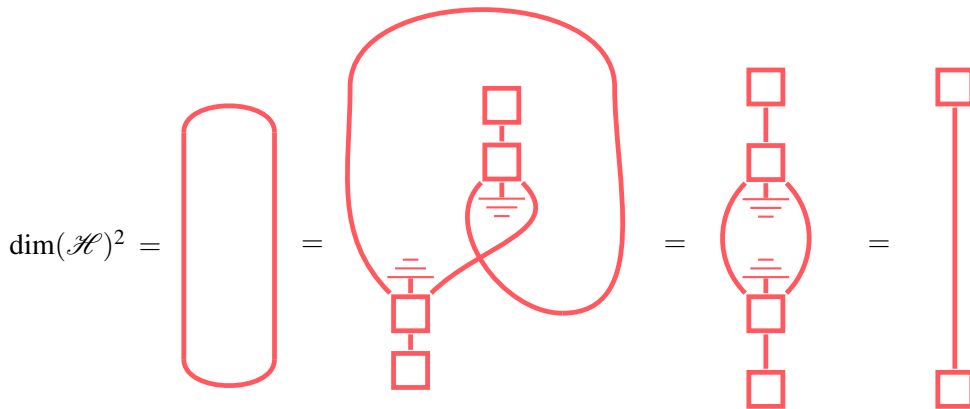
We now proceed to show that $\lambda_{i,j} = \delta_{ij}$, using the left snake equation. The proof that $\rho_{i,j} = \delta_{ij}$ is analogous, using the right snake equation. Taking the trace on both sides of the snake equation, we obtain the following equation, where \mathcal{H} is the underlying Hilbert space for the comonoid (we get $\dim(\mathcal{H})^2$ because we are working in the CPM category, where the scalar is doubled):



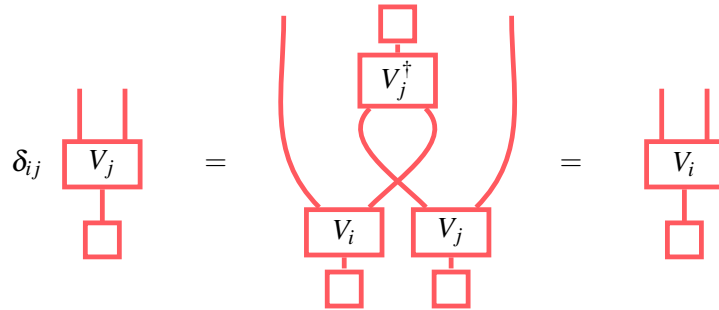
Taking the trace on the LHS of the snake equation and using the fact that $V_j^\dagger V_i = \delta_{ij} \mathbb{1}$, we get a different equation:



The rightmost equality follows from isometry and the snake equation:

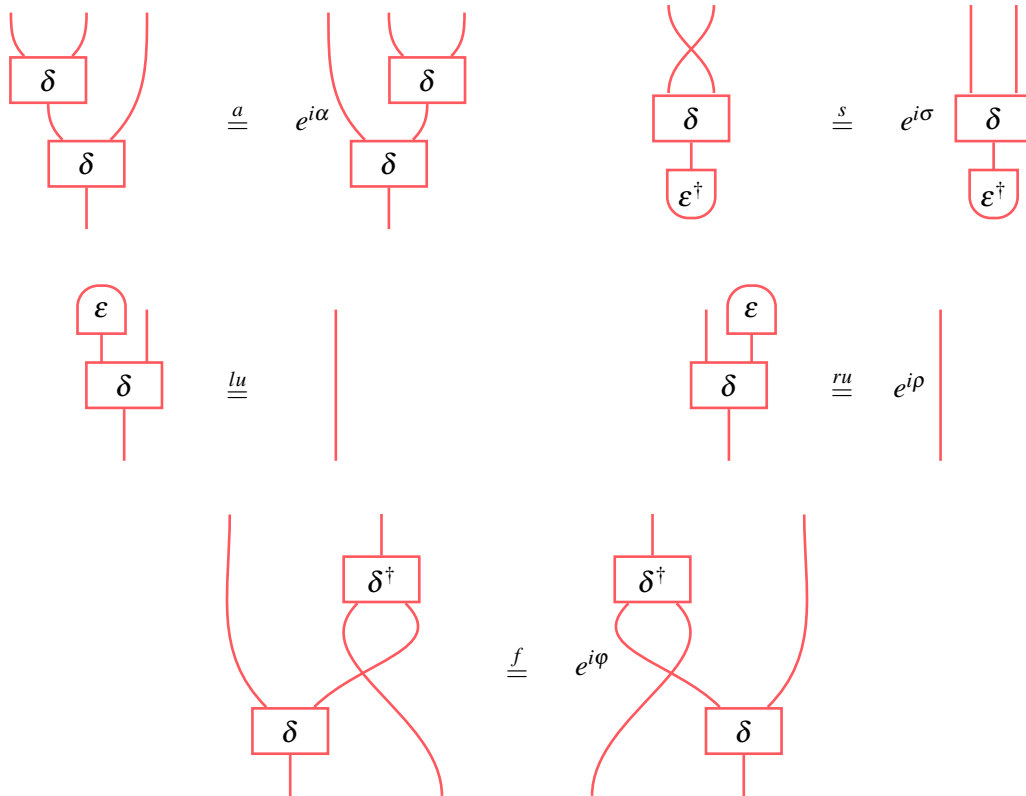


Having established that $\lambda_{i,j} = \delta_{ij} = \rho_{i,j}$, we now show that $n \geq 2$ leads to a contradiction unless $\dim(\mathcal{H}) = 0$ (in which case the statement of this theorem is trivial). Indeed, we have the following equation for all i, j :



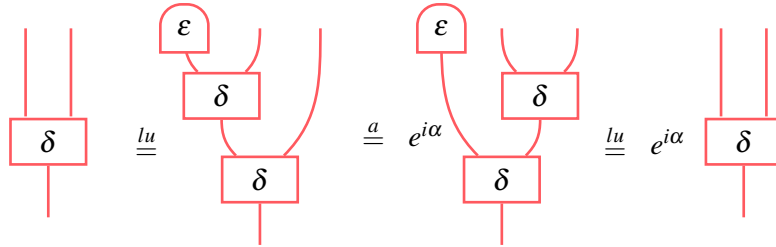
If we have $n \geq 2$, then we can set $i \neq j$ in the equation above, concluding that the RHS state is the zero state. However, we also know that V_i is an isometry and that the adjoint of the counit is a state of norm $\dim(\mathcal{H})^2$, hence the RHS state also has norm $\dim(\mathcal{H})^2$. So either $n = 1$, in which case the comultiplication is pure, or $n \geq 2$ and $\dim(\mathcal{H}) = 0$, in which case the comultiplication is the zero map, which is also pure. \square

Lemma 6. Let $\delta : \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{H}$ and $\varepsilon : \mathcal{H} \rightarrow \mathbb{C}$ be morphisms in fHilb which satisfy the associativity law up to a phase $e^{i\alpha}$, the symmetry law up to a phase $e^{i\sigma}$, the left unit law exactly, the right unit law up to a phase $e^{i\rho}$ and the Frobenius law up to a phase $e^{i\varphi}$:

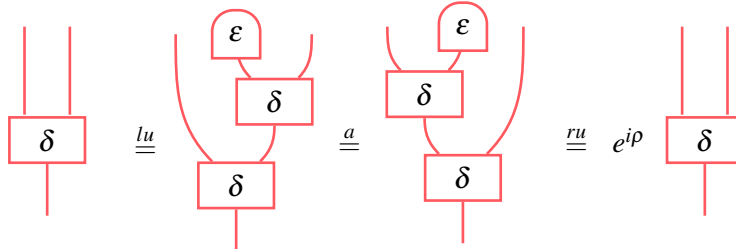


Then $(\delta, \varepsilon, \delta^\dagger, \varepsilon^\dagger)$ is a symmetric \dagger -Frobenius algebra in fHilb, i.e. $\alpha = \rho = \sigma = \varphi = 0 \pmod{2\pi}$.

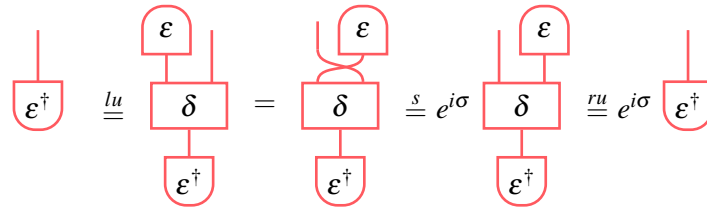
Proof. The associativity law and left unit law prove that $\alpha = 0 \pmod{2\pi}$:



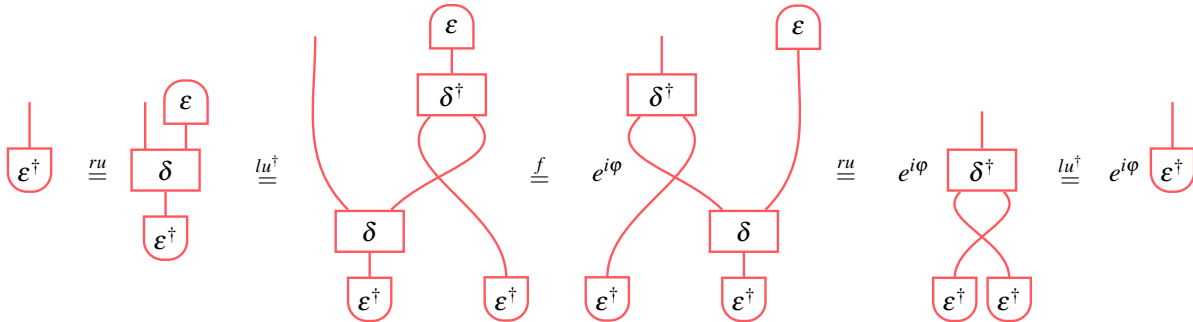
Then, the associativity law and the unit laws prove that $\rho = 0 \pmod{2\pi}$:



Then, the symmetry law and the unit laws prove that $\sigma = 0 \pmod{2\pi}$:



Finally, the Frobenius law and the right unit law prove that $\varphi = 0 \pmod{2\pi}$:



This concludes our proof. □

Corollary 7 (CP \dagger -SSFAs are all canonical).

The \dagger -SSFAs in CPM (fHilb) are all canonical, i.e. they all arise by doubling of \dagger -SSFAs in fHilb.

Proof. Now let $(\Delta, E, \Delta^\dagger, E^\dagger)$ be a \dagger -SSFA in CPM (fHilb). Because (Δ, E) is a comonoid which is isometric and satisfies the snake equations, we now know that Δ and E are pure, i.e. that we can find linear

maps δ and ε in \mathbf{fHilb} such that $\Delta = \text{CPM}(\delta)$ and $E = \text{CPM}(\varepsilon)$ arise by doubling. We now wish to conclude that $(\delta, \varepsilon, \delta^\dagger, \varepsilon^\dagger)$ form a \dagger -SSFA in \mathbf{fHilb} , but this doesn't immediately follow from the equations in $\text{CPM}(\mathbf{fHilb})$: while δ is certainly an isometry in \mathbf{fHilb} , the associativity law, unit laws, symmetry law and Frobenius law for δ and ε are only guaranteed to hold up to phase. In fact, $(\delta, \varepsilon, \delta^\dagger, \varepsilon^\dagger)$ is not, in general, a \dagger -SSFA in \mathbf{fHilb} . However, it is easy to show (cf. Lemma 6 below) that $(\delta, e^{-i\lambda}\varepsilon, \delta^\dagger, e^{i\lambda}\varepsilon^\dagger)$ is always a \dagger -SSFA in \mathbf{fHilb} , where $e^{i\lambda}$ is the phase associated to the identity in the left unit law. Because $\text{CPM}(e^{-i\lambda}\varepsilon) = \text{CPM}(\varepsilon)$, this is enough to prove our result. \square

Acknowledgements

The author would like to thank Chris Heunen [5, 6] and Sergio Boixo [6] for formulating the original question. The author would also like to thank an anonymous QPL 2022 reviewer for thoroughly checking the correctness of the proofs, as well as suggesting several improvements to the presentation of this work.

References

- [1] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Logic in computer science, 2004. Proceedings of the 19th Annual IEEE Symposium on*, pages 415–425. IEEE, 2004. doi:10.1109/LICS.2004.1319636.
- [2] Sergio Boixo and Chris Heunen. Entangled and sequential quantum protocols with dephasing. *Physical Review Letters*, 108(12):120402, 2012. doi:10.1103/PhysRevLett.108.120402.
- [3] Giulio Chiribella, Giacomo Mauro D’Ariano, and Paolo Perinotti. Probabilistic theories with purification. *Physical Review A*, 81(6):062348, 2010. doi:10.1103/PhysRevA.81.062348.
- [4] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes*. Cambridge University Press, 2017. doi:10.1017/9781316219317.
- [5] Chris Heunen. When is this map completely positive?, 2011. URL <https://mathoverflow.net/questions/64407/when-is-this-map-completely-positive>.
- [6] Chris Heunen and Sergio Boixo. Completely positive classical structures and sequentializable quantum protocols. *Electronic Proceedings in Theoretical Computer Science*, 95:91–101, 2012. doi:10.4204/EPTCS.95.9.
- [7] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory: an introduction..* Oxford University Press, 2019. doi:10.1093/oso/9780198739623.001.0001.
- [8] Peter Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical computer science*, 170:139–163, 2007. doi:10.1016/j.entcs.2006.12.018.
- [9] Jamie Vicary. Categorical formulation of finite-dimensional quantum algebras. *Communications in Mathematical Physics*, 304(3):765–796, 2011. doi:10.1007/s00220-010-1138-0.