Pauli Fusion: a computational model to realise quantum transformations from ZX terms

Niel de Beaudrap
University of Oxford
Oxford, UK
niel.debeaudrap@cs.ox.ac.uk

Ross Duncan
University of Strathclyde
Glasgow, UK
Cambridge Quantum Computing Ltd.,
Cambridge, UK
ross.duncan@strath.ac.uk

Dominic Horsman
Université Grenoble Alpes
Grenoble, France
dom.horsman@gmail.com

Simon Perdrix
Université de Lorraine
CNRS, Inria, LORIA
Nancy, France
simon.perdrix@loria.fr

The ZX calculus is an abstract mathematical tool to represent — and importantly to calculate with — tensors of a sort that are common in quantum computational theory. From the outset, the ZX calculus was developed to be universal, and more recently, it has been extended to be made complete. These features make the ZX calculus attractive as a potential component of an optimising compiler for quantum computers — provided that one can determine whether and how a transformation denoted by a ZX term can be deterministically realised. Until recently, it was not known how to do so, unless the ZX term was in a certain sense "circuit-like". We present an abstract model of quantum computation, the *Pauli Fusion* model, whose primitive operations correspond closely to generators of the ZX calculus (and are also straightforward abstractions of basic operations in some leading proposed quantum technologies). These operations have non-deterministic heralded effects, similarly to measurement-based quantum computation. We describe sufficient conditions for Pauli Fusion procedures to be deterministically realisable, so that it performs a given transformation independently of its non-deterministic outcomes. This provides an operational model to realise ZX terms beyond the circuit model.

1 Introduction

Physically existing quantum computers have now become a reality (although not yet fault tolerant) [12, 10]. These computers need software, for algorithm design, verification, and compilation. Previously, quantum protocols have been represented largely using circuit notation or, less frequently, the measurement calculus. However, with actual quantum devices to now compare theory with, the shortcomings of these models become clear. The circuit gate model in particular turns out not to model basic technological operations, is inflexible (circuits cannot easily be re-written to equivalent ones), and computationally hard to verify. Especially in near-term noisy intermediate scale quantum (NISQ) devices, these properties give bloated software with only basic optimisation tools, which cannot be verified at scales of more than a few tens of qubits.

Recent work has placed a different way of representing quantum processes at the forefront of optimisation, verification, and design for NISQ devices and beyond. The ZX calculus [3] is a formal diagrammatic language equipped with equational re-write rule sets that are complete for

Submitted to:
QPL 2019

Clifford [1], Clifford+T [13], and full [17, 18] pure-state qubit quantum mechanics. The ZX calculus has led to optimisation strategies that out-perform all others in T-count reduction [15] (an important metric for fault-tolerant computing) and gate compilation [6, 14]. The generators of the calculus directly map the basic operations of lattice surgery in the surface code (notoriously complicated operations to describe using the circuit model), and ZX has been used to verify and find novel error correction procedures [7, 5, 9]. It comes with a scalable notation capable of representing repeated structures at arbitrary qubit scales [2]. The calculus also acts in the crucial role of an intermediate representation in a new commercial quantum compiler [4].

With the success of the ZX calculus as a design, verification, and optimisation tool the question now remains: what is the *computational model* that the ZX calculus works best with? A computational model in the sense we use it here is a set of primitive operations and their composition, out of which algorithms and protocols can be formed. They form the basis of the denotational meaning given to composite procedures, and demonstrate the fundamental information processing capabilities at the designer's disposal. If we are to get the best out of using the ZX calculus as an intermediate representation, then it is most efficient to use a computational model that reflects, and is reflected by, the basic structure of the calculus. The model will be used both at the top of the compiler stack, to conceptualise, design, and verify protocols, and also at the bottom to extract operational procedures from a ZX calculus diagram. Previously, this 'operational extraction' was known only as 'circuit extraction' (e.g. as in [14]). The use of circuit-model gates at extraction is extremely wasteful, especially if is then turned into procedures such as lattice surgery that very closely model ZX generators. The key then, is to produce a computational model that works efficiently and conceptually clearly with the ZX calculus.

In this paper we introduce the *Pauli Fusion* model of quantum computing. The model uses the fundamental operations of the merging and splitting of logical (Pauli) operators. Complementarity, not unitarity, is the guiding principle, and Pauli Fusion (PF) procedures are in general non-deterministic. These fundamental elements very closely model those found in lattice surgery [11], and in operations of optical fusion gates that have similar effects [16] The PF model takes these operations as its primitives, from which (if desired) elements in a circuit or MBQC model could be constructed. We show how the PF model has a direct representation in terms of generators of annotated ZX diagrams, which we term PF diagrams. As the diagrams can include indeterminism, we give a definition and a procedure for finding the *PF flow* of a ZX diagram, meaning it can be deterministically implemented by a series of (in themselves probabilistic) PF procedures. Therefore, by thus 'splitting the atom' of well known logic gates into more fundamental, compositional, operations, we give a general solution to the problem of extracting operational meaning from a ZX diagram, and a new foundational computational model corresponding directly to the ZX calculus.

2 The Pauli Fusion model

We define the elements of the Pauli fusion model of computation in terms of CPTP maps, not all of which are unitary. These CPTP maps are our primitive *Pauli Fusion (PF) operations*, described in terms of their Kraus operators, and may also be classically controlled. Intuitively, the basic op-

erations can be viewed as the splitting and merging of Pauli logical operators for qubits, and the biproducts produced when two logical qubits merge into one (readers familiar with the procedure of lattice surgery in surface codes will find this a straightforward abstraction). PF operations also have a representation by *Pauli Fusion (PF) diagrams*, which we introduce here. PF diagrams are annotated ZX (AZX) diagrams where the annotations include probabilistic elements. The requirement that a PF diagram correspond to a set of PF operations imposes a restriction which we define as *runnability*. We show that this requirement is equivalent to the PF diagram having a *time ordering* of its elements. This adds another layer of structure (essentially directed edges) to PF diagrams that we will see in the next section is crucial to understanding when a (plain) ZX diagram can deterministically be implemented by a set of PF operations.

2.1 Pauli Fusion operations

We define the following linear transformations on one- or two-qubit state vectors:

$$A_{V,0} = \langle +| \tag{1a}$$
$$A_{V,1} = \langle -| \tag{1b}$$
$$K_{V,0} = |+\rangle\langle ++| + |-\rangle\langle --| \tag{1c}$$
$$K_{V,1} = |+\rangle\langle +-| + |-\rangle\langle -+| \tag{1d}$$
$$R_V = \exp\left(-\tfrac{1}{2}iZ\right) \tag{1e}$$
$$A_{H,0} = \langle 0| \tag{1f}$$
$$A_{H,1} = \langle 1| \tag{1g}$$
$$K_{H,0} = |0\rangle\langle 00| + |1\rangle\langle 11| \tag{1h}$$
$$K_{H,1} = |0\rangle\langle 01| + |1\rangle\langle 10| \tag{1i}$$
$$R_H = \exp\left(-\tfrac{1}{2}iX\right) \tag{1j}$$

We conceive of $A_{V,s}$ and $A_{H,s}$ as "annihilators" mapping one qubit to zero; we use these as Kraus maps of destructive measurement operations in the X or Z eigenbasis respectively, with the index s representing the outcome. (The maps $A^\dagger_{V,s}$ and $A^\dagger_{H,s}$ are then preparation maps of X or Z eigenstates.) The maps $K_{V,s}$ and $K_{H,s}$ are maps from two-qubit states to single qubit states, projecting onto the $(-1)^s$ eigenstates of $X\otimes X$ or $Z\otimes Z$, and producing a single qubit which is an eigenstate of X or Z. (In the case of $K_{V,1}$ and $K_{H,1}$, this involves breaking the symmetry between the two qubits, in a way which is arbitrary but ultimately unimportant.) The operations R_V and R_H are single-qubit Pauli Z and X rotations by one radian: exponentiating by a real-valued angle α in radians yields a single-qubit $R_z(\alpha)$ or $R_x(\alpha)$ rotation respectively.

We use these linear maps on state-vectors, together with the single-qubit Hadamard gate H and the two-qubit SWAP gate, to define the *(elementary) PF operations* as the following CPTP maps, described here as acting on states ρ variously on one or two qubits:

$$\mathsf{Had}(\rho) = H\rho H^\dagger \tag{2a}$$
$$\mathsf{VInit}(\rho) = \left(A^\dagger_{V,0}\otimes\mathbb{1}\right)\rho\left(A_{V,0}\otimes\mathbb{1}\right) \tag{2b}$$
$$\mathsf{VProj}(\rho) = \sum_{s\in\{0,1\}} A_{V,s}\rho A^\dagger_{V,s} \tag{2c}$$
$$\mathsf{VSplit}(\rho) = K^\dagger_{V,0}\rho K_{V,0} \tag{2d}$$
$$\mathsf{VMerge}(\rho) = \sum_{s\in\{0,1\}} K_{V,s}\rho K^\dagger_{V,s} \tag{2e}$$
$$\mathsf{VRot}^{\alpha,S,T}(\rho) = R_V^{\Theta(\alpha,S,T)}\rho R_V^{-\Theta(\alpha,S,T)} \tag{2f}$$
$$\sigma(\rho) = (\mathrm{SWAP})\rho(\mathrm{SWAP})^\dagger \tag{2g}$$
$$\mathsf{HInit}(\rho) = \left(A^\dagger_{H,0}\otimes\mathbb{1}\right)\rho\left(A_{H,0}\otimes\mathbb{1}\right) \tag{2h}$$

$$\mathsf{HProj}(\rho) \;=\; \sum_{s\in\{0,1\}} A_{H,s}\,\rho\,A_{H,s}^{\dagger} \qquad (2\mathrm{i}) \qquad \mathsf{HMerge}(\rho) \;=\; \sum_{s\in\{0,1\}} K_{H,s}\,\rho\,K_{H,s}^{\dagger} \qquad (2\mathrm{k})$$

$$\mathsf{HSplit}(\rho) \;=\; K_{H,0}^{\dagger}\,\rho\,K_{H,0} \qquad (2\mathrm{j}) \qquad \mathsf{HRot}^{\alpha,S,T}(\rho) \;=\; R_H^{\Theta(\alpha,S,T)}\,\rho\,R_H^{-\Theta(\alpha,S,T)} \qquad (2\ell)$$

Note that the maps VProj, HProj, VMerge, and HMerge all are non-unitary. The first two actually are measurement operations, which we suppose also produce a classical bit as a side-effect, representing the measurement outcome (the bit s which indexes the Kraus operator). We also suppose that VMerge and HMerge produce a classical bit s as a side-effect, indicating which of the two Kraus operators were realised, according to a distribution which is determined by the square of the Euclidean norm of the state $K_{V,s_u}|\psi\rangle$, $K_{H,s_u}|\psi\rangle$, $A_{V,s_u}|\psi\rangle$, or $A_{H,s_u}|\psi\rangle$ which would result from application of one of the Kraus operators to an input state $|\psi\rangle$, for $s_u \in \{0,1\}$.

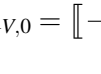
We present VInit and HInit as maps on a system, but their effect is to prepare fresh qubits in the $|+\rangle$ or $|0\rangle$ state. The maps VSplit and HSplit realise unitary embeddings of one qubit into two. The operations $\mathsf{VRot}^{\alpha,S,T}$ and $\mathsf{HRot}^{\alpha,S,T}$ depend on sets S and T of labels, which indicate classical bits s_x for $x \in S$ or $x \in T$ which may affect the angle of rotation (some of which may be the outcomes of the maps above), according to the function

$$\Theta(\alpha,S,T) \;=\; \left[\prod_{v\in S}(-1)^{s_v}\right]\alpha + \sum_{w\in T} s_w\,\pi\,. \qquad (3)$$

The operations of Eqn. (2) may be performed in tensor product, and composed in any way which is well-typed. For the operations $\mathsf{VRot}^{\alpha,S,T}$ and $\mathsf{HRot}^{\alpha,S,T}$ which may depend on classical bits, we also require that the value of the bit is determined (as an input, through a probability distribution, or through an operation which determines its value) at the time the map is performed.

2.2 Pauli Fusion diagrams

We may readily observe that the Kraus operations defined in Eqns. (2.1) have straightforward representations in the ZX calculus,

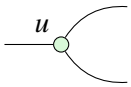
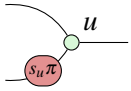
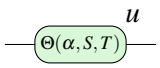
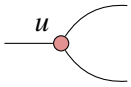
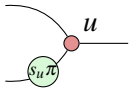
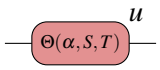
$A_{V,0} = [\![$  $]\!]$; $A_{V,1} = [\![$ $]\!]$; $K_{V,0} = [\![$  $]\!]$; $K_{V,1} = [\![$  $]\!]$; $R_V^{\alpha} = [\![$  $]\!]$;

$A_{H,0} = [\![$ $]\!]$; $A_{H,1} = [\![$  $]\!]$; $K_{H,0} = [\![$  $]\!]$; $K_{H,1} = [\![$  $]\!]$; $R_H^{\alpha} = [\![$  $]\!]$, (4)

where $[\![\cdot]\!]$ is the standard interpretation of ZX diagrams (which in this article are read from left to right). Note that these maps, together with their adjoints, generate the ZX calculus. Considering ZX as a potential intermediate language for quantum compilers, this close relation between the Kraus operators of Pauli Fusion and the generators of ZX provides a tantalising prospect, of using the PF model to directly represent ZX diagrams.

To pursue this line of investigation, we consider how we might use the ZX calculus to represent linear superoperators, whose Kraus operators can be obtained (more more precisely, denoted) by composing the diagrams of Eqn. (4). We may then consider when such diagrams represent an operation which can be realised by a Pauli Fusion procedure.

Definition 1. *A* Pauli Fusion diagram *(or PF-diagram) is an AZX diagram, with labelled vertices $V(D)$ and directed edges $E(D)$ which can be generated from the set of generators below. (We label these diagrams with the names of Pauli Fusion operations,* e.g.*, "VMerge_u", to indicate the operation for which the node u is intended to stand.) This diagram is accompanied by a set $\mathscr{B}$ of labels of bits, $s_u \in \{0,1\}$ for $u \in \mathscr{B}$, which are involved in the annotations (*e.g.*, the sets S and T in $\mathsf{VRot}^{\alpha,S,T}$ and $\mathsf{HRot}^{\alpha,S,T}$.*

VSplit_u 	VMerge_u 	$\mathsf{VRot}_u^{\alpha,S,T}$ 
HSplit_u 	HMerge_u 	$\mathsf{HRot}_u^{\alpha,S,T}$ 
VInit_u 	HInit_u 	Had_u 
VProj_u 	HProj_u 	σ 

Remark. In the case of $\mathsf{VRot}^{\alpha,S,T}$ and $\mathsf{HRot}^{\alpha,S,T}$, we may write an explicit formula for the angle in place of $\Theta(\alpha,S,T)$ when that formula is simple enough: for instance, we may substitute the expression "$\Theta(0,\emptyset,\{u\})$" with "$s_u\pi$".

Following [8], we define the semantics of the Pauli-Fusion diagrams relying on the semantics of the ZX-calculus.

Definition 2 (Denotational semantics of Pauli Fusion diagrams)**.** *Given a PF-diagram D with a set $\mathscr{B}$ of index-labels for classical bits $s \in \{0,1\}^{\mathscr{B}}$:*

- *For a given $x \in \{0,1\}^{\mathscr{B}}$, $D(x)$ denotes the ZX-diagram where $s_b \leftarrow x_b$ for each $b \in \mathscr{B}$. For a given $z \in \{0,1\}^{\mathscr{B}\setminus V(D)}$, let $D|_z$ be the Pauli Fusion diagram obtained by the partial assignment $s_b \leftarrow z_b$ for all $b \in \mathscr{B} \setminus V(D)$.*
- *If $\mathscr{B} \subseteq V(D)$, $[\![D]\!]^\natural$ denotes the superoperator $\rho \mapsto \sum_{s\in\{0,1\}^{\mathscr{B}}} [\![D(s)]\!]\,\rho\,[\![D(s)]\!]^\dagger$, where $[\![\cdot]\!]$ is the standard interpretation of the ZX-diagrams.*
- *For a probability distribution p on $\{0,1\}^{\mathscr{B}\setminus V(D)}$, let $[\![D]\!]^\natural_p$ denote the superoperator*

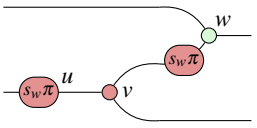
$$\rho \;\mapsto \sum_{\substack{s\in\{0,1\}^{\mathscr{B}\cap V(D)}\\ z\in\{0,1\}^{\mathscr{B}\setminus V(D)}}} p(z)\,[\![D|_z(s)]\!]\,\rho\,[\![D|_z(s)]\!]^\dagger \;. \tag{5}$$

As a special case, for a fixed string $r \in \{0,1\}^{\mathscr{B}\setminus V(D)}$, let $[\![D]\!]^\natural_r$ denote $[\![D]\!]^\natural_p$ for p the point-mass distribution on r.

We define $[\![D]\!]_p^\natural$ and $[\![D]\!]_r^\natural$ above in the case $\mathscr{B} \not\subseteq V(D)$ as we anticipate that this will be useful to describe procedures which are subject to noise or classical control. In much of what follows below, we suppose that $\mathscr{B} \subseteq V(D)$: when $\mathscr{B}$ is not taken to be a subset of $V(D)$ we shall clearly indicate that this is the case.

Definition 3. *For a given PF diagram D, a string $x \in \{0,1\}^{\mathscr{B}\cap V(D)}$ is called a* branch (*or* branch string) *of the PF diagram. If $\mathscr{B} \subseteq V(D)$, we call $[\![D(x)]\!]$ for such a string x a* branch map (*and in particular,* the branch map for *x*) *of D; if $\mathscr{B} \not\subseteq V(D)$, then for $r \in \{0,1\}^{\mathscr{B}\setminus V(D)}$, we call $[\![D|_r(x)]\!]$ a* branch map of *D* given *r* (*and in particular,* the branch map of *D* for *x* given *r*).

It will be useful to analyse PF procedures entirely in terms of PF diagrams. However, because of the simple way in which we have defined them, not all Pauli Fusion diagrams correspond to an actual Pauli Fusion procedure. For instance, consider the diagram

$D =$ 

: (6)

this is a well-formed PF diagram, and denotes a superoperator $\mathbf{D} = [\![D]\!]^\natural$ (which is also in fact a CPTP map), with two Kraus operators indexed by the single bit $s_w \in \{0,1\}$. However, the elements from which the PF diagram D was composed cannot be mapped straightforwardly to a PF procedure, as the bit s_w which is generated by the VMerge_w operation is used at the $\mathsf{HRot}_u^{s_w\pi}$ operation acting on one of its inputs. We wish to consider under what conditions a Pauli Fusion diagram corresponds, part by part, to a Pauli Fusion procedure.

Definition 4. *Let D be a Pauli-Fusion diagram with Kraus operators governed by bits s_u for $u \in \mathscr{B}$. A* time-ordering *of D is a function $t : V(D) \to \mathbb{N}$ such that, for all $u, v \in V(D)$,*

(*i*) *if there is a directed edge $u \to v$ in D, then $t(u) < t(v)$;*

(*ii*) *if the operation at v is either $\mathsf{VRot}_v^{\alpha,S,T}$ or $\mathsf{HRot}_v^{\alpha,S,T}$ operation for some $S, T \subseteq \mathscr{B}$, and $u \in S \cup T$, then $t(u) < t(v)$.*

If D has a time-ordering t, we say that D is runnable.

Lemma 1. *D is a runnable Pauli Fusion diagram if and only if it is the diagram of a Pauli Fusion procedure $\mathfrak{P}$.*

Proof (sketch). It is easy to show that the diagram of any Pauli Fusion procedure has a time-ordering in the above sense. Conversely, if D has a time-ordering, then for each $\tau \in \mathbb{N}$, recursively form the tensor product $\mathfrak{P}_\tau$ of all operations associated with nodes $v \in V(D)$ with $t(v) = \tau$, together with the identity operation on any qubits which are input wires of the diagram which have not yet been acted on, or qubits which have been produced by one operation but not acted on by another. Let $\mathfrak{P} = \mathfrak{P}_T \circ \cdots \circ \mathfrak{P}_1 \circ \mathfrak{P}_0$ for $T = \max_{v\in V} t(v)$: then D is the diagram of $\mathfrak{P}$. □

3 PF-diagram extraction

Considering ZX diagrams as an intermediate language, we wish to consider when such a diagram D can be operationally realised by a Pauli Fusion procedure — specifically, one which realises D *deterministically*, in the sense that all of the Kraus operators of the procedure are proportional to $[\![D]\!]$. To this end, we define a "flow" condition (analogous to the flow conditions of measurement-based quantum computation), which suffices for such a Pauli Fusion procedure to exist.

We use the following standard graph theoretic properties:

Definition 5. *In a graph G (possibly with self-loops) and a vertex-set $C \subseteq V(G)$, we write* $\mathrm{Odd}(C) \subseteq V(G)$ *for the set of vertices adjacent to an odd number of elements of C (where a vertex with a loop is counted as a neighbour to itself).*

Definition 6. *In a graph G and a partial order $\preccurlyeq$ on $V(G)$, let $\prec$ stand for the irreflexive relation $(a \preccurlyeq b) \,\&\, (a \neq b)$. For a vertex $u \in V(G)$, we then define the* future neighbourhood $N^+(u) \subset V(G)$ of u, *and the* past neighbourhood $N^-(u) \subset V(G)$ of u, *by*

$$N^+(u) := \big\{ v \in N(u) \;\big|\; u \prec v \big\}, \qquad N^-(u) := \big\{ v \in N(u) \;\big|\; v \prec u \big\}.$$

We further define the shorthand $\delta^{\pm}(u) := \big|N^{\pm}(u)\big|$.

3.1 Signatures of ZX diagrams

In the following, in order to maintain a close connection to PF diagrams, we suppose that ZX diagrams have distinct labels for each node, and that each open wire is explicitly indicated as either an *input* or an *output* wire. We refer to such ZX diagrams as *labelled* ZX diagrams.

Definition 7. *A (labelled) ZX-diagram is in a* graph-like form (*or is* graph-like) *if it is H-free, has no connections between spiders of the same colour, and has no parallel wires or loops on any single vertex.*

By rewriting all H nodes using the Euler decomposition, condensing all spiders, and removing all loops and (pairs of) parallel edges, it is easy to show that:

Lemma 2. *Any ZX-diagram can be transformed into a graph-like ZX-diagram.*

To a graph-like ZX diagram D, we associate a corresponding *signature* $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$, which will enable us to reduce certain properties of D to combinatorial properties of its signature.

Definition 8. *For a ZX diagram D, the* signature graph $\mathscr{G}_D$ *is the undirected graph obtained by:*

1. *Adding a vertex to the open end of each input wire of D;*
2. *Adding a vertex to the open end of each output wire of D;*
3. *Adding a self-loop to each vertex of D whose phase is an odd multiple of $\pi/2$.*

Then the signature *of D is a tuple $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$ consisting of $\mathscr{G}_D$, together with the sets $\mathscr{I}, \mathscr{O} \subseteq V(\mathscr{G}_D) \setminus V(D)$ of added end-points to input/output wires, and a set $\mathscr{P} \subseteq V(D)$ consisting of those vertices of D whose phases are an integer multiple of $\pi/2$.*

An example of a signature graph obtained from a ZX diagram is show in Figure 1.

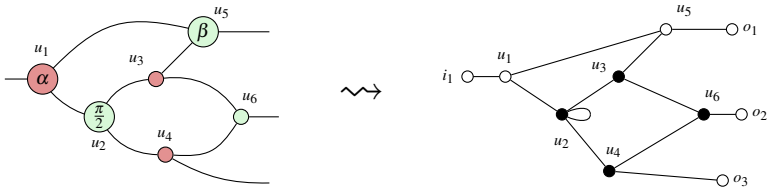


Figure 1: A graph-like ZX-diagram D, with a corresponding signature $\mathscr{G}_D$. Also indicated are the added input vertices $\mathscr{I} = \{i_1\}$ and output vertices $\mathscr{O} = \{o_1, o_2, o_3\}$; vertices in $\mathscr{P}$ are black (all other vertices are white).

3.2 Corrector sets and PF Flows

To realise a ZX diagram as a sequence of operations, one obstacle is the fact that some simple ZX diagrams D_0 — *e.g.*, the maps $A_{V,0}$, $A_{H,0}$, $K_{V,0}$, and $K_{H,0}$ as denoted in Eqn. (4) — do not represent trace-preserving maps on their own, and must be paired with another ZX diagram D_1 as in the AZX diagrams of Definition 1, representing the Kraus operators of a CPTP map.

In each case, D_1 differs from D_0 by a phase operation, which raises the question of the conditions under which such a phase operation can be corrected by adapting operations which may be performed later. For a partial order $\preccurlyeq$ representing a (somewhat flexible) time-ordering of operations, we may consider the conditions under which this is possible for a single ZX generator.

Definition 9 (Correctors)**.** *Let $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$ be a signature of a graph-like ZX diagram D, and $\preccurlyeq$ a partial order on $V(\mathscr{G}_D)$. For vertices $u, v \in V(\mathscr{G}_D)$ and a subset $C \subseteq V(\mathscr{G}_D)$, we say that C is a* v-corrector of u *if $u \in \mathrm{Odd}(C)$, and also $v \prec w$ for all $w \in \big(C \setminus \mathscr{P}\big) \cup \big(\mathrm{Odd}(C) \setminus \{u\}\big)$.*

A v-corrector at u describes a way that a π-phase on some node $u \prec v$ in a ZX diagram D can be propagated into the "future" through some set of nodes C. If we surround all of the nodes $t \in C$ by π-phases of the opposite colour (one on each edge to a different vertex), and if we also negate the phase on x, this preserves the meaning of the diagram D. For a graph-like ZX diagram D, we then propagate those π-phases to the neighbours of t, where they cancel in pairs. As $u \in \mathrm{Odd}(C)$, the overall phase contributed to u by this process is π. Thus, a π-phase at u is equivalent to a π-phase at all nodes $w \in \mathrm{Odd}(C) \setminus \{u\}$, together with a change of sign at all nodes $t \in C$.

The constraint that $v \prec w$ for (some of) the nodes $w \in C \cup \big(\mathrm{Odd}(C) \setminus \{u\}\big)$ is motivated by the idea that the phase on u is determined by an operation (represented by the vertex v) in the immediate past, and must be compensated for by operations which are yet to be performed. A vertex w whose phase angle is changed by a sign, or (apart from u) by a shift of π, is represents an operation which must be adapted to compensate for the phase on u, and must therefore occur in the future of v. The role of $\mathscr{P}$ in the definition of u-correctors arises as follows:

- For a vertex $w \in V(D)$ whose phase is a multiple of π, changing the sign has no effect modulo 2π. Thus, these vertices can be included in C, without requiring $v \prec w$ for that *particular* reason (we may still require $v \prec w$ if $w \in \mathrm{Odd}(C)$, as we must shift then its phase by π.)

- For a vertex w whose phase is an odd multiple of $\pi/2$, negating the phase is equivalent to shifting the phase by π. If $w \in C$, then we may ignore the change of sign if the total phase from *other* elements of C adjacent to it is equivalent to π. Using the fact that w is adjacent to itself, it suffices to adjust its phase (thereby requiring $v \prec w$) only if $w \in \mathrm{Odd}(C)$ as well.

These observations motivate the condition that $v \prec w$ only for those vertices w which either belong to $C \setminus \mathscr{P}$, or to $\mathrm{Odd}(C) \setminus \{u\}$.

We now consider conditions under which every vertex comes is equipped with a corrector set (using the standard definitions given at the start of the section).

Definition 10 (PF-Flow)**.** *For $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$ a signature of a graph-like ZX diagram D, a* PF-flow *is a triple $(\preccurlyeq, f, \mathfrak{C})$ consisting of a partial order $\preccurlyeq$ on $V(\mathscr{G}_D)$, a function $f : V(D) \to V(\mathscr{G}_D)$ and a set $\mathfrak{C} \subset \wp(V(\mathscr{G}_D))$, such that for all $v \in V(D)$:*

(i) If u is adjacent to v in $\mathscr{G}_D$, then either $u \preccurlyeq v$ (and $u \notin \mathscr{O}$), or $v \preccurlyeq u$ (and $u \notin \mathscr{I}$), or both;

(ii) If $\delta^+(v) = 0$, there is a set $C_{v,v} \in \mathfrak{C}$ which is a v-corrector of v;

(iii) For all $u \in N^-(v) \setminus \{f(v)\}$, there is a set $C_{u,v} \in \mathfrak{C}$ which is a v-corrector of u.

That is, any pair of neighbours have some definite ordering in $\preccurlyeq$; if v is a node with no neighbours in its future (which we model as a projection onto some state of one or more qubits), we require a strategy to correct a π-phase on that node; and if v is a node with more than one input (which we model as a composition of merges), we must have a strategy to correct π-phases which might accumulate on its past neighbours, possibly apart from a single distinguished past neighbour.

3.3 Compilation of ZX diagrams to Pauli Fusion diagrams

Having defined PF-Flows as a strategy for correcting phases in a Pauli Fusion procedure, resulting from the different Kraus maps as we attempt to realise different ZX generators as transformations, we consider how this information can be used to deterministically realise a ZX diagram as a transformation. This section is dedicated to the proof of the following Theorem:

Theorem 1. *For any graph-like ZX-diagram D with a PF-Flow, one may construct a runnable Pauli Fusion diagram D_{PF} (with a set $\mathscr{B} \subseteq V(D_{\mathrm{PF}})$ of bit-labels) which realises D in every branch: that is to say, for which $\forall x \in \{0,1\}^{\mathscr{B}} : [\![D_{\mathrm{PF}}(x)]\!] \propto [\![D]\!]$.*

The proof involves a procedure PF-COMPILATION to construct D_{PF}, shown in Figure 2. In the following, we occasionally refer to $u \in V(D)$ as vertex *labels*, as well as vertices. This is important because the diagram D_{PF} is constructed from D in such a way that every node-label in $V(D)$ is also a node-label in $V(D_{\mathrm{PF}})$, but in some cases with significantly different relationships to other vertices. In particular, by the construction of PF-COMPILATION, any label $u \in V(D)$ corresponds to a vertex in D_{PF} with degree at most two.

Lemma 3 (Runnability)**.** *Let D_{PF} be the Pauli Fusion diagram which* PF-COMPILATION *produces from graph-like ZX diagram D and a PF-Flow $(\preccurlyeq, f, \mathfrak{C})$. Then D_{PF} is runnable.*

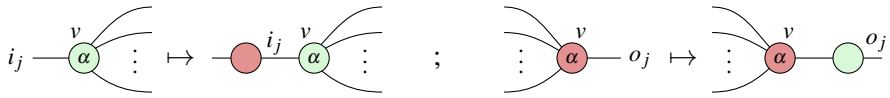
Lemma 4 (Determinism)**.** *Let $\mathscr{B} := \{u \in V(D_{\mathrm{PF}}) \mid u \text{ is a merge or a projection}\}$. For any $s \in \{0,1\}^{\mathscr{B}}$, $[\![D_{\mathrm{PF}}(s)]\!] = \pm [\![D]\!]$.*

The proofs for these Lemmas are given in Appendix A.1 and A.2.

PF-COMPILATION.

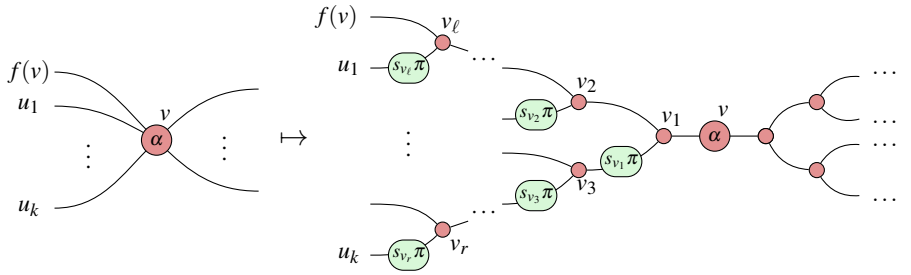
For a graph-like ZX diagram D together with a PF-Flow $(\preccurlyeq, f, \mathfrak{C})$, and given the signature $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$ for D, perform the following transformations on D:

0. **Orientation, inputs, and outputs.** Direct the edges of D consistently with the partial order $\preccurlyeq$. At each input and output, add a trivial node with the corresponding vertex-label $i_j \in \mathscr{I}$ or $o_j \in \mathscr{O}$, with the opposite colour to the first/final node on that input/output, *e.g.*:



(7a)

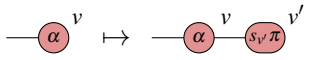
1. **Merge-Split Decomposition.** Decompose each $v \in V(D)$ of degree > 2 using merges, rotations, splits, as in Eqn. (7b). If $\delta^-(v) = 0$, replace the merges with a preparation; and if $\delta^+(v) = 0$, replace the splits with a projection.



(7b)

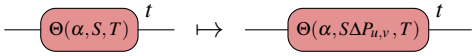
Define $P_{u_i,v} := \{v_k \mid \text{the dot with a } s_{v_k}\pi \text{ phase is on the path from } u_i \text{ to } v\}$.

2. **Projections.** Implement each projection by a rotation and a measurement, *e.g.*:

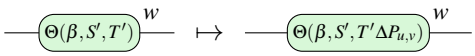


(7c)

3. **Merge-Correction.** For each $v \in V(D)$, each neighbour $u \in N^-(v) \setminus \{f(v)\}$, and each $t \in C_{u,v} \setminus \mathscr{P}$ and $w \in \mathrm{Odd}(C_{u,v}) \setminus \{u\}$, modify the nodes t and w as follows:

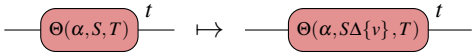


(7d)

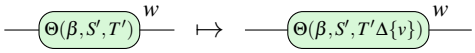


(7e)

4. **Projector-Correction.** For each $v \in V(D)$, each neighbour $u \in N^-(v) \setminus \{f(v)\}$, and each $t \in C_{u,v} \setminus \mathscr{P}$ and $w \in \mathrm{Odd}(C_{u,v}) \setminus \{u\}$, modify the nodes t and w as follows:



(7f)



(7g)

Figure 2: An illustrated procedure to transform a ZX-diagram D with a PF-Flow into a corresponding Pauli Fusion diagram D_{PF}.

PF-FLOW FINDING. For the signature $(\mathscr{G}_D, \mathscr{I}, \mathscr{O}, \mathscr{P})$ of a graph-like ZX diagram D:

Initialise $M := \mathscr{O}$, the set of marked elements;
$\delta M := \mathscr{O}$, a set of newly marked elements;
$\preccurlyeq := \{(u,u) \mid u \in V(\mathscr{G}_D)\}$, a partial order relation on M;
$f := \emptyset$, a function on the empty subset $\emptyset \subset V(\mathscr{G}_D)$;
$\mathfrak{C} := \emptyset$, an empty set of corrector-sets.

Repeat until $\delta M = \emptyset$:

1. Reset $\delta M := \emptyset$.
2. Let R be the set of vertices $u \in V(D) \setminus M$, for which there exists a set $C_u \subseteq M \cup \mathscr{P}$ such that $\mathrm{Odd}(C_u) \setminus M = \{u\}$.
3. For each $v \in V(D) \setminus M$:
   a. Let N_v be a set of vertices "nearby" to v to test for correctability:
      If $N(v) \cap M = \emptyset$, let $N_v := N(v) \cup \{v\}$; otherwise let $N_v := N(v) \setminus M$.
   b. Let $F_v := N_v \setminus R$ be the set of non-correctable vertices nearby to v.
   c. If $|F_v| \leq 1$:
      (*i*) Add v to the set of newly marked elements, $\delta M := \delta M \cup \{v\}$.
      (*ii*) For each $u \in N_v \cap R$: let $C_{u,v} := C_u$ as constructed above, and add it to the set of corrector sets, $\mathfrak{C} := \mathfrak{C} \cup \{C_{u,v}\}$.
   d. If $F_v = \{w\}$ for some $w \in N(v)$, let $f := f \cup \{(v,w)\}$.
   e. If $F_v = \emptyset$, pick some $m \in M$ and let $f := f \cup \{(v,m)\}$.
4. Update the partial order $\preccurlyeq := \preccurlyeq \cup (\delta M \times M)$, so that the old marked elements bound the newly marked elements from above.
5. Update the set of marked elements $M := M \cup \delta M$.

Return $(\preccurlyeq, f, \mathfrak{C})$ if $V(D) \subseteq M$; otherwise return $(\emptyset, \emptyset, \emptyset)$.

Figure 3: A procedure to efficiently construct a PF-Flow, provided one exists.

4 An efficient algorithm for find PF-Flows

Theorem 2. *Given a graph-like ZX-diagram D, there is an efficient algorithm to decide whether it has a PF-Flow, and to construct a PF-Flow if one exists.*

Figure 3 presents an algorithm PF-FLOW FINDING to construct a PF-Flow, if one exists. It determines the partial order $\preccurlyeq$ and the corrector-sets $C_{u,v} \in \mathfrak{C}$, starting from the output, and working back to earlier elements in $\preccurlyeq$ towards the preparations and input.

Lemma 5. PF-FLOW FINDING *halts in time* $\mathrm{poly}(n)$ *for* $n := |V(D)|$.

Lemma 6. *If D is a graph-like ZX diagram with a PF-Flow, then* PF-FLOW FINDING *constructs such a PF-Flow.*

The proofs for these Lemmas are given in Appendix A.3 and A.4.

5 Conclusions

We have here introduced the Pauli Fusion model for quantum computing, enabling us to describe as fundamental information processing operations the splitting and merging of logical Pauli data. We have given the annotated ZX diagrams that directly represent such operations. The PF operations and diagrams are thus designed to represent actual physical processing operations, fulfilling the analogous role to gates in the circuit model.

The relationship between the PF model and standard ZX is an important consideration. Its development was prompted heavily by the many and exciting uses of the ZX calculus in quantum computing. The issue with using standard ZX operationally, as noted in the Introduction, was to translate a ZX diagram into a set of operations to run on a device. This 'circuit extraction' problem has proved both difficult and costly in terms of operational overheads. By introducing an operational representation that is very closely aligned to ZX, we solved the extraction problem almost by fiat – but with one important issue outstanding, whether the ZX diagram could be implemented deterministically with the probabilistic operations of Pauli Fusion. The results in this paper give our solution: when a ZX diagram has a PF-Flow then it may be implemented deterministically. Moreover, we give the polytime algorithm PF-FLOW FINDING that finds the flow and, in the process, converts a ZX diagram to a PF diagram by giving the running time ordering of operations, and demonstrating where any correction operations will have to be performed to preserve the deterministic running of a protocols.

The introduction of Pauli Fusion, and its position as a native operational model for ZX, allows us to envisage using ZX to work the full stack of quantum computing, from design, through to compilation and then operational extraction. This supersedes any need for the old notions of circuits and unitary gates inherited from classical computing. We hope that the PF model will enable full use of the power of ZX for compilation, optimisation, and verification; and new ways of understanding how physical systems process quantum information.

Acknowledgements

We are grateful to Pieter Kok for pointing out that optical fusion gates have similar effects to lattice surgery procedures. NB is supported by the EPSRC National Hub in Networked Quantum Information Technologies (NQIT.org). DH acknowledges financial support from the 'Investissements d'avenir' (ANR-15-IDEX-02) program of the French National Research Agency. SP acknowledges support from the projects ANR-17-CE25-0009 SoftQPro, ANR-17-CE24-0035 VanQuTe, PIA-GDN/Quantex, and LUE / UOQ

References

[1] M. Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014.

[2] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical structures for design and verification of quantum error correction. *arXiv preprint arXiv:1611.08012*, 2016.

[3] B. Coecke and A. Kissinger. *Picturing Quantum Processes: A first course in quantum theory and diagrammatic reasoning*. Cambridge University Press, 2017.

[4] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. *arXiv preprint arXiv:1902.08091*, 2019.

[5] Niel de Beaudrap and Dominic Horsman. The zx calculus is a language for surface code lattice surgery. *arXiv preprint arXiv:1704.08670*, 2017. Accepted to QPL17.

[6] Ross Duncan, Aleks Kissinger, Simon Pedrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. *arXiv preprint arXiv:1902.03178*, 2019.

[7] Ross Duncan and Maxime Lucas. Verifying the steane code with quantomatic. *Electronic Proceedings in Theoretical Computer Science*, (171):33–49, 2014. arXiv preprint arXiv:1902.03178.

[8] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 285–296, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[9] Craig Gidney and Austin G Fowler. Efficient magic state factories with a catalyzed —ccz¿ to 2—t¿ transformation. *arXiv preprint arXiv:1812.01238*, 2018.

[10] Google. `https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html`. Accessed 10/04/2019.

[11] C. Horsman, A. G Fowler, S. Devitt, and R. Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012.

[12] IBM. `https://www.research.ibm.com/ibm-q/`. Accessed 10/04/2019.

[13] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. 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*, pages 559–568. ACM, 2018.

[14] Aleks Kissinger and Arianne Meijer-van de Griend. Cnot circuit extraction for topologically-constrained quantum memories. *arXiv preprint arXiv:1904.00633*, 2019.

[15] Aleks Kissinger and John van de Wetering. Reducing t-count with the zx-calculus. *arXiv preprint arXiv:1903.10477*, 2019.

[16] Pieter Kok. Five lectures on optical quantum computing. *Theoretical Foundations of Quantum Information Processing and Communication: Selected Topics*, 787:187, 2009.

[17] Kang Feng Ng and Quanlong Wang. A universal completion of the zx-calculus. *arXiv preprint arXiv:1706.09877*, 2017.

[18] Renaud Vilmart. A near-optimal axiomatisation of zx-calculus for pure qubit quantum mechanics. *arXiv preprint arXiv:1812.09114*, 2018.

A Proofs

A.1 Proof of runnability

Every node $v \in V(D_{\mathrm{PF}})$ satisfies either $u \prec v$ or $v \prec u$ for each of its neighbours $u \in V(D)$. Then either v is a preparation, adjacent to an input, or has a non-empty set of neighbours which precede it in $\preccurlyeq$. Define a function $t : V(D_{\mathrm{PF}}) \to \mathbb{N}$ recursively by setting $t(v) = 1$ for all nodes v which are preparations or which are adjacent to an input, and by recursively defining

$$t(v) \;=\; 1 + \max\left\{ t(u) \;\middle|\; \big(u \in V(\mathscr{G}_D) \;\&\; u \prec v\big) \text{ or } (u \to v) \in E(D_{\mathrm{PF}}) \right\} \tag{8}$$

for all other $v \in V(D_{\mathrm{PF}})$. By construction, we have $t(u) < t(v)$ for $u, v \in V(D_{\mathrm{PF}})$ with an edge $u \to v$. Furthermore, if for some $u, v \in V(D_{\mathrm{PF}})$ we have $\tau(v) \in \{\mathsf{VRot}^{\alpha,S,T}, \mathsf{HRot}^{\alpha,S,T}\}$ where $u \in S \cup T$, it follows that one of the following two cases hold:

- $\delta^+(u) = 0$, and $v \in \big(C \setminus \mathscr{P}\big) \cup \big(\mathrm{Odd}(C_{u,u}) \setminus \{u\}\big)$, from which it follows that $u \prec v$ by the conditions which hold of $C_{u,u}$. Then $t(u) < t(v)$.
- $v \in \big(C_{\tilde{u},\tilde{v}} \setminus \mathscr{P}\big) \cup \big(\mathrm{Odd}(C_{\tilde{u},\tilde{v}}) \setminus \{\tilde{u}\}\big)$ for some vertices $\tilde{u}, \tilde{v} \in V(\mathscr{G}_D)$ such that $\tilde{u} \in N^-(\tilde{v})$, and it happens that u is an element of $P_{\tilde{u},\tilde{v}}$, the set of vertices which are generated in the decomposition of a higher-degree node with label $\tilde{v}$ in the ZX diagram D, and which lie on a path between $\tilde{u}$ and $\tilde{v}$. From this it follows that $t(\tilde{u}) < t(u) < t(\tilde{v})$; and as $\tilde{v} \prec v$ from the conditions on $C_{\tilde{u},\tilde{v}}$, it follows that $t(u) < t(\tilde{v}) < t(v)$.

Then t is a time-ordering of D_{PF}, from which the Lemma follows.

A.2 Proof of determinism

The proof for Lemma 4 is as follows.

We show, given $s \in \{0,1\}^{\mathscr{B}}$, how to transform $D_{\mathrm{PF}}(s)$ into D using transformations of the ZX-calculus which preserve the semantics (up to a sign). We proceed by induction on the Hamming weight $\|s\|_{\mathrm{H}}$ of s. If $s = 00\cdots0$, then it is easy to see that $[\![D_{\mathrm{PF}}(s)]\!] = [\![D]\!]$: all π-phases depending on bits s_b for $b \in \mathscr{B}$ either contribute 0 to a rotation/projection node, or are equivalent to the identity. We may then use the ZX calculus to reverse the transformations of PF-COMPILATION, thereby obtaining the diagram D.

Suppose instead that there is some $v \in \mathscr{B}$ such that $s_v = 1$. Let $D_{\mathrm{PF}}|_{s_v=b}$ be the AZX diagram obtained from D_{PF} by setting $s_v \leftarrow b$, for any $b \in \{0,1\}$. We show how to rewrite $D_{\mathrm{PF}}|_{s_v=1}$ to $D_{\mathrm{PF}}|_{s_v=0}$ using the ZX calculus, accruing at most a sign of -1 in the semantics as we do so, allowing us to prove the equivalence of $D_{\mathrm{PF}}(s)$ to $D_{\mathrm{PF}}(\tilde{s})$ for a string $\tilde{s}$ that has fewer bits set to 1 than s does.

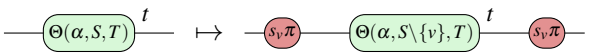
- If $\tau(v) \in \big\{\mathsf{HProj}, \mathsf{VProj}\big\}$, there is a set $C_{v,v} \in \mathfrak{C}$ which is a v-corrector at v: denote this by C, let $\tilde{v} := v$. There is also a node in D_{PF} with a label $w \in \mathrm{Odd}(C)$, such that $\delta^+(w) = 0$ in $\mathscr{G}_D$, with the same colour as v. We rewrite the diagram D_{PF} as an AZX diagram, by propagating the phase $s_v\pi$ from v to w. Denote the resulting diagram by D'_{PF}.

- Otherwise, if $\tau(v) \in \{\mathsf{HMerge}, \mathsf{VMerge}\}$, then there is an associated vertex $\tilde{v} \in V(D_{\mathrm{PF}})$, corresponding to a node-label $\tilde{v} \in V(D)$ of degree 3 or greater, and one or more node-labels $\tilde{u}_1, \tilde{u}_2, \ldots \in N^-(\tilde{v})$ in $\mathscr{G}_D$, such that $v \in P_{\tilde{u}_j,\tilde{v}}$ for each $j \geq 1$. For each of these nodes $\tilde{u}_j$, there is a $\tilde{u}_j$-corrector at $\tilde{v}$, $C_{\tilde{u}_j,\tilde{v}}$. We let C be the symmetric difference $C := C_{\tilde{u}_1,\tilde{v}} \Delta C_{\tilde{u}_2,\tilde{v}} \Delta \cdots$ of these (just $C_{\tilde{u},\tilde{v}}$ if there is only one such $\tilde{u}$). We also rewrite the diagram D_{PF} as an AZX diagram, by propagating the phase $s_v\pi$ away from the node v, against the orientation of the edges, towards the nodes with labels $\tilde{u}_j$, explicitly accumulating the phase $s_v\pi$ on these nodes. (This may involve one or application of the bialgebra law between $s_v\pi$ nodes and opposite-coloured nodes of degree 3, and applications of the spider rule between the nodes $\tilde{u}_j$ and $s_v\pi$ phase nodes of the same colour.) Denote the resulting diagram by D'_{PF}.

In either case, we have a set C which is involved in the correction of vertices, which is involved in annotations on other vertices involving the bit s_v. Specifically, if $\tilde{v} = v$ is a projection, then s_v governs a π-phase contribution for all nodes $w \in \mathrm{Odd}(C) \setminus \{v\}$, and a change in sign of the phase of all nodes $t \in C \setminus \mathscr{P}$. Otherwise v is a vertex in each of a collection of paths $P_{\tilde{u}_j,v}$, so that s_v governs (possibly canceling) π-phase contributions for all nodes $w \in \mathrm{Odd}(C_{\tilde{u}_j,\tilde{v}}) \setminus \{\tilde{v}\}$ for each $\tilde{u}_j$, and (possibly canceling) sign-flips of the phase of all nodes $t \in C_{\tilde{u}_j,\tilde{v}} \setminus \mathscr{P}$ for each $\tilde{u}_j$.

Using the set C constructed as above, we consider the following rewrites on D_{PF}:

1. For each vertex-label $t \in C \setminus \mathscr{P}$, by construction the diagram D_{PF} will contain a generator $\mathsf{VRot}_t^{\alpha,S,T}$ or $\mathsf{HRot}_t^{\alpha,S,T}$ where $v \in S$. We thus rewrite the diagram as an AZX diagram by surrounding t with $s_v\pi$-phase nodes of the opposite colour, and remove v from the set of variables S which may govern a change of sign of the rotation, *e.g.*:

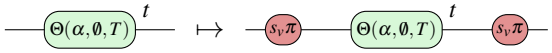


(9a)

   Denote the resulting diagram by D''_{PF}.

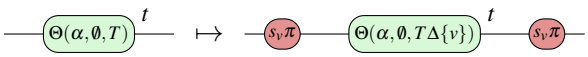
2. For each vertex-label $t \in C \cap \mathscr{P}$, by construction the diagram D_{PF} will contain a generator $\mathsf{VRot}_t^{\alpha,\emptyset,T}$ or $\mathsf{HRot}_t^{\alpha,\emptyset,T}$ where α is an integer multiple of $\pi/2$ (*i.e.*, $2\alpha/\pi$ is an integer), and v may or may not be an element of T.

   - If $2\alpha/\pi$ is even, then we rewrite the diagram by surrounding t with $s_v\pi$-phase nodes of the opposite colour, without any other changes, *e.g.*:



(9b)

   - If $2\alpha/\pi$ is odd, then we rewrite the diagram by surrounding t with $s_v\pi$-phase nodes of the opposite colour, and "toggling" the membership of v in T, *e.g.*:



(9c)

   Denote the resulting diagram by D'''_{PF}.

3. For any $s_v\pi$ phase produced in the previous steps which is adjacent to a node $t \in C$, propagate the phase node away from t (either consistently with the orientation of the edges, or consistently against the orientation) until it is adjacent to a node with label $w \in \mathrm{Odd}(C)$ of the same colour, or more generally forms part of a chain of $s_v\pi$ phase nodes of which one end is adjacent to a node with label $w \in \mathrm{Odd}(C)$ of the same colour. To do this, it may be necessary to propagate two nodes with phase $s_v\pi$ of opposite colour past one another in opposite directions: this induces at most a change of sign due to anticommutation of X and Z. Denote the resulting diagram by $\tilde{D}_{\mathrm{PF}}$.
4. For each $w \in \mathrm{Odd}(C)$, $\tilde{D}_{\mathrm{PF}}$ contains a generator $\mathsf{HRot}_w^{\alpha,S,T}$ or a generator $\mathsf{VRot}_w^{\alpha,S,T}$.
   - If α is an odd multiple of $\pi/2$ and $w \in C$, then by construction there will be an even number of $s_v\pi$ phase nodes, either adjacent to w it or more generally in one or two chains which are adjacent to w, and we will have $v \notin T$. We may then absorb all of these phases into w without modifying the generator at w.
   - If α is an odd multiple of $\pi/2$ and $w \notin C$, or if α is not an odd multiple of $\pi/2$, then by construction there will be an odd number of $s_v\pi$ phase nodes, either adjacent to w it or more generally in one or two chains which are adjacent to w, and we will have $v \in T$. We may then absorb all of these phases into w if we replace $\mathsf{HRot}_w^{\alpha,S,T}$ with $\mathsf{HRot}_w^{\alpha,S,T\Delta\{v\}}$ or replace $\mathsf{VRot}_w^{\alpha,S,T}$ with $\mathsf{VRot}_w^{\alpha,S,T\Delta\{v\}}$.

This sequence of rewrites has the effect of removing all instances of s_v from the diagram, *i.e.*, it removes v from all sets which modify the phases of rotations, without affecting any other influences on the phases and (at the end) without there being any other change to to the structure of the diagram from D_{PF}. Thus the diagram is equivalent to $D_{\mathrm{PF}}|_{s_v=0}$, while incurring a change in semantics by at most a sign. From this it follows that $[\![D_{\mathrm{PF}}(s)]\!] = \pm[\![D_{\mathrm{PF}}(\tilde{s})]\!]$; by induction, it follows that $[\![D_{\mathrm{PF}}(s)]\!] = \pm[\![D_{\mathrm{PF}}(00\cdots0)]\!]$ for any $s \in \{0,1\}^{\mathscr{B}}$.

This completes the proof.

A.3 Proof of halting

The proof of Lemma 5 is as follows.

As the first operation of the loop is to set $\delta M := \emptyset$, the algorithm will terminate in any loop where Step 3c(i) is not executed at least once, in which an element of $V(D) \setminus M$ is added to δM. If this step is run, then Step 5. increases the size of M, decreasing the set of elements which may be added to δM in subsequent loops. It follows that the loop is executed at most n times.

In each iteration, the operations performed consist largely of tests for membership in sets, or constructions of intersections, subtractions, unions, or products of sets, each of which can be performed in polynomial time. The step whose cost is least obvious is the computation of the set R, whose cost we describe below.

Given a vertex u, finding whether there is a corresponding set C_u amounts to solving a system of equations in $\mathbb{F}_2$,

$$\mathbf{A}_{[M]}\,\mathbf{x} = \mathbf{e}_u\,, \tag{10}$$

where $\mathbf{e}_u \in \mathbb{F}_2^{V(D)\setminus M}$ is the characteristic vector of the set $\{u\} \subseteq V(D) \setminus M$, and $\mathbf{A}_{[M]}$ denotes the submatrix of the adjacency matrix $\mathbf{A}$ of D whose rows are indexed by $V(D) \setminus M$ and whose columns are indexed by $M \cup \mathscr{P}$. Each column of $\mathbf{A}_{[M]}$ is the characteristic vector for the set of vertices which are adjacent to an element of $t \in M \cup \mathscr{P}$; and which are not yet marked. Then $\mathbf{A}_{[M]}\,\mathbf{x}$ represents the set of unmarked vertices which are adjacent to an odd number of some set of vertices $X \subset M \cup \mathscr{P}$, whose elements are indicated by the non-zero coefficients of $\mathbf{x}$. Determining whether Eqn. (10) has solutions can be performed efficiently, as can producing one such solution $\mathbf{c}_u$, which then represents a characteristic vector of a corrector set C_u.

It follows that PF-FLOW FINDING halts in polynomial time for any graph-like ZX diagram D.

A.4 Proof of success of PF-FLOW FINDING algorithm

The proof of Lemma 5 is as follows.

Let $(\precsim, \tilde{f}, \tilde{\mathfrak{C}})$ be a PF-Flow for D. In particular, any two vertices which are adjacent in $\mathscr{G}_D$ are comparable in $\preccurlyeq$; there is a v-corrector at v ($\tilde{C}_{v,v} \in \tilde{\mathfrak{C}}$) for any vertex $v \in V(D)$ whose neighbours all precede it; and there is a u-corrector at v ($\tilde{C}_{u,v} \in \mathfrak{C}$) for any adjacent vertices $u \preccurlyeq v$ from D such that $u \neq \tilde{f}(v)$.

Consider a maximal element $\omega \in V(D)$ with respect to $\precsim$. As any $t \in V(\mathscr{G}_D) \setminus \{\omega\}$ for which $\omega \precsim t$ can neither be an element of $V(D)$ nor $\mathscr{I}$, it would follow that $t \in \mathscr{O}$. Thus for C any u-corrector set at ω, whether $u = \omega$ or $u \in N(\omega)$, must have the properties that $C \setminus \mathscr{P} \subseteq \mathscr{O}$ and $\mathrm{Odd}(C) \setminus \{w\} \subseteq \mathscr{O}$. On the first iteration of the loop, we have $M = \mathscr{O}$, so that the corrector sets $\tilde{C}_{u,\omega}$ satisfy the conditions in the construction of the set R (though the algorithm may find different corrector sets C_u), for all $u \in N(\omega) \setminus M$ excepting possibly $u = \tilde{f}(\omega)$. (If ω has no neighbours in $\mathscr{O}$, a corrector set $\tilde{C}_{\omega,\omega}$ also exists, so that $\omega \in R$ at this stage as well. Again, the corrector set C_ω may differ from $\tilde{C}_\omega$.) In the iteration through $v \in V(D) \setminus M$ in this first loop, we will then find $|F_\omega| \leq 1$, and add ω to δM, so that it will become a maximal element of the relation $\preccurlyeq$. If F_ω is non-empty, it will contain $\tilde{f}(\omega)$, so that we will have $f(\omega) = \tilde{f}(\omega)$ in this case.

We may show by induction that PF-FLOW FINDING will construct a PF-Flow $(\preccurlyeq, f, \mathfrak{C})$, by noting that in each iteration there will be a maximal element of $\precsim$ in $V(D)$ subject to not yet having been marked in a previous iteration, which by a similar analysis will be added to δM in that iteration, and that the data $(\preccurlyeq, f, \mathfrak{C})$ satisfy the properties of a PF-Flow by construction.