

## There and back again

Backens, Miriam; Miller-Bakewell, Hector; de Felice, Giovanni; Lobski, Leo; van de Wetering, John

DOI:

[10.22331/q-2021-03-25-421](https://doi.org/10.22331/q-2021-03-25-421)

License:

Creative Commons: Attribution (CC BY)

*Document Version*

Publisher's PDF, also known as Version of record

*Citation for published version (Harvard):*

Backens, M, Miller-Bakewell, H, de Felice, G, Lobski, L & van de Wetering, J 2021, 'There and back again: a circuit extraction tale', *Quantum*, vol. 5, pp. 421. <https://doi.org/10.22331/q-2021-03-25-421>

[Link to publication on Research at Birmingham portal](#)

### General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
- User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
- Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

### Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact [UBIRA@lists.bham.ac.uk](mailto:UBIRA@lists.bham.ac.uk) providing details and we will remove access to the work immediately and investigate.

# There and back again: A circuit extraction tale

Miriam Backens<sup>1</sup>, Hector Miller-Bakewell<sup>2</sup>, Giovanni de Felice<sup>2</sup>, Leo Lobski<sup>3</sup>, and John van de Wetering<sup>4</sup>

<sup>1</sup>University of Birmingham

<sup>2</sup>University of Oxford

<sup>3</sup>ILLC, University of Amsterdam (until 2020)

<sup>4</sup>Radboud University Nijmegen

March 25, 2021

Translations between the quantum circuit model and the measurement-based one-way model are useful for verification and optimisation of quantum computations. They make crucial use of a property known as gflow. While gflow is defined for one-way computations allowing measurements in three different planes of the Bloch sphere, most research so far has focused on computations containing only measurements in the XY-plane. Here, we give the first circuit-extraction algorithm to work for one-way computations containing measurements in all three planes and having gflow. The algorithm is efficient and the resulting circuits do not contain ancillae. One-way computations are represented using the ZX-calculus, hence the algorithm also represents the most general known procedure for extracting circuits from ZX-diagrams. In developing this algorithm, we generalise several concepts and results previously known for computations containing only XY-plane measurements. We bring together several known rewrite rules for measurement patterns and formalise them in a unified notation using the ZX-calculus. These rules are used to simplify measurement patterns by reducing the number of qubits while preserving both the semantics and the existence of gflow. The results can be applied to circuit optimisation by translating circuits to patterns and back again.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Our contributions . . . . .	3
<b>2</b>	<b>Preliminaries</b>	<b>5</b>
2.1	The ZX-calculus . . . . .	5
2.2	Measurement-based quantum computation . . . . .	8
2.3	Graph-theoretic rewriting . . . . .	12
2.4	Generalised flow . . . . .	15

Miriam Backens: [m.backens@cs.bham.ac.uk](mailto:m.backens@cs.bham.ac.uk)

Hector Miller-Bakewell: [hector.miller-bakewell@cs.ox.ac.uk](mailto:hector.miller-bakewell@cs.ox.ac.uk)

Giovanni de Felice: [giovanni.defelice@cs.ox.ac.uk](mailto:giovanni.defelice@cs.ox.ac.uk)

Leo Lobski: [contact@leolob.ski](mailto:contact@leolob.ski)

John van de Wetering: [john@vdwetering.name](mailto:john@vdwetering.name)

2.5	Focusing gflow for XY plane measurements . . . . .	19
<b>3</b>	<b>Rewriting while preserving the existence of gflow</b>	<b>21</b>
3.1	Graph operations that preserve the existence of gflow . . . . .	21
3.2	Finding extended gflow . . . . .	24
3.3	Focusing extended gflow . . . . .	25
<b>4</b>	<b>Simplifying measurement patterns</b>	<b>27</b>
4.1	Transforming local Cliffords into measurements . . . . .	28
4.2	Local complementation and pivoting on patterns . . . . .	29
4.3	Removing Clifford vertices . . . . .	32
4.4	Phase-gadget form . . . . .	35
4.5	Further pattern optimisation . . . . .	37
<b>5</b>	<b>Circuit extraction</b>	<b>39</b>
5.1	General description of the algorithm . . . . .	42
5.2	A more practical algorithm . . . . .	45
<b>6</b>	<b>Conclusions and Future Work</b>	<b>47</b>
<b>A</b>	<b>Gflow and determinism</b>	<b>51</b>
<b>B</b>	<b>Proofs for Section 3</b>	<b>55</b>
<b>C</b>	<b>Finding maximally delayed gflow in multiple measurement planes</b>	<b>59</b>
<b>D</b>	<b>Pseudocode for algorithms</b>	<b>65</b>

## 1 Introduction

The gate-based model and the measurement-based model are two fundamentally different approaches to implementing quantum computations. In the gate-based model [47], the bulk of the computation is performed via unitary (i.e. reversible) one- and two-qubit gates. Measurements serve mainly to read out data and may be postponed to the end of the computation. Conversely, in the measurement-based model, the bulk of the computation is performed via measurements on some general resource state, which is independent of the specific computation. We focus here on the one-way model [49], where the resource states are *graph states* (see Section 2.2). In this paper, we study ways of converting between these two different approaches to quantum computation with a view towards optimising the implementation of both.

While computations in the gate-based model are represented as quantum circuits, computations in the one-way model are usually represented by *measurement patterns*, which describe both the graph state and the measurements performed on it [18, 19]. Measurement patterns in the one-way model generally do not allow arbitrary single-qubit measurements. Instead, measurements are often restricted to the ‘planes’ of the Bloch sphere that are orthogonal to the principal axes, labelled the XY-, XZ-, and YZ-planes. In fact, most research has focused on measurements in just the XY-plane, which alone are sufficient for universal quantum computation [18]. Similarly, measurements in the XZ-plane are also universal [43], although this model has been explored less in the literature. In this paper, we will consider measurements in all three of the planes, since this usually leads

to patterns involving fewer qubits, and allows for more non-trivial transformations of the graph state.

Due to the non-deterministic nature of quantum measurements, a one-way computation needs to be adaptive, with later measurement angles depending on the outcomes of earlier measurements [49]. While the ability to correct undesired measurement outcomes is necessary for obtaining a deterministic computation, not all sequences of measurements support such corrections. The ability to perform a deterministic computation depends on the underlying graph state and the choice of measurement planes, which together form an object called a labelled open graph. If all measurements are in the XY-plane, *causal flow* (sometimes simply called ‘flow’) is a sufficient condition for the labelled open graph to support deterministically implementable patterns [17]. Yet causal flow is not necessary for determinism. Instead, the condition of *generalized flow* (or *gflow*) [9] is both sufficient and necessary for deterministic implementability. Gflow can be defined for labelled open graphs containing measurements in all three planes, in which case it is sometimes called *extended gflow* [9, Theorems 2 & 3].

A given representation of some computation can be transformed into a different representation of the same computation using local rewriting. The new representation may be chosen to have more desirable properties. For quantum circuits, such desirable properties include low depth [2], small total gate count [40], or small counts of some particular type of gate, such as the T-gate [3]. For measurement patterns, desirable properties include a small number of qubits [27, 33] or a particularly simple underlying graph state [43].

Local processes can also be used to translate a pattern into a circuit. This is used, for example, to verify that the pattern represents the desired operation [15, 17, 20, 24, 45]. Conversely, a translation of a circuit into a pattern can be used to implement known algorithms in the one-way model, or it can be combined with a translation back to a circuit to trade depth against width, to parallelise Clifford operations, or to reduce the number of T gates [8, 16, 32]. No complete set of rewrite rules is known for quantum circuits or for measurement patterns, although a completeness result does exist for 2-qubit circuits over the Clifford+T gate set [7].

Rewriting of patterns or circuits, as well as translations between the two models, can be performed using the ZX-calculus, a graphical language for quantum computation [11]. This language is more flexible than quantum circuit notation and also has multiple complete sets of graphical rewrite rules [28, 34, 35, 51]. While translating a measurement pattern to a quantum circuit can be difficult, the translation between patterns and ZX-diagrams is straightforward [24, 26, 37].

## 1.1 Our contributions

In this paper, we give an algorithm that extracts a quantum circuit from any measurement pattern whose underlying labelled open graph has extended gflow. Our algorithm does not use ancillae. While circuit extraction algorithms for patterns containing only XY-plane measurements were already known [8, 24, 26, 45], this is the first circuit extraction algorithm for extended gflow, i.e. where patterns may contain measurements in more than one plane. The algorithm works by translating the pattern into the ZX-calculus and transforming the resulting ZX-diagram into a circuit-like form. It generalises a similar algorithm, which works only for patterns where all measurements are in the XY-plane [26]. The circuit extraction algorithm employs the ZX-calculus, so it can be used not only on diagrams arising from measurement patterns but on any ZX-diagram satisfying certain properties. Thus, this procedure is not only the most general circuit extraction algorithm for measurement patterns but also the most general known circuit extraction algorithm

for ZX-diagrams. In particular, our algorithm can handle ZX-diagrams that contain *phase gadgets* [39], which have been used recently in a variety of circuit optimisation schemes [13, 14, 21].

In developing the circuit extraction algorithm, we derive a number of explicit rewrite rules for ZX-diagrams representing measurement patterns, in particular rewrites that involve graph transformations on the underlying resource state. We show how the gflow changes for each of these rewrite rules, i.e. how the rewrites affect the instructions for correcting undesired measurement outcomes, and hence that these transformations preserve the ability to deterministically implement the pattern. Several of these rules were previously employed in the literature, e.g. the pivot-minor transformation in Ref. [43] or the elimination of Clifford measurements first derived in a different context in Ref. [30], but we are the first to explicitly work out the general transformations of the gflow resulting from these transformations of the resource state and the pattern.

The rewrite rules serve not only to prove the correctness of the algorithm, but also to simplify the measurement patterns by reducing the number of qubits involved. It was already known that qubits measured in a Pauli basis can be treated in special ways [9], for instance that these measurements can be performed simultaneously without losing determinism [22], and hence that the depth of the pattern only depends on the number of non-Clifford operations. Additionally, based on [30], Houshmand et al. [33] stated that Pauli measurements can be removed from a pattern as long as the rest of the pattern is transformed appropriately. Note that their work is restricted to patterns with XY-plane measurements only, and that they worked out the effects of the transformation on the gflow for only a few example cases. We extend these results by combining our different rewrite rules to show exactly how the gflow is transformed when a qubit measured in a Pauli basis is removed from a pattern. In particular, we explicitly prove that the existence of gflow is preserved, so that deterministic implementability is maintained. So whereas [22] showed that the depth of a computation depends directly on the number of non-Clifford operations, our results show that the number of measured qubits needed to perform a measurement-based computation is directly related to the number of non-Clifford operations required for the computation, as every non-input qubit measured in a Pauli basis can be removed from a pattern without losing determinism.

These results require us to generalise several concepts originally developed for patterns containing only XY-plane measurements to patterns with measurements in multiple planes. In particular, we adapt the definitions of *focused gflow* [44] and *maximally delayed gflow* [42] to the extended gflow case. Our generalisation of focused gflow differs from the three generalisations suggested by Hamrit and Perdrix [29]; indeed, the desired applications naturally lead us to one unique generalisation (see Section 3.3).

Combined with the known procedure for transforming a quantum circuit into a measurement pattern using the ZX-calculus [26], our pattern simplification and circuit extraction procedure can be used to reduce the T-gate count of quantum circuits [39]. This involves translating the circuit into a ZX-diagram, transforming this to a diagram which corresponds to a measurement pattern, simplifying the pattern, and then re-extracting a circuit. A rough overview of the different translation and optimisation procedures is given in Figure 1.

The remainder of this paper is structured as follows. Known definitions and results relating to ZX-calculus, measurement patterns and gflow are presented in Section 2. We derive the rewrite rules for extended measurement patterns and the corresponding gflow transformations in Section 3. These results are used in Section 4 to simplify measurement patterns in various ways. Then in Section 5, we demonstrate the algorithm for extracting

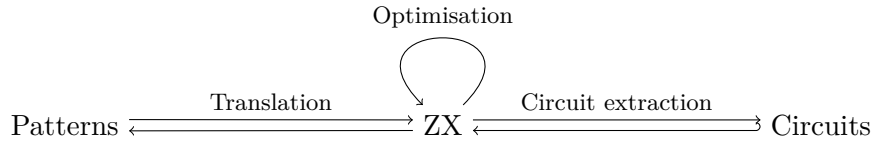


Figure 1: A rough overview over the translation procedures between the three paradigms, indicating where the optimisation steps are carried out.

a circuit from a measurement pattern whose underlying labelled open graph has extended gflow. The conclusions are given in Section 6.

## 2 Preliminaries

We give a quick overview over the ZX-calculus in Section 2.1 and introduce the one-way model of measurement-based quantum computing in Section 2.2. The graph-theoretic operations of local complementation and pivoting (on which the rewrite rules for measurement patterns are based) and their representation in the ZX-calculus are presented in Section 2.3. Section 2.4 contains the definitions and properties of different notions of flow.

### 2.1 The ZX-calculus

The ZX-calculus is a diagrammatic language similar to the widely-used quantum circuit notation. We will provide only a brief overview here, for an in-depth reference see [12].

A ZX-diagram consists of *wires* and *spiders*. Wires entering the diagram from the left are called *input wires*; wires exiting to the right are called *output wires*. Given two diagrams we can compose them horizontally (denoted  $\circ$ ) by joining the output wires of the first to the input wires of the second, or form their tensor product (denoted  $\otimes$ ) by simply stacking the two diagrams vertically.

Spiders are linear maps which can have any number of input or output wires. There are two varieties:  $Z$  spiders, depicted as green dots, and  $X$  spiders, depicted as red dots.<sup>1</sup> Written explicitly in Dirac ‘bra-ket’ notation, these linear maps are:

$$\begin{array}{c} \diagup \\ \text{⋮} \\ \text{⊙} \\ \text{⋮} \\ \diagdown \end{array} := |0\dots 0\rangle\langle 0\dots 0| + e^{i\alpha} |1\dots 1\rangle\langle 1\dots 1| \quad \begin{array}{c} \diagup \\ \text{⋮} \\ \text{⊙} \\ \text{⋮} \\ \diagdown \end{array} := |+\dots +\rangle\langle +\dots +| + e^{i\alpha} |-\dots -\rangle\langle -\dots -|$$

A ZX-diagram with  $m$  input wires and  $n$  output wires then represents a linear map  $(\mathbb{C}^2)^{\otimes m} \rightarrow (\mathbb{C}^2)^{\otimes n}$  built from spiders (and permutations of qubits) by composition and tensor product of linear maps. As a special case, diagrams with no inputs and  $n$  outputs represent vectors in  $(\mathbb{C}^2)^{\otimes n}$ , i.e. (unnormalised)  $n$ -qubit states.

**Example 2.1.** We can immediately write down some simple state preparations and unitaries in the ZX-calculus:

$$\begin{array}{cc}
 \text{⊙} - & = |0\rangle + |1\rangle \propto |+\rangle & \text{⊙} - & = |+\rangle + |-\rangle \propto |0\rangle \\
 \text{⊙} - & = |0\rangle\langle 0| + e^{i\alpha} |1\rangle\langle 1| = Z_\alpha & \text{⊙} - & = |+\rangle\langle +| + e^{i\alpha} |-\rangle\langle -| = X_\alpha
 \end{array}$$

In particular we have the Pauli matrices:

$$\text{⊙} - \text{⊙} = Z \quad \text{⊙} - \text{⊙} = X$$

<sup>1</sup>If you are reading this document in monochrome or otherwise have difficulty distinguishing green and red,  $Z$  spiders will appear lightly-shaded and  $X$  spiders will appear darkly-shaded.

It will be convenient to introduce a symbol – a yellow square – for the Hadamard gate. This is defined by the equation:

$$\text{---} \boxed{\text{H}} \text{---} = \text{---} \square \text{---} := \text{---} \begin{array}{c} \pi \\ \circ \\ 2 \end{array} \begin{array}{c} \pi \\ \circ \\ 2 \end{array} \begin{array}{c} \pi \\ \circ \\ 2 \end{array} \text{---} \quad (1)$$

We will often use an alternative notation to simplify the diagrams, and replace a Hadamard between two spiders by a blue dashed edge, as illustrated below.

$$\begin{array}{c} \diagup \\ \diagdown \end{array} \circ \text{---} \text{---} \text{---} \text{---} \circ \begin{array}{c} \diagdown \\ \diagup \end{array} := \begin{array}{c} \diagup \\ \diagdown \end{array} \circ \text{---} \square \text{---} \circ \begin{array}{c} \diagdown \\ \diagup \end{array}$$

Both the blue edge notation and the Hadamard box can always be translated back into spiders when necessary. We will refer to the blue edge as a *Hadamard edge*.

**Definition 2.2.** The *interpretation* of a ZX-diagram  $D$  is the linear map that such a diagram represents and is written  $\llbracket D \rrbracket$ . For a full treatment of the interpretation of a ZX diagram see, e.g. Ref. [35]. We say two ZX-diagrams  $D_1$  and  $D_2$  are *equivalent* when  $\llbracket D_1 \rrbracket = z \llbracket D_2 \rrbracket$  for some non-zero complex number  $z$ .

We define equivalence up to a global scalar, as those scalars will not be important for the class of diagrams we study in this paper.

**Remark 2.3.** There are many different sets of rules for the ZX-calculus. The version we present only preserves equality up to a global scalar. Versions of the ZX-calculus where equality is ‘on the nose’ can be found in Ref. [5] for the stabiliser fragment, in Ref. [35] for the Clifford+T fragment, and in Ref. [34, 46] for the full language.

The interpretation of a ZX-diagram is invariant under moving the vertices around in the plane, bending, unbending, crossing, and uncrossing wires, so long as the connectivity and the order of the inputs and outputs is maintained. Furthermore, there is an additional set of equations that we call the *rules* of the ZX-calculus; these are shown in Figure 2. Two diagrams are equivalent if one can be transformed into the other using the rules of the ZX-calculus.

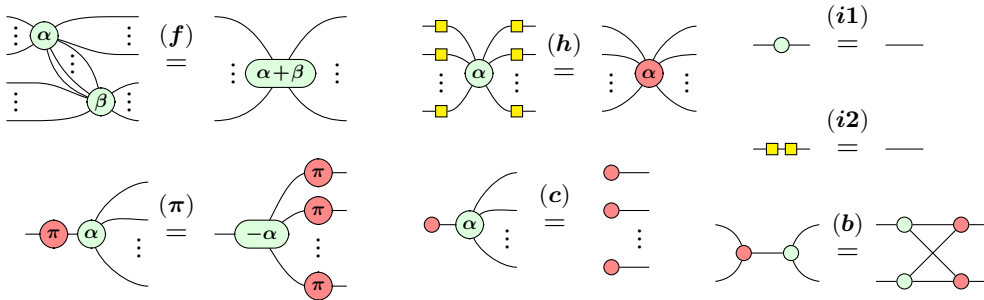


Figure 2: A convenient presentation for the ZX-calculus. These rules hold for all  $\alpha, \beta \in [0, 2\pi)$ , and due to (h) and (i2) all rules also hold with the colours interchanged. Note the ellipsis should be read as ‘0 or more’, hence the spiders on the LHS of (f) are connected by one or more wires.

**Remark 2.4.** The ZX-calculus is *universal* in the sense that any linear map can be expressed as a ZX-diagram. Furthermore, when restricted to *Clifford ZX-diagrams*, i.e. diagrams whose phases are all multiples of  $\pi/2$ , the version we present in Figure 2 can be made *complete* as follows: for any two non-zero Clifford ZX-diagrams that are equivalent (cf. Definition 2.2), there exists a series of rewrites transforming one into the other if a meta

rule is added that allows non-zero scalar subdiagrams to be dropped. For completeness when considering exact equality, see Remark 2.3. Recent extensions to the calculus have been introduced which are complete for the larger *Clifford+T* family of ZX-diagrams [35], where phases are multiples of  $\pi/4$ , and for *all* ZX-diagrams [28, 34, 51].

Quantum circuits can be translated into ZX-diagrams in a straightforward manner. We will take as our starting point circuits constructed from the following universal set of gates:

$$\text{CNOT} \qquad Z_\alpha \qquad H$$

We choose this gate set because it admits a convenient representation in terms of spiders:

$$\text{CNOT} = \begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} \qquad Z_\alpha = \text{---} \circ(\alpha) \text{---} \qquad H = \text{---} \square \text{---} \quad (2)$$

These gates have the following interpretations:

$$\llbracket \begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} \rrbracket = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \llbracket \text{---} \circ(\alpha) \text{---} \rrbracket = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \qquad \llbracket \text{---} \square \text{---} \rrbracket = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

For the CNOT gate, the green spider is the first (i.e. control) qubit and the red spider is the second (i.e. target) qubit. Other common gates can easily be expressed in terms of these gates. In particular,  $S := Z_{\frac{\pi}{2}}$ ,  $T := Z_{\frac{\pi}{4}}$  and:

$$X_\alpha = \begin{array}{c} \square \\ | \\ \circ(\alpha) \\ | \\ \square \end{array} = \begin{array}{c} \bullet(\alpha) \end{array} \qquad \text{CZ} = \begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \bullet \text{---} \\ | \\ \text{---} \square \text{---} \end{array} = \begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \square \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} = \begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \bullet \text{---} \\ | \\ \text{---} \circ \text{---} \end{array} \quad (3)$$

**Remark 2.5.** Note that the directions of the wires in the depictions of the CNOT and CZ gates are irrelevant, hence we can draw them vertically without ambiguity. Undirected wires are a general feature of ZX-diagrams, and from hence forth we will ignore wire directions without further comment. We will also freely draw inputs/outputs entering or exiting the diagram from arbitrary directions if the meaning is clear from context (as for example in Example 2.43).

For our purposes, we will define quantum circuits as a special case of ZX-diagrams.

**Definition 2.6.** A *circuit* is a ZX-diagram generated by composition and tensor products of the ZX-diagrams in equations (2) and (3). The interpretation of such a circuit is given by the composition and tensor products of the interpretations of the generating diagrams given in equation (2), in accordance with:

$$\llbracket D \otimes D' \rrbracket = \llbracket D \rrbracket \otimes \llbracket D' \rrbracket \qquad \llbracket D \circ D' \rrbracket = \llbracket D \rrbracket \circ \llbracket D' \rrbracket$$

Important subclasses of circuits are *Clifford circuits*, sometimes called stabilizer circuits, which are obtained from compositions of only CNOT,  $H$ , and  $S$  gates. They are efficiently classically simulable, thanks to the *Gottesman-Knill theorem* [1]. A unitary is *local Clifford* if it arises from a single-qubit Clifford circuit, i.e. a composition of  $H$  and  $S$  gates. The addition of  $T$  gates to Clifford circuits yields *Clifford+T circuits*, which are capable of approximating any  $n$ -qubit unitary to arbitrary precision, whereas the inclusion of  $Z_\alpha$  gates for all  $\alpha$  enables one to construct any unitary exactly [47].



## 2.2 Measurement-based quantum computation

Measurement-based quantum computing (MBQC) is a universal model for quantum computation, underlying the *one-way quantum computing* scheme [48]. The basic resource for MBQC is a *graph state* [30, 31], built from a register of qubits by applying  $CZ$ -gates pairwise to obtain an entangled quantum state.<sup>2</sup> The graph state is then gradually consumed by measuring single qubits. By choosing an appropriate resource state and appropriate measurements, any quantum computation can be performed. The difficulty is the non-determinism inherent in quantum measurements, which means computations need to be adaptive to implement deterministic operations.

Measurement-based quantum computations are usually formalised in terms of *measurement patterns*, a syntax describing both the construction of graph states and their processing via successive measurements. In the following, we first present measurement patterns, and then introduce other – and, in our opinion, simpler – formalisms for reasoning about these computations. Instead of allowing arbitrary single-qubit measurements, measurements are usually restricted to three planes of the Bloch sphere, labelled XY, XZ, and YZ. For each measurement, the state denoted ‘+’ is taken to be the desired result of the measurement and the state denoted ‘–’ is the undesired result, which will need to be adaptively corrected. The allowed measurements are ([19, p. 292]):

$$\begin{aligned} |+\text{XY},\alpha\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + e^{i\alpha} |1\rangle) & |-\text{XY},\alpha\rangle &= \frac{1}{\sqrt{2}} (|0\rangle - e^{i\alpha} |1\rangle) \\ |+\text{XZ},\alpha\rangle &= \cos\left(\frac{\alpha}{2}\right) |0\rangle + \sin\left(\frac{\alpha}{2}\right) |1\rangle & |-\text{XZ},\alpha\rangle &= \sin\left(\frac{\alpha}{2}\right) |0\rangle - \cos\left(\frac{\alpha}{2}\right) |1\rangle \\ |+\text{YZ},\alpha\rangle &= \cos\left(\frac{\alpha}{2}\right) |0\rangle + i \sin\left(\frac{\alpha}{2}\right) |1\rangle & |-\text{YZ},\alpha\rangle &= \sin\left(\frac{\alpha}{2}\right) |0\rangle - i \cos\left(\frac{\alpha}{2}\right) |1\rangle \end{aligned}$$

where  $0 \leq \alpha \leq 2\pi$ . Usually, the desired measurement outcome is labelled 0 and the undesired measurement outcome is labelled 1.

**Definition 2.7** ([18]). A *measurement pattern* consists of an  $n$ -qubit register  $V$  with distinguished sets  $I, O \subseteq V$  of input and output qubits and a sequence of commands consisting of the following operations:

- Preparations  $N_i$ , which initialise a qubit  $i \notin I$  in the state  $|+\rangle$ .
- Entangling operators  $E_{ij}$ , which apply a  $CZ$ -gate to two distinct qubits  $i$  and  $j$ .
- Destructive measurements  $M_i^{\lambda,\gamma(\alpha,s,t)}$ , which project a qubit  $i \notin O$  onto the orthonormal basis  $\{|+\lambda,\gamma\rangle, |-\lambda,\gamma\rangle\}$ , where  $\lambda \in \{\text{XY}, \text{XZ}, \text{YZ}\}$  is the measurement plane,  $\alpha$  is the non-corrected measurement angle and  $\gamma(\alpha, s, t) := (-1)^s \alpha + t\pi$  is the corrected one with variables  $s, t \in \{0, 1\}$  indicating measurement outcomes. The projector  $|+\lambda,\gamma\rangle \langle +\lambda,\gamma|$  corresponds to outcome 0 and  $|-\lambda,\gamma\rangle \langle -\lambda,\gamma|$  corresponds to outcome 1. If there are no corrections involved, we will simply write  $M_i^{\lambda,\alpha}$ .
- Corrections  $[X_i]^s$ , which depend on a measurement outcome (or a linear combination of measurement outcomes)  $s \in \{0, 1\}$  and act as the Pauli- $X$  operator on qubit  $i$  if  $s$  is 1 and as the identity otherwise,

---

<sup>2</sup>There are models of MBQC where the basic resource is not a graph state, but we do not consider those models in this paper.

- Corrections  $[Z_j]^t$ , which depend on a measurement outcome (or a linear combination of measurement outcomes)  $t \in \{0, 1\}$  and act as the Pauli- $Z$  operator on qubit  $j$  if  $t$  is 1 and as the identity otherwise.

We will only consider *runnable patterns*, which satisfy certain conditions ensuring they are implementable in practice.

**Definition 2.8** ([18]). A measurement pattern is *runnable* if the following conditions hold.

- No correction depends on an outcome not yet measured.
- All non-input qubits are prepared.
- All non-output qubits are measured.
- A command  $C$  acts on a qubit  $i$  only if  $i$  has not already been measured, and one of (1)-(3) holds:
  - (1)  $i$  is an input and  $C$  is not a preparation,
  - (2)  $i$  has been prepared and  $C$  is not a preparation,
  - (3)  $i$  has not been prepared,  $i$  is not an input, and  $C$  is a preparation.

The entangling operators  $E_{ij}$  in a pattern determine the resource graph state. In fact, they correspond to the edges of the underlying *labelled open graph*. This is formalised in the following definitions and remark, which will be used throughout the paper.

**Definition 2.9.** An *open graph* is a tuple  $(G, I, O)$  where  $G = (V, E)$  is an undirected graph, and  $I, O \subseteq V$  are distinguished (possibly overlapping) subsets representing *inputs* and *outputs*. We will write  $\bar{O} := V \setminus O$  for the *non-outputs* and  $\bar{I} := V \setminus I$  for the *non-inputs*. If  $v \in \bar{I}$  and  $v \in \bar{O}$ , we call  $v$  an *internal vertex*, and if  $v \in I \cup O$ , we call  $v$  a *boundary vertex*. For vertices  $u, v \in V$  we write  $u \sim v$  when  $u$  and  $v$  are adjacent in  $G$ , and denote by  $N_G(u) := \{w \in V \mid w \sim u\}$  the set of neighbours of  $u$ .

**Definition 2.10** (cf. [9, p.5]). A *labelled open graph* is a tuple  $\Gamma = (G, I, O, \lambda)$  where  $(G, I, O)$  is an open graph, and  $\lambda : \bar{O} \rightarrow \{XY, YZ, XZ\}$  assigns a measurement plane to each non-output vertex.<sup>3</sup>

**Remark 2.11.** A labelled open graph and an assignment of measurement angles  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  carry the same information as a runnable measurement pattern with no corrections.

Given the measurement pattern, the corresponding labelled open graph  $(G, I, O, \lambda)$  may be determined as follows: the vertices of the graph  $G$  are given by the set of qubits  $V$ . The edges of  $G$  are given by the set of pairs  $(i, j) \in V \times V$  such that the total number of occurrences of  $E_{ij}$  and  $E_{ji}$  in the pattern is odd. The sets  $I$  and  $O$  are the ones given in Definition 2.7. The functions  $\lambda$  and  $\alpha$  are determined by the measurement operators  $M_i^{\lambda, \alpha}$ ; Definition 2.8 guarantees that both are well-defined.

Given a labelled open graph we can apply this process in reverse to construct a runnable measurement pattern without corrections.

A labelled open graph and an assignment of measurement angles can also be determined from a measurement pattern including corrections by simply ignoring the latter (i.e. by

---

<sup>3</sup>We adopt the notation of this from Ref. [9] where their corresponding notion is called an ‘open graph state’. While they explicitly consider this to be a particular type of quantum state, our notion of a labelled open graph is just a type of graph. We define the associated ‘open graph state’ separately in Definition 2.12.






operator	$N_i$	$E_{ij}$	$\langle +_{XY,\alpha(i)} \rangle_i$	$\langle +_{XZ,\alpha(i)} \rangle_i$	$\langle +_{YZ,\alpha(i)} \rangle_i$
diagram					

Table 1: Translation from an associated linear map to a ZX-diagram.

assuming that all measurements yield the desired result). In Section 2.4, we discuss necessary and sufficient conditions under which appropriate corrections commands can be determined when constructing a measurement pattern from a labelled open graph; these corrections then put constraints on the order of the measurements.

In general, a single measurement pattern can result in a variety of measurement instructions, with each instruction depending on earlier measurement outcomes and the resulting corrections that must be applied. We are, however, interested in the subset of measurement patterns that always implement the same linear map, regardless of the measurement outcomes. For these patterns, we can identify the unique linear map implemented by the pattern with the set of measurement instructions obtained when all the measurement outcomes are 0 (and thus no corrections are necessary). This leads us to the following definition:

**Definition 2.12.** Suppose  $\Gamma = (G, I, O, \lambda)$  is a labelled open graph, and let  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  be an assignment of measurement angles. The *linear map associated with  $\Gamma$  and  $\alpha$*  is given by

$$M_{\Gamma, \alpha} := \left( \prod_{i \in \bar{O}} \langle +_{\lambda(i), \alpha(i)} \rangle_i \right) E_G N_{\bar{I}},$$

where  $E_G := \prod_{i \sim j} E_{ij}$  and  $N_{\bar{I}} := \prod_{i \in \bar{I}} N_i$ .

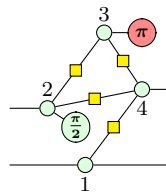
**Remark 2.13.** Note that the projections  $\langle +_{\lambda(i), \alpha(i)} \rangle_i$  on different qubits  $i$  commute with each other. Similarly, the controlled-Z operations  $E_{ij}$  commute even if they involve some of the same qubits. Finally, all the state preparations  $N_i$  on different qubits commute. Thus,  $M_{\Gamma, \alpha}$  is fully determined by  $\Gamma$  and  $\alpha$ .

**Definition 2.14.** Suppose  $\Gamma = (G, I, O, \lambda)$  is a labelled open graph and  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  is an assignment of measurement angles. Then its *associated ZX-diagram*  $D_{\Gamma, \alpha}$  is defined by translating the expression for  $M_{\Gamma, \alpha}$  from Definition 2.12 according to Table 1. The elements are composed in the obvious way and any sets of adjacent phase-free green spiders other than measurement effects are merged. In other words, merging affects green spiders which come from the translation of a preparation or entangling command.

**Example 2.15.** The measurement pattern with the qubit register  $V = \{1, 2, 3, 4\}$ , input and output sets  $I = \{1, 2\}$  and  $O = \{1, 4\}$  and the sequence of commands

$$M_2^{XY, \frac{\pi}{2}} M_3^{YZ, \pi} E_{14} E_{23} E_{24} E_{34} N_3 N_4$$

is represented by the following ZX-diagram:



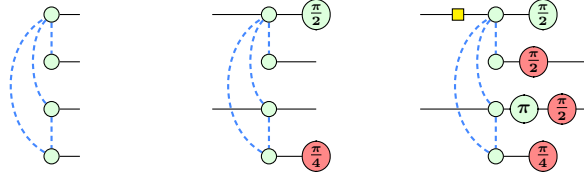


Figure 3: On the left, a graph state diagram. In the middle, a diagram in MBQC form with the same underlying graph. On the right, an MBQC+LC form diagram with the same underlying labelled open graph.

**Lemma 2.16.** Suppose  $\Gamma = (G, I, O, \lambda)$  is a labelled open graph and  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  is an assignment of measurement angles. Let  $M_{\Gamma, \alpha}$  be the linear map specified in Definition 2.12 and let  $D_{\Gamma, \alpha}$  be the ZX-diagram constructed according to Definition 2.14. Then  $\llbracket D_{\Gamma, \alpha} \rrbracket = M_{\Gamma, \alpha}$ .

*Proof.* For each operator  $M$  in Table 1 and its corresponding diagram  $D_M$ , it is straightforward to check that  $\llbracket D_M \rrbracket = M$ . The result thus follows by the compositional properties of the interpretation  $\llbracket \cdot \rrbracket$  and the fact that rewriting ZX-diagrams preserves semantics.  $\square$

In order to specify a converse direction to this result, we will define a special class of ZX-diagrams. Before we do that, we recall which ZX-diagrams correspond to graph states.

**Definition 2.17** ([23]). A *graph state diagram* is a ZX-diagram where all vertices are green, all the connections between vertices are Hadamard edges and a single output wire is incident on each vertex in the diagram. The *graph corresponding to a graph state diagram* is the graph whose vertices are the green spiders of the ZX-diagram and whose edges are given by the Hadamard edges of the ZX-diagram.

**Definition 2.18.** A ZX-diagram is in *MBQC form* if it consists of a graph state diagram in which each vertex of the graph may also be connected to:

- an input (in addition to its output), and
- a measurement effect (in one of the three measurement planes) instead of the output.

**Definition 2.19.** Given a ZX-diagram  $D$  in MBQC form, its *underlying graph*  $G(D)$  is the graph corresponding to the graph state part of  $D$ .

See Figure 3 for an example of a graph state diagram and a diagram in MBQC form. Given these definitions we can now show the following:

**Lemma 2.20.** Let  $\Gamma = (G, I, O, \lambda)$  be a labelled open graph and let  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  be an assignment of measurement angles. Then the ZX-diagram  $D_{\Gamma, \alpha}$  constructed according to Definition 2.14 is in MBQC form.

*Proof.* Consider performing the translation described in Definition 2.14 in two steps. The first step involves translating the preparation and entangling commands of the linear map  $M_{\Gamma, \alpha}$  according to Table 1 and then merging any sets of adjacent green spiders. This yields a graph state diagram with some additional inputs. (The underlying graph is  $G$ .) The second step is the translation of the measurement projections of  $M_{\Gamma, \alpha}$ . This yields measurement effects on some of the outputs of the graph state diagram. Thus, the resulting ZX-diagram is in MBQC form by Definition 2.18.  $\square$

The converse of Lemma 2.20 also holds.

**Lemma 2.21.** Suppose  $D$  is a ZX-diagram in MBQC form. Then there exists a labelled open graph  $\Gamma = (G, I, O, \lambda)$  and an assignment of measurement angles  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  such that  $\llbracket D \rrbracket = M_{\Gamma, \alpha}$ .

*Proof.* Let  $G := G(D)$  be the underlying graph of the ZX-diagram  $D$ , cf. Definition 2.19. Define  $I \subseteq V$  to be the set of vertices of  $D$  on which an input wire is incident. Analogously, define  $O \subseteq V$  to be the set of vertices of  $D$  on which an output wire is incident. Fix  $\lambda : \bar{O} \rightarrow \{XY, XZ, YZ\}$  by using Table 1 in reverse to determine the measurement planes from the measurement effects in the ZX-diagram. Let  $\Gamma := (G, I, O, \lambda)$ . Finally, define  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  to be the phase of the measurement effect connected to each non-output vertex in the ZX-diagram. Then  $D = D_{\Gamma, \alpha}$  and thus the desired result follows from Lemma 2.16.  $\square$

**Remark 2.22.** Lemmas 2.20 and 2.21 show that the correspondence between MBQC form ZX-diagrams and the pairs  $(\Gamma, \alpha)$ , where  $\Gamma$  is a labelled open graph and  $\alpha$  is an assignment of measurement angles, is one-to-one up to isomorphisms of labelled open graphs that also preserve  $\alpha$ .

It will turn out to be useful to consider a ‘relaxed’ version of the MBQC form for ZX-diagrams.

**Definition 2.23.** We say a ZX-diagram is in *MBQC+LC* form when it is in MBQC form (see Definition 2.18) up to arbitrary single-qubit Clifford unitaries on the input and output wires (LC stands for ‘local Clifford’). When considering the underlying graph of a ZX-diagram in MBQC+LC form, we ignore these single qubit Clifford unitaries.

Note that an MBQC form diagram is an MBQC+LC form diagram with trivial single-qubit unitaries on its inputs and outputs. An example diagram in MBQC+LC form is given in Figure 3.

### 2.3 Graph-theoretic rewriting

The rewrites we will use are based on the graph-theoretic notions of *local complementation* and *pivoting*. We present these operations (and their effects) in Definitions 2.24 and 2.25 as they appear in Ref. [25]. Our interest is in the effect these operations have on a measurement pattern. In particular, we consider whether a ZX-diagram in MBQC form will remain in MBQC form after applying a local complementation or pivot (or remain close enough to MBQC form to still be useful).

**Definition 2.24** ([41]). Let  $G = (V, E)$  be a graph and  $u \in V$  a vertex. The *local complementation of  $G$  about the vertex  $u$*  is the operation resulting in the graph

$$G \star u := (V, E \Delta \{(b, c) : (b, u), (c, u) \in E \text{ and } b \neq c\}),$$

where  $\Delta$  is the symmetric set difference, i.e.  $A \Delta B := (A \cup B) \setminus (A \cap B)$ .

In other words,  $G \star u$  is a graph that has the same vertices as  $G$ . Two neighbours  $b$  and  $c$  of  $u$  are connected in  $G \star u$  if and only if they are not connected in  $G$ . All other edges are the same as in  $G$ .

**Definition 2.25** ([41]). Let  $G = (V, E)$  be a graph and  $u, v \in V$  two vertices connected by an edge. The *pivot of  $G$  about the edge  $u \sim v$*  is the operation resulting in the graph  $G \wedge uv := G \star u \star v \star u$ .

If we denote the set of vertices connected to both  $u$  and  $v$  by  $A$ , the set of vertices connected to  $u$  but not to  $v$  by  $B$  and the set of vertices connected to  $v$  but not to  $u$  by  $C$ , then pivoting consists of interchanging  $u$  and  $v$  and complementing the edges between each pair of sets  $A$ ,  $B$  and  $C$ . That is, a vertex in  $A$  is connected to a vertex in  $B$  after pivoting if and only if the two vertices are not connected before pivoting; and similarly for the two other pairs. All the remaining edges are unchanged, including the edges internal to  $A$ ,  $B$  and  $C$ . We illustrate this by the following picture, where crossing lines between two sets indicate complementing the edges.



**Remark 2.26.** From the above characterisation it follows that pivoting is symmetric in the (neighbouring) vertices, that is,  $G \wedge uv = G \wedge vu$ .

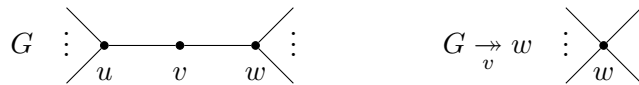
In the ZX-calculus, a spider with a zero phase and exactly two incident wires is equivalent to a plain wire (representing the identity operation) by rule (i1) in Figure 2. The following definition represents the corresponding graph operation, which will be used to remove such vertices.

**Definition 2.27.** Let  $G = (V, E)$  be a graph and let  $u, v, w \in V$  be vertices such that  $N_G(v) = \{u, w\}$  and  $u \notin N_G(w)$ , that is, the neighbours of  $v$  are precisely  $u$  and  $w$ , and  $u$  is not connected to  $w$ . We then define *identity removal* as

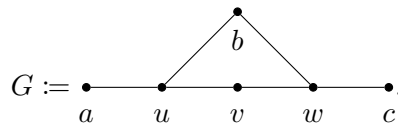
$$G \xrightarrow{v} w := ((G \wedge uv) \setminus \{u\}) \setminus \{v\}.$$

Since  $v$  has exactly two neighbours, one of which is  $w$ , the choice of  $u$  is implicit in the notation. We think of this as ‘dragging  $u$  along  $v$  to merge with  $w$ ’.

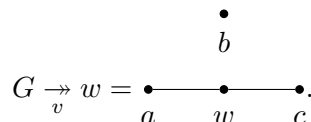
The effect of the identity removal is to remove the middle vertex  $v$  and to fuse the vertices  $u$  and  $w$  into one, as illustrated in the picture below. Thus the operation is symmetric in  $u$  and  $w$ , in the sense that  $G \xrightarrow{v} u$  and  $G \xrightarrow{v} w$  are equal up to a relabelling of one vertex. Note that  $u$  and  $w$  are allowed to have common neighbours (which will disconnect from the fused vertex  $w$  as a result of identity removal).



**Example 2.28.** Consider the following graph:



Note that the vertices  $u, v$  and  $w$  satisfy the condition for identity removal:  $u$  and  $w$  are not connected and are precisely the neighbours of  $v$ . Hence identity removal results in the graph



**Remark 2.29.** If we have vertices  $u, v$  and  $w$  with  $N_G(v) = \{u, w\}$  but, unlike in the definition above,  $u \in N_G(w)$ , we can first perform a local complementation on  $v$ , so that  $u$  and  $w$  become disconnected, and then remove the identity vertex  $v$ . In symbols:

$$(G \star v) \xrightarrow{v} w.$$

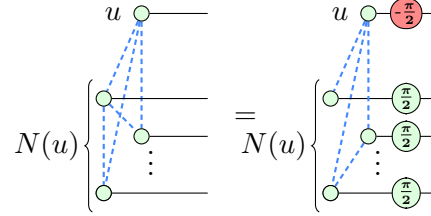
The abstract application of a local complementation to a graph corresponds to the application of a specific set of local Clifford gates on the corresponding graph state:

**Theorem 2.30** ([50], in the manner of [23, Theorem 2]). Let  $G = (V, E)$  be a graph with adjacency matrix  $\theta$  and let  $u \in V$ , then

$$|G \star u\rangle = X_{\pi/2, u} \otimes \bigotimes_{v \in V} Z_{-\pi/2, v}^{\theta_{uv}} |G\rangle.$$

This result can be represented graphically in the ZX-calculus:

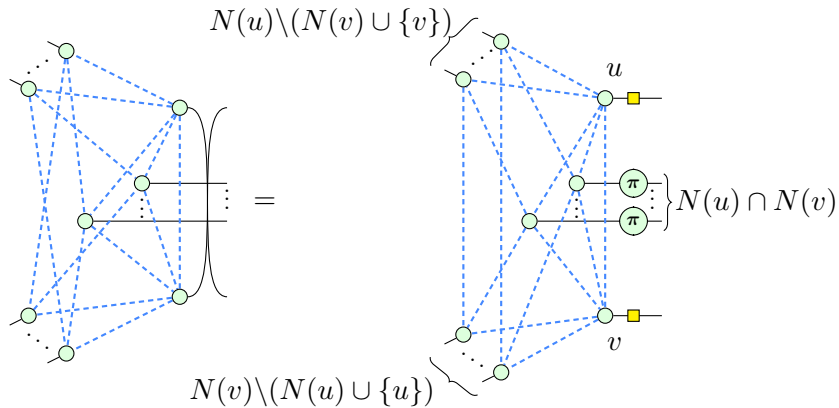
**Lemma 2.31** ([23, Theorem 3]). The following equality involving graph state diagrams and Clifford phase shifts follows from the graphical rewrite rules:



Here, the underlying graph on the LHS is  $G \star u$  and the underlying graph on the RHS is  $G$ . Any vertices not adjacent to  $u$  are unaffected and are not shown in the above diagram.

Combining this result with the definition of pivoting in terms of local complementations (cf. Definition 2.25) we also get:

**Lemma 2.32** ([25, Theorem 3.3]). The following diagram equality follows from the graphical rewrite rules:



Here  $u$  and  $v$  are a connected pair of vertices, and the underlying graph on the LHS is  $G \wedge uv$  while the RHS is  $G$ . Any vertices not adjacent to  $u$  or  $v$  are unaffected and are not shown in the above diagram.

## 2.4 Generalised flow

The notion of *flow* or *causal flow* on open graphs was introduced by Danos and Kashefi [17] as a sufficient condition to distinguish those open graphs capable of supporting a deterministic MBQC pattern with measurements in the XY-plane. Causal flow, however, is not a necessary condition for determinism. That is there are graphs that implement a deterministic pattern even though they do not have causal flow [9, 24]. Browne et al. [9] adapted the notion of flow to what they called *generalised flow* (gflow), which is both a necessary and sufficient condition for ‘uniformly and strongly stepwise deterministic’ measurement patterns (defined below). Unlike causal flow, gflow can also be applied to arbitrary labelled open graphs, i.e. it supports measurement patterns with measurements in all three planes. This even more general case is sometimes called *extended gflow*.

**Definition 2.33** ([9, p.5]). The linear map implemented by a measurement pattern for a specific set of measurement outcomes is called a *branch* of the pattern. A pattern is *deterministic* if all branches are equal up to a scalar. A pattern is *strongly deterministic* if all branches are equal up to a global phase. It is *uniformly deterministic* if it is deterministic for any choice of measurement angles. Finally, the pattern is *stepwise deterministic* if any intermediate pattern – resulting from performing some subset of the measurements and their corresponding corrections – is again deterministic.

The existence of gflow implies the uniform, strong and stepwise determinism of any pattern on the open graph (cf. Theorem 2.39 below). Hence, by applying transformations to an open graph that preserve the existence of a gflow, we ensure that the modified open graph still supports a uniformly, strongly and stepwise deterministic pattern. Note that the condition of interest is preservation of the *existence* of gflow, not preservation of the specific gflow itself.

We will now give the formal definitions of (causal) flow and (extended) gflow.

**Definition 2.34** ([17, Definition 2]). Let  $(G, I, O)$  be an open graph. We say  $G$  has (*causal*) *flow* if there exists a map  $f : \overline{O} \rightarrow \overline{I}$  (from measured qubits to prepared qubits) and a strict partial order  $\prec$  over  $V$  such that for all  $u \in \overline{O}$ :

- $u \sim f(u)$
- $u \prec f(u)$
- $u \prec v$  for all neighbours  $v \neq u$  of  $f(u)$ .

When vertices are smaller than a vertex  $v$  in the order  $\prec$ , they are referred to as being ‘behind’ or ‘in the past of’ of  $v$ .

The notion of gflow differs from the above definition of causal flow in two ways. The value of  $f(u)$  is allowed to be a set of vertices instead of a single vertex, so that corrections can be applied to more than one vertex at a time. As a result of this change, the third condition of causal flow is now too strong: requiring that no element of  $f(u)$  is adjacent to any vertex ‘in the past’ of  $u$  would be too restrictive. Since corrections are applied to sets of vertices at a time, it is possible to make use of the following idea: if corrections are simultaneously applied to an even number of neighbours of  $v$ , then there is no net effect on  $v$ . Thus, the second change takes the form of a parity condition: all vertices in the neighbourhood of  $f(u)$  that lie ‘in the past’ of  $u$  are required to be in the even neighbourhood of  $f(u)$ . As a result, net effects of corrections do not propagate into ‘the past’.



Allowing measurements in more than one measurement plane requires further careful adjustment of the parity conditions depending on the measurement plane of the vertex being measured.

**Definition 2.35** ([9, p.7]). Given a graph  $G = (V, E)$ , for any  $K \subseteq V$ , let  $\text{Odd}_G(K) = \{u \in V : |N(u) \cap K| \equiv 1 \pmod{2}\}$  be the *odd neighbourhood* of  $K$  in  $G$ , i.e. the set of vertices having an odd number of neighbours in  $K$ . If the graph  $G$  is clear from context, we simply write  $\text{Odd}(K)$ . The *even neighbourhood* of  $K$  in  $G$ ,  $\text{Even}_G(K)$ , is defined in a similar way;  $\text{Even}_G(K) = \{u \in V : |N(u) \cap K| \equiv 0 \pmod{2}\}$ .

**Definition 2.36** ([9, Definition 3]). A labelled open graph  $(G, I, O, \lambda)$  has generalised flow (or *gflow*) if there exists a map  $g : \overline{O} \rightarrow \mathcal{P}(\overline{I})$  and a partial order  $\prec$  over  $V$  such that for all  $v \in \overline{O}$ :

- (g1) If  $w \in g(v)$  and  $v \neq w$ , then  $v \prec w$ .
- (g2) If  $w \in \text{Odd}(g(v))$  and  $v \neq w$ , then  $v \prec w$ .
- (g3) If  $\lambda(v) = XY$ , then  $v \notin g(v)$  and  $v \in \text{Odd}(g(v))$ .
- (g4) If  $\lambda(v) = XZ$ , then  $v \in g(v)$  and  $v \in \text{Odd}(g(v))$ .
- (g5) If  $\lambda(v) = YZ$ , then  $v \in g(v)$  and  $v \notin \text{Odd}(g(v))$ .

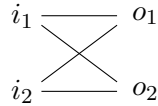
The set  $g(v)$  is called the *correction set* of  $v$ .

**Remark 2.37.** Every causal flow is indeed a gflow, where  $g(v) := \{f(v)\}$  and the partial order remains the same. To see this, first note causal flow can only be defined on labelled open graphs where  $\lambda(v) = XY$  for all  $v \in \overline{O}$ , so conditions (g4) and (g5) are vacuously satisfied. Now, (g1) follows from the second bullet point of Definition 2.34, (g2) follows from the third bullet point, and (g3) follows from the first bullet point.

**Remark 2.38.** In the original definition of gflow in Ref. [9], condition (g2) is given as:

$$\text{if } j \preceq i \text{ and } j \neq i \text{ then } j \notin \text{Odd}(g(i)) \quad (4)$$

In other publications, such as Ref. [19], the definition is changed to the version we give as (g2), yet this is usually done without comment. For completeness, we provide an example which demonstrates that the condition (4) is insufficient for determinism. Consider the following open graph:



Here, the set of inputs is  $\{i_1, i_2\}$ , the set of outputs is  $\{o_1, o_2\}$ , and the non-outputs are measured in the planes  $\lambda(i_1) = \lambda(i_2) = XY$ . If we choose both measurement angles to be 0, it is straightforwardly checked that this diagram implements the linear map:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

This has rank 2 and thus is not invertible, and in particular not unitary. It therefore cannot be deterministically implementable and hence it should not have a gflow. However, it is

considered to have a gflow under condition (4) instead of (g2): pick the partial order  $i_1 \prec o_1, o_2$  and  $i_2 \prec o_1, o_2$  with all other vertices incomparable. Set  $g(i_1) = \{o_1\}$  and  $g(i_2) = \{o_2\}$ . It is then easily checked that  $(g, \prec)$  satisfies conditions (g1), (4), and (g3)–(g5). Yet  $(g, \prec)$  does not satisfy condition (g2) and hence is not a gflow under the revised definition.

To demonstrate that, with condition (g2), the presence of a gflow indeed guarantees determinism of a pattern, we give a detailed proof of the following sufficiency theorem, which was first stated in Ref. [9] as Theorem 2 with a sketch proof. The precise statement of the theorem requires some additional notation and the proof is quite lengthy and technical, so we state a coarse version of the theorem here and refer the reader to Appendix A (and more specifically to Theorem A.5) for the details.

**Theorem 2.39.** Let  $\Gamma = (G, I, O, \lambda)$  be a labelled open graph with a gflow and let  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  be an assignment of measurement angles. Then there exists a runnable measurement pattern which is uniformly, strongly and stepwise deterministic, and which realizes the associated linear map  $M_{\Gamma, \alpha}$  (cf. Definition 2.12).

The converse also holds:

**Theorem 2.40.** If a pattern is stepwise, uniformly and strongly deterministic, then its underlying labelled open graph  $(G, I, O, \lambda)$  has a gflow.

*Proof.* We present the proof sketch here, a complete treatment of the proof can be found in Ref. [19, Theorem 7.9.7]. The proof is by induction on the number of measurements. If the number of measurements is  $n = 0$ , then the pattern trivially has a gflow. Suppose the pattern has  $n + 1$  qubits to be measured. Since the pattern is assumed to be stepwise deterministic, after performing the first measurement, the remaining pattern is still stepwise, uniformly and strongly deterministic. Hence it has a gflow by the induction hypothesis, where the partial order is given by the order in which the measurements are performed.

It remains to extend this gflow to include the first qubit to be measured. The essential part of this is to find a subset  $S \subseteq \bar{I}$  that can act as the correction set of the first measurement (cf. Definition 2.36). Given such a subset  $S$ , we define the full gflow as:

$$g'(i) := \begin{cases} g(i) & i \neq n \\ S & i = n, \end{cases}$$

where  $g$  is the gflow of the smaller pattern. □

It is not straightforward to actually find a concrete gflow from the procedure in this proof. A constructive algorithm has since been given in Ref. [44], which finds gflow on labelled open graphs where all measurements are in the XY plane. In Section 3.2, we extend this algorithm to find extended gflow on labelled open graphs with measurements in all three planes.

Gflow is a property that applies to labelled open graphs. For convenience we also define it for ZX-diagrams.

**Definition 2.41.** We say a ZX-diagram in MQBC(+LC) form has *gflow*  $(g, \prec)$  if the corresponding labelled open graph  $\Gamma$  has gflow  $(g, \prec)$ .

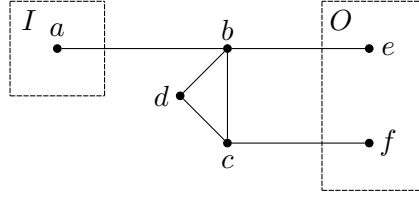
The following result from Ref. [26] shows that any unitary circuit can be converted into a deterministic measurement pattern.

**Proposition 2.42** ([26, Lemma 3.7]). Given a circuit there is a procedure for converting it into an equivalent measurement pattern. Furthermore, this measurement pattern only contains XY-plane measurements and has causal flow, so it also has gflow.

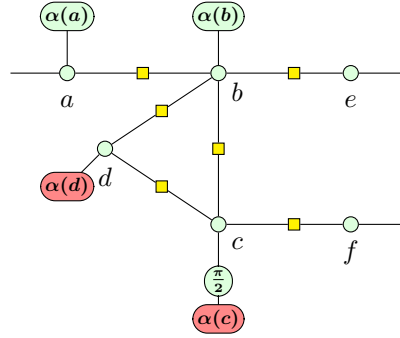
Note that Ref. [26] does not explicitly talk about measurement patterns. What they call graph-like diagrams however correspond in a straightforward manner to diagrams in MBQC form with every measured vertex being measured in the XY-plane. (We also note that the procedure of Proposition 2.42 takes  $O(n^2)$  operations, where  $n$  is the number of vertices.)

Below, we present a concrete example illustrating how the presence of gflow allows measurement errors to be corrected as the computation progresses.

**Example 2.43.** Consider the following labelled open graph  $\Gamma$ :



where  $a$  is an input,  $e$  and  $f$  are outputs, and the measurement planes are given by  $\lambda(a) = \lambda(b) = XY$ ,  $\lambda(c) = XZ$  and  $\lambda(d) = YZ$ . As usual, we denote the measurement angles by  $\alpha : V \rightarrow [0, 2\pi)$ , where  $V$  is the vertex set of  $\Gamma$ . Using Definition 2.14 (that is, we translate according to Table 1), we obtain the corresponding ZX-diagram:



Note that the labelled open graph we started with has a gflow  $(g, \prec)$  given by the following partial order

$$a \prec b \prec c \prec d \prec e, f,$$

with the function  $g$  taking the values

$$\begin{aligned} g(a) &= \{b\} \\ g(b) &= \{c\} \\ g(c) &= \{c, d\} \\ g(d) &= \{d, e, f\}. \end{aligned}$$

It follows that we have

$$\begin{aligned} \text{Odd}(g(a)) &= \{a, c, d, e\} \\ \text{Odd}(g(b)) &= \{b, d, f\} \\ \text{Odd}(g(c)) &= \{c, d, f\} \\ \text{Odd}(g(d)) &= \emptyset, \end{aligned}$$

from which it is easy to verify that the conditions (g1)-(g5) hold.

By Theorem 2.39, the presence of gflow guarantees that we can correct the measurement errors provided that we measure the qubits according to the partial order. We demonstrate this for  $\Gamma$  in Figure 4.

Thus suppose a measurement error of  $\pi$  occurs when performing the measurement corresponding to the vertex  $c$ , as indicated in the top left part of Figure 4. The labels in this figure refer to the rules in Figure 2. In order to get the left diagram on the second row, we move each red  $\pi$ -phase past the corresponding Hadamard gate, which changes the colour of the phase to green. For the right diagram, the left green  $\pi$  travels past the green node on the left and flips the sign of  $\alpha(d)$ . Next, to obtain the left diagram on the third row, the middle green  $\pi$  travels along the middle triangle and past another Hadamard gate to become a red  $\pi$ . Finally, in the bottom left diagram, the red  $\pi$  on the left has been fused with  $-\alpha(d)$ ; and the red  $\pi$  on the right has passed through the Hadamard gate switching its colour to green, and the adjacent green nodes have fused into a node with phase  $\pi - \frac{\pi}{2} = \frac{\pi}{2}$ . We then rearrange the diagram so that it looks like a measurement pattern again.

Note that all the vertices that are affected by the error are above  $c$  in the partial order and hence ‘not yet measured’ at this stage of the computation. Thus the necessary corrections may be applied to these vertices when they are measured.

## 2.5 Focusing gflow for XY plane measurements

For a labelled open graph  $(G, I, O, \lambda)$  in which all measurements are in the XY-plane, there is a special type of gflow which is specified by the correction function alone. This gflow is called *focused* because of the property that, among non-output vertices, corrections only affect the vertex they are meant to correct. A labelled open graph where all measurements are in the XY plane has gflow if and only if it has focused gflow.

**Definition 2.44** (adapted from [44, Definition 5]). Suppose  $(G, I, O, \lambda)$  is a labelled open graph with the property that  $\lambda(v) = \text{XY}$  for all  $v \in \overline{O}$ . Then  $(g, \prec)$  is a *focused gflow* on  $(G, I, O, \lambda)$  if for all  $v \in \overline{O}$ , we have  $\text{Odd}_G(g(v)) \cap \overline{O} = \{v\}$ , and furthermore  $\prec$  is the transitive closure of the relation  $\{(v, w) \mid v \in \overline{O} \wedge w \in g(v)\}$ .

**Theorem 2.45** (reformulation of [44, Theorem 2]). Suppose  $(G, I, O, \lambda)$  is a labelled open graph with the property that  $\lambda(v) = \text{XY}$  for all  $v \in \overline{O}$ , then  $(G, I, O, \lambda)$  has gflow if and only if it has a focused gflow.

A labelled open graph in which all measurements are in the XY-plane can be *reversed* by swapping the roles of inputs and outputs. More formally:

**Definition 2.46.** Suppose  $(G, I, O, \lambda)$  is a labelled open graph with the property that  $\lambda(v) = \text{XY}$  for all  $v \in \overline{O}$ . The corresponding *reversed labelled open graph* is the labelled open graph where the roles of inputs and outputs are swapped, i.e. it is  $(G, O, I, \lambda')$ , where  $\lambda'(v) := \text{XY}$  for all  $v \in \overline{I}$ .

Now if the number of inputs and outputs in the labelled open graph is the same, its focused gflow can also be reversed in the following sense.

**Corollary 2.47.** Suppose  $(G, I, O, \lambda)$  is a labelled open graph with the properties that  $|I| = |O|$  and  $\lambda(v) = \text{XY}$  for all  $v \in \overline{O}$ , and suppose it has a focused gflow  $(g, \prec)$ . For all  $v \in \overline{I}$ , let  $g'(v) := \{w \in \overline{O} \mid v \in g(w)\}$ , and for all  $u, w \in V$ , let  $u \prec' w$  if and only if  $w \prec u$ . Then  $(g', \prec')$  is a focused gflow for the reversed labelled open graph  $(G, O, I, \lambda')$ .

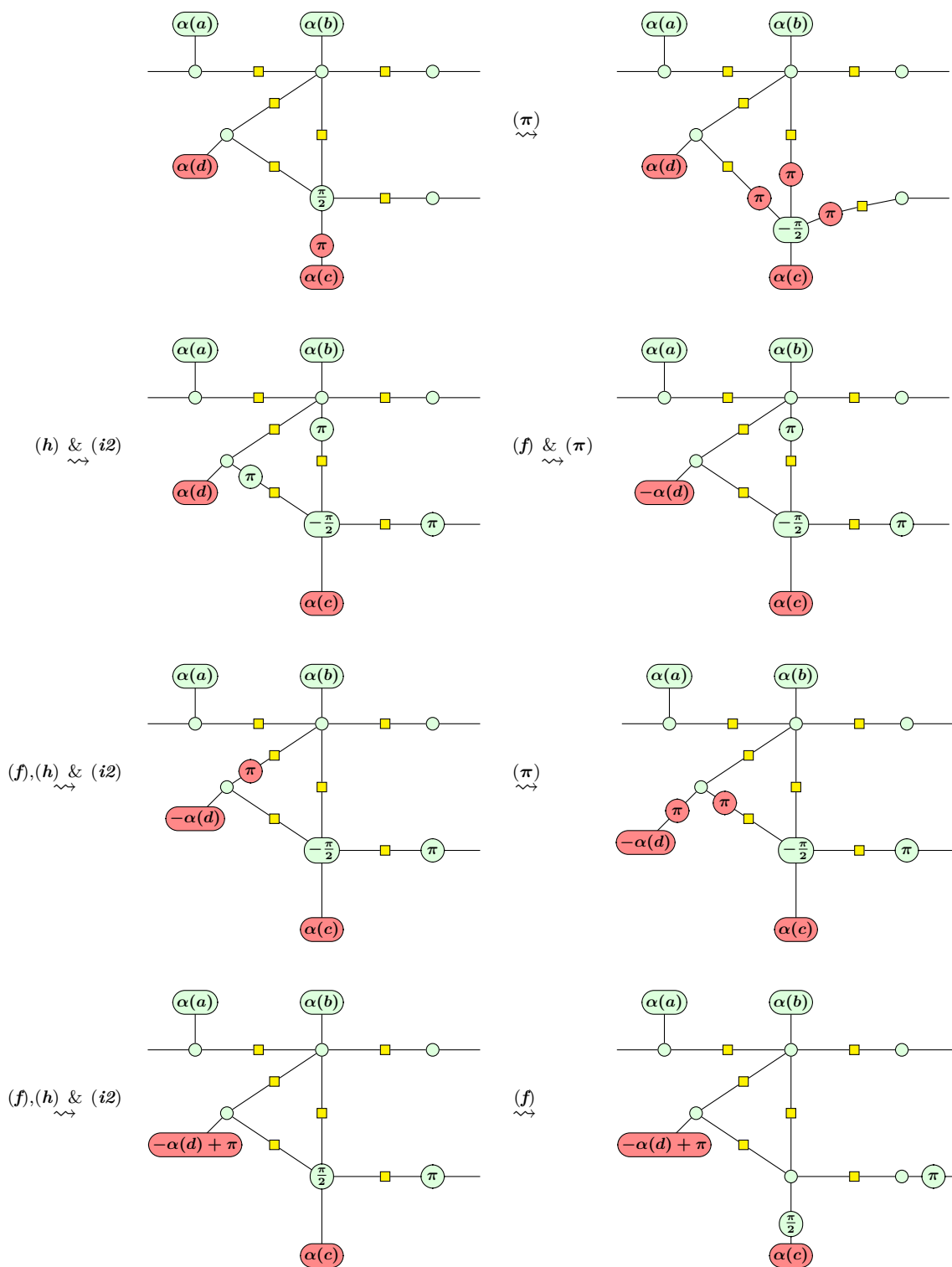


Figure 4: Propagation of a measurement error of the pattern in Example 2.43.

This follows immediately from the proofs of Ref. [44, Theorems 3–4] but it is not explicitly stated in that paper.

### 3 Rewriting while preserving the existence of gflow

In this section, we study a variety of topics dealing with labelled open graphs and gflow. In the first subsection, we show how certain graph operations, such as local complementation and pivoting, affect the gflow. In Section 3.2 we give a polynomial time algorithm for finding extended gflow using the concept of *maximally delayed* gflow. We combine this notion with that of a *focused* extended gflow in Section 3.3 to transform a given gflow to give it certain useful properties.

#### 3.1 Graph operations that preserve the existence of gflow

In this section, we prove some of our main technical lemmas, establishing that local complementation and related graph rewrites interact well with the gflow of the graph.

First, we show that a labelled open graph resulting from the local complementation of a labelled open graph with gflow will also have a gflow.

**Lemma 3.1.** Let  $(g, \prec)$  be a gflow for  $(G, I, O, \lambda)$  and let  $u \in \overline{O} \cap \overline{I}$ . Then  $(g', \prec)$  is a gflow for  $(G \star u, I, O, \lambda')$ , where

$$\lambda'(u) := \begin{cases} \text{XZ} & \text{if } \lambda(u) = \text{XY} \\ \text{XY} & \text{if } \lambda(u) = \text{XZ} \\ \text{YZ} & \text{if } \lambda(u) = \text{YZ} \end{cases}$$

and for all  $v \in \overline{O} \setminus \{u\}$

$$\lambda'(v) := \begin{cases} \text{YZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{XZ} \\ \text{XZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{YZ} \\ \lambda(v) & \text{otherwise.} \end{cases}$$

Furthermore,

$$g'(u) := \begin{cases} g(u) \Delta \{u\} & \text{if } \lambda(u) \in \{\text{XY}, \text{XZ}\} \\ g(u) & \text{if } \lambda(u) = \text{YZ} \end{cases}$$

and for all  $v \in \overline{O} \setminus \{u\}$ ,

$$g'(v) := \begin{cases} g(v) & \text{if } u \notin \text{Odd}_G(g(v)) \\ g(v) \Delta g'(u) \Delta \{u\} & \text{if } u \in \text{Odd}_G(g(v)). \end{cases}$$

The proof of this lemma can be found in Appendix B. Note that the condition that the complemented vertex is not an output can in fact be dropped:

**Lemma 3.2.** Let  $(g, \prec)$  be a gflow for  $(G, I, O, \lambda)$  and let  $u \in O$ . Then  $(g', \prec)$  is a gflow for  $(G \star u, I, O, \lambda')$ , where for all  $v \in \overline{O}$

$$\lambda'(v) := \begin{cases} \text{YZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{XZ} \\ \text{XZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{YZ} \\ \lambda(v) & \text{otherwise.} \end{cases}$$

Furthermore, for all  $v \in \overline{O}$ ,

$$g'(v) := \begin{cases} g(v) & \text{if } u \notin \text{Odd}_G(g(v)) \\ g(v) \Delta \{u\} & \text{if } u \in \text{Odd}_G(g(v)). \end{cases}$$

*Proof.* The proof is basically the same as that of Lemma 3.1 if we take  $g(u)$  and  $g'(u)$  to be empty. The output vertex has no label, so its label does not need to be updated.  $\square$

Now by applying this lemma three times we see that a pivot also preserves the existence of a gflow.

**Corollary 3.3.** Let  $(G, I, O, \lambda)$  be a labelled open graph which has a gflow, and let  $u, v \in \overline{O} \cap \overline{I}$  be connected by an edge. Then  $(G \wedge uv, I, O, \hat{\lambda})$ , where

$$\hat{\lambda}(a) = \begin{cases} YZ & \text{if } \lambda(a) = XY \\ XZ & \text{if } \lambda(a) = XZ \\ XY & \text{if } \lambda(a) = YZ \end{cases}$$

for  $a \in \{u, v\}$ , and  $\hat{\lambda}(w) = \lambda(w)$  for all  $w \in \overline{O} \setminus \{u, v\}$  also has a gflow.

For more details regarding the correctness of this corollary, we refer to Appendix B.

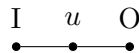
Somewhat surprisingly, the deletion of some types of vertices preserves the existence of gflow:

**Lemma 3.4.** Let  $(g, \prec)$  be a gflow for  $(G, I, O, \lambda)$  and let  $u \in \overline{O}$  with  $\lambda(u) \neq XY$ . Then  $(g', \prec)$  is a gflow for  $(G \setminus \{u\}, I, O, \lambda)$  where  $\forall v \in V, v \neq u$ :

$$g'(v) := \begin{cases} g(v) & \text{if } u \notin g(v) \\ g(v) \Delta g(u) & \text{if } u \in g(v) \end{cases}$$

*Proof.* First, observe that  $u \in g(u)$  since  $\lambda(u) \neq XY$ . Thus  $u \notin g'(v)$  for either case of the definition. Hence,  $g'$  is indeed a function on the graph  $G \setminus u$ . To check that  $g'$  is indeed a gflow we check the necessary conditions for all  $v \in G \setminus \{u\}$ . If  $u \notin g(v)$ , then  $g'(v) = g(v)$  and hence we are done. If  $u \in g(v)$ , then  $v \prec u$  and hence also  $v \prec w$  for all  $w \in g(u)$  or  $w \in \text{Odd}_G(g(u))$ . Since  $g'(v) = g(v) \Delta g(u)$ , conditions (g1) and (g2) are satisfied. For conditions (g3)-(g5), note that we cannot have  $v \in g(u)$  or  $v \in \text{Odd}_G(g(u))$  because  $v \prec u$ . As a result,  $v \in g'(v) \iff v \in g(v)$  and  $v \in \text{Odd}_{G \setminus \{u\}}(g'(v)) \iff v \in \text{Odd}_G(g(v))$ . Since the labels of all the vertices stay the same, (g3)-(g5) remain satisfied.  $\square$

**Remark 3.5.** The condition that  $\lambda(u) \neq XY$  is necessary in the previous lemma. Removing a vertex with label XY will, in general, lead to a labelled open graph which no longer has a gflow. For instance consider the following labelled open graph:



where the first two vertices both have label XY. This graph has a gflow specified by  $I \prec u \prec O$  and  $g(I) = \{u\}$ ,  $g(u) = \{O\}$ , but removing  $u$  will disconnect the graph and hence the resulting graph does not have a gflow.

Note that if we were to consider the same labelled open graph, but with  $u$  measured in a different plane, it would *not* have a gflow to start with. This is because, if it did, we would need  $u \in g(I)$ , so that  $I \prec u$  but also  $u \in g(u)$  so that  $I \in \text{Odd}(g(u))$  giving  $u \prec I$ . Hence this does not contradict the lemma.

The next corollary shows how the previous results can be combined to remove a vertex with arity 2 from a labelled open graph while preserving gflow. In the ZX-calculus, the idea behind this is that we use **(i1)** to remove a vertex and then **(f)** to fuse the adjacent vertices (cf. Definition 2.27). Recall from that definition that  $G \xrightarrow{v} w := ((G \wedge uv) \setminus \{u\}) \setminus \{v\}$ .

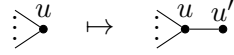
**Corollary 3.6.** Let  $(g, \prec)$  be a gflow for the labelled open graph  $(G, I, O, \lambda)$ , and let  $u, v \in \overline{O}$  and  $w \in V$  be vertices such that  $N_G(v) = \{u, w\}$  and  $\lambda(u), \lambda(v) \neq YZ$ . Then  $(\tilde{g}, \prec)$  as defined below is a gflow for  $(G \xrightarrow{v} w, I, O, \lambda)$ . For all  $z \in \overline{O} \setminus \{u, v\}$  we have

$$\tilde{g}(z) = \begin{cases} \hat{g}(z) & \text{if } u \notin \hat{g}(z), v \notin \hat{g}(z) \\ \hat{g}(z) \Delta \hat{g}(v) & \text{if } u \notin \hat{g}(z) \Delta \hat{g}(v), v \in \hat{g}(z) \\ \hat{g}(z) \Delta \hat{g}(u) \Delta \hat{g}(v) & \text{if } u \in \hat{g}(z) \Delta \hat{g}(v), v \in \hat{g}(z) \Delta \hat{g}(u) \\ \hat{g}(z) \Delta \hat{g}(u) & \text{if } u \in \hat{g}(z), v \notin \hat{g}(z) \Delta \hat{g}(u), \end{cases}$$

where  $\hat{g}$  is as defined in Corollary 3.3.

*Proof.* A computation using Corollary 3.3 and Lemma 3.4.  $\square$

**Lemma 3.7.** Let  $\Gamma = (G, I, O, \lambda)$  be a labelled open graph with  $G = (V, E)$ . Let  $\Gamma'$  be the labelled open graph that results from converting an output  $u \in O$  into a vertex measured in the XY-plane and adding a new output vertex  $u'$  in its stead:



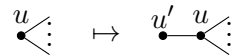
Formally, let  $\Gamma' = (G', I, O', \lambda')$ , where  $G' = (V', E')$  with  $V' = V \cup \{u'\}$  and  $E' = E \cup \{u \sim u'\}$ ,  $O' = (O \setminus \{u\}) \cup \{u'\}$ , and  $\lambda'(v)$  is the extension of  $\lambda$  to domain  $V' \setminus O'$  with  $\lambda'(u) = XY$ . Then if  $\Gamma$  has gflow,  $\Gamma'$  also has gflow.

*Proof.* Suppose  $\Gamma$  has gflow  $(g, \prec)$ . Let  $g'$  be the extension of  $g$  to domain  $V' \setminus O'$  which satisfies  $g'(u) = \{u'\}$ , and let  $\prec'$  be the transitive closure of  $\prec \cup \{(u, u')\}$ .

The tuple  $(g', \prec')$  inherits **(g1)** and **(g3)**–**(g5)** for all  $v \in V \setminus O$  because the correction sets have not changed for any of the original vertices. Furthermore,  $u' \in \text{Odd}_{G'}(g'(v))$  for any  $v$  implies  $u \in g'(v)$ , as  $u$  is the only neighbour of  $u'$ . Hence  $u' \in \text{Odd}_{G'}(g'(v))$  implies  $v \prec' u \prec' u'$ . Therefore **(g2)** continues to be satisfied for all  $v \in V \setminus O$ .

Now, for  $u$ , **(g1)** holds because  $u \prec' u'$  by definition, **(g2)** holds because  $\text{Odd}_{G'}(g'(u)) = \{u\}$ , and **(g3)** can easily be seen to hold. Thus,  $(g', \prec')$  is a gflow for  $\Gamma'$ .  $\square$

**Lemma 3.8.** Let  $\Gamma = (G, I, O, \lambda)$  be a labelled open graph with  $G = (V, E)$ . Let  $\Gamma'$  be the labelled open graph that results from adding an additional vertex measured in the XY-plane ‘before’ the input  $u \in I$ :



Formally, let  $\Gamma' = (G', I', O, \lambda')$ , where  $G' = (V', E')$  with  $V' = V \cup \{u'\}$  and  $E' = E \cup \{u \sim u'\}$ ,  $I' = (I \setminus \{u\}) \cup \{u'\}$ , and  $\lambda'(v)$  is the extension of  $\lambda$  to domain  $V' \setminus O$  which satisfies  $\lambda'(u') = XY$ . Then if  $\Gamma$  has gflow,  $\Gamma'$  also has gflow.

*Proof.* Suppose  $\Gamma$  has gflow  $(g, \prec)$ . Let  $g'$  be the extension of  $g$  to domain  $V' \setminus O$  which satisfies  $g'(u') = \{u\}$ , and let  $\prec'$  be the transitive closure of  $\prec \cup \{(u', w) : w \in N_G(u) \cup \{u\}\}$ .

The tuple  $(g', \prec')$  inherits the gflow properties for all  $v \in V \setminus O$  because the correction sets have not changed for any of the original vertices and because the additional inequalities in  $\prec'$  do not affect the gflow properties for any  $v \in V \setminus O$ . The latter is because



- $u' \notin g'(v) = g(v)$  for any  $v \in V \setminus O$ , and
- $u' \notin \text{Odd}_{G'}(g'(v)) = \text{Odd}_{G'}(g(v))$  for any  $v \in V \setminus O$  since its only neighbour  $u$  was an input in  $\Gamma$  and thus satisfies  $u \notin g(v)$  for any  $v \in V \setminus O$ .

Now, for  $u'$ , (g1) holds by the definition of  $\prec'$ . Note that  $\text{Odd}_{G'}(g(u')) = N_{G'}(u)$ , so (g2) also holds by the definition of  $\prec'$ . Finally, (g3) holds because  $u' \notin g(u')$  and  $u' \in \text{Odd}_{G'}(g(u')) = N_{G'}(u)$ . Thus,  $(g', \prec')$  is a gflow for  $\Gamma'$ .  $\square$

### 3.2 Finding extended gflow

Ref. [42] gives a polynomial time algorithm for finding gflow for labelled open graphs with all measurements in the XY-plane. In this section, we present an extension of this algorithm that works for measurements in all three measurement planes.

Before doing so, we note a few details from the algorithm. The intuition behind the procedure is to ‘maximally delay’ any measurements, thereby keeping potential correction options available for as long as possible. As a result, the algorithm finds a gflow of minimal ‘depth’ (a notion we will make precise later).

The algorithm works backwards: Starting from the output vertices, it iteratively constructs disjoint subsets of vertices that can be corrected by vertices chosen in previous steps. Viewed instead as information travelling forwards, from the inputs to the outputs, this corresponds to only correcting a vertex at the last possible moment, hence the name maximal delayed.

**Definition 3.9** (Generalisation of [42, Definition 4] to multiple measurement planes). For a given labelled open graph  $(G, I, O, \lambda)$  and a given gflow  $(g, \prec)$  of  $(G, I, O, \lambda)$ , let

$$V_k^\prec = \begin{cases} \max_\prec(V) & \text{if } k = 0 \\ \max_\prec(V \setminus (\bigcup_{i < k} V_i^\prec)) & \text{if } k > 0 \end{cases}$$

where  $\max_\prec(X) := \{u \in X \text{ s.t. } \forall v \in X, \neg(u \prec v)\}$  is the set of the maximal elements of  $X$ .

**Definition 3.10** (Generalisation of [42, Definition 5] to multiple measurement planes). For a given labelled open graph  $(G, I, O, \lambda)$  and two given gflows  $(g, \prec)$  and  $(g', \prec')$  of  $(G, I, O, \lambda)$ ,  $(g, \prec)$  is *more delayed* than  $(g', \prec')$  if for all  $k$ ,

$$\left| \bigcup_{i=0}^k V_i^\prec \right| \geq \left| \bigcup_{i=0}^k V_i^{\prec'} \right|$$

and there exists a  $k$  such that the inequality is strict. A gflow  $(g, \prec)$  is *maximally delayed* if there exists no gflow of the same open graph that is more delayed.

**Theorem 3.11** (Generalisation of [42, Theorem 2] to multiple measurement planes). There exists a polynomial time algorithm that decides whether a given labelled open graph has an extended gflow, and that outputs such a gflow if it exists. Moreover, the output gflow is maximally delayed.

The proof of this theorem can be found in Appendix C. The algorithm we find is a relatively straightforward generalisation of that of Ref. [42], but the proofs required to show its correctness are complicated somewhat by the case distinctions required for the three measurement planes. We note also that our detailed proof shows that the original proofs of Ref. [42] missed some case distinctions and conditions.

### 3.3 Focusing extended gflow

In Section 2.5, we introduced the notion of focused gflow for labelled open graphs in which all measurements are in the XY plane. There is no canonical generalisation of this notion to labelled open graphs with measurements in multiple planes. Hamrit and Perdrix suggest three different extensions of focused gflow to the case of multiple measurement planes, which restrict correction operations on non-output qubits to only a single type of Pauli operator overall [29, Definition 2]. Here, we go a different route by requiring that non-output qubits only appear in correction sets, or odd neighbourhoods of correction sets, if they are measured in specific planes. This means the correction operators which may be applied to some non-output qubit depend on the plane in which that qubit is measured. The new notion of focused gflow will combine particularly nicely with the phase-gadget form of MBQC+LC diagrams of Section 4.4.

We begin by proving some lemmas that allow any gflow to be focused in our sense.

**Lemma 3.12.** Let  $(G, I, O, \lambda)$  be a labelled open graph which has gflow  $(g, \prec)$ . Suppose there exist  $v, w \in \bar{O}$  such that  $v \prec w$ . Define  $g'(v) := g(v) \Delta g(w)$  and  $g'(u) := g(u)$  for all  $u \in \bar{O} \setminus \{v\}$ , then  $(g', \prec)$  is a gflow.

*Proof.* As the correction set only changes for  $v$ , the gflow properties remain satisfied for all other vertices. Now, suppose  $w' \in g'(v)$ , then  $w' \in g(v) \vee w' \in g(w)$ . In the former case,  $v \prec w'$ , and in the latter case,  $v \prec w \prec w'$ , since  $(g, \prec)$  is a gflow, so (g1) holds. Similarly, suppose  $w' \in \text{Odd}(g'(v))$ , then by linearity of  $\text{Odd}(\cdot)$  we have  $w' \in \text{Odd}(g(v)) \vee w' \in \text{Odd}(g(w))$ . Again, this implies  $v \prec w'$  or  $v \prec w \prec w'$  since  $(g, \prec)$  is a gflow, so (g2) holds. Finally,  $v \prec w$  implies  $v \notin g(w)$  and  $v \notin \text{Odd}(g(w))$  since  $(g, \prec)$  is a gflow. Therefore  $v \in g'(v) \iff v \in g(v)$  and  $v \in \text{Odd}(g'(v)) \iff v \in \text{Odd}(g(v))$ . Thus (g3)–(g5) hold and  $(g', \prec)$  is a gflow.  $\square$

**Lemma 3.13.** Let  $(G, I, O, \lambda)$  be a labelled open graph, let  $(g, \prec)$  be a gflow for this open graph, and let  $v \in \bar{O}$ . Then there exists  $g' : \bar{O} \rightarrow \mathcal{P}(\bar{I})$  such that

1. for all  $w \in \bar{O}$ , either  $v = w$  or  $g'(w) = g(w)$ ,
2. for all  $w \in g'(v) \cap \bar{O}$ , either  $v = w$  or  $\lambda(w) = \text{XY}$ ,
3. for all  $w \in \text{Odd}(g'(v)) \cap \bar{O}$ , either  $v = w$  or  $\lambda(w) \neq \text{XY}$ , and
4.  $(g', \prec)$  is a gflow for  $(G, I, O, \lambda)$ .

We can construct this  $g'$  in a number of steps that is polynomial in the number of vertices of  $G$ .

*Proof.* Let  $g_0 := g$ , we will modify the function in successive steps to  $g_1, g_2$ , and so on. For each non-negative integer  $k$  in turn, define

$$\begin{aligned} S_{k,\text{XY}} &:= \{u \in (\text{Odd}(g_k(v)) \cap \bar{O}) \setminus \{v\} : \lambda(u) = \text{XY}\}, \\ S_{k,\text{XZ}} &:= \{u \in (g_k(v) \cap \bar{O}) \setminus \{v\} : \lambda(u) = \text{XZ}\}, \\ S_{k,\text{YZ}} &:= \{u \in (g_k(v) \cap \bar{O}) \setminus \{v\} : \lambda(u) = \text{YZ}\}, \end{aligned}$$

and set  $S_k := S_{k,\text{XY}} \cup S_{k,\text{XZ}} \cup S_{k,\text{YZ}}$ . Finding  $S_k$  takes  $O(|V|^2)$  operations. If  $S_k = \emptyset$ , let  $g' := g_k$  and stop. Otherwise, choose  $w_k \in S_k$  among the elements minimal in  $\prec$ , and define

$$g_{k+1}(u) := \begin{cases} g_k(v) \Delta g_k(w_k) & \text{if } u = v \\ g_k(u) & \text{otherwise.} \end{cases}$$

Note  $w_k \in S_k$  implies  $w_k \neq v$ , as well as either  $w_k \in g_k(v)$  or  $w_k \in \text{Odd}(g_k(v))$ . Thus if  $(g_k, \prec)$  is a gflow, then  $v \prec w_k$ , and hence by Lemma 3.12,  $(g_{k+1}, \prec)$  is also a gflow. Since  $(g_0, \prec)$  is a gflow, this means  $(g_k, \prec)$  is a gflow for all  $k$ .

Now, if  $w_k \in S_{k,XY}$ , then  $w_k \in \text{Odd}(g_k(w_k))$  by (g3). This implies  $w_k \notin \text{Odd}(g_{k+1}(v))$ , and thus  $w_k \notin S_{k+1}$ . Similarly, if  $w_k \in S_{k,XZ} \cup S_{k,YZ}$ , then  $w_k \in g_k(w_k)$  by (g4) or (g5). This implies  $w_k \notin g_{k+1}(v)$ , and thus  $w_k \notin S_{k+1}$ . Hence, in each step we remove a minimal element from the set.

Suppose there exists  $w' \in S_{k+1} \setminus S_k$ , then either  $w' \in g_k(w_k)$  or  $w' \in \text{Odd}(g_k(w_k))$ ; in either case  $w_k \prec w'$ . In other words, we always remove a minimal element from the set and add only elements that come strictly later in the partial order. Therefore, the process terminates after  $n \leq |V|$  steps, at which point  $S_n = \emptyset$ , and the process requires  $O(|V|^2)$  operations at each step. The total complexity is therefore  $O(|V|^3)$ . The function  $g' = g_n$  has the desired properties: (1) holds because we never modify the value of the function on inputs other than  $v$ , (2) and (3) hold because  $S_n = \emptyset$ , and (4) was shown to follow from Lemma 3.12.  $\square$

Based on these lemmas, we can now show the focusing property. These results state that correction sets can be simplified to only contain qubits measured in the XY plane. Moreover, side-effects of corrections (i.e. effects on qubits other than the one being corrected) never affect qubits measured in the XY plane.

**Proposition 3.14.** Let  $(G, I, O, \lambda)$  be a labelled open graph which has gflow. Then  $(G, I, O, \lambda)$  has a maximally delayed gflow  $(g, \prec)$  with the following properties for all  $v \in V$ :

- for all  $w \in g(v) \cap \overline{O}$ , either  $v = w$  or  $\lambda(w) = XY$ , and
- for all  $w \in \text{Odd}(g(v)) \cap \overline{O}$ , either  $v = w$  or  $\lambda(w) \neq XY$ .

This maximally delayed gflow can be constructed in a number of steps that is polynomial in the number of vertices in  $G$ .

*Proof.* Let  $(g_0, \prec)$  be a maximally delayed gflow of  $(G, I, O, \lambda)$ . Set  $n := |V|$  and consider the vertices in some order  $v_1, \dots, v_n$ . For each  $k = 1, \dots, n$ , let  $g_k$  be the function that results from applying Lemma 3.13 to the gflow  $(g_{k-1}, \prec)$  and the vertex  $v_k$ . Then  $g_k$  satisfies the two properties for the vertex  $v_k$ . The function  $g_k$  also equals  $g_{k-1}$  on all inputs other than  $v_k$ , so in fact  $g_k$  satisfies the two properties for all vertices  $v_1, \dots, v_k$ . Thus,  $g_n$  satisfies the two properties for all vertices. Moreover, the partial order does not change, so  $(g_n, \prec)$  is as delayed as  $(g_0, \prec)$ ; i.e. it is maximally delayed. Hence if  $g := g_n$ , then  $(g, \prec)$  has all the desired properties. The construction of each successive  $g_{k+1}$  via Lemma 3.13 takes  $O(n^3)$  operations, which we perform at most  $n$  times, giving a complexity of  $O(n^4)$ .  $\square$

The extended notion of focused gflow also allows us to prove another result which will be useful for the optimisation algorithm later.

First, note that if a labelled open graph has gflow, then the labelled open graph that results from deleting all vertices measured in the XZ or YZ planes still has gflow.

**Lemma 3.15.** Suppose  $(G, I, O, \lambda)$  is a labelled open graph which has gflow. Let  $(G', I, O, \lambda')$  be the induced labelled open graph on the vertex set  $V' = \{v \in V \mid v \in O \text{ or } \lambda(v) = XY\}$ . Then  $(G', I, O, \lambda')$  has gflow.

*Proof.* Apply Lemma 3.4 to each vertex measured in the XZ or YZ plane one by one. Recall from Definition 2.36 that input vertices are measured in the XY plane and so are not removed by this process.  $\square$

We can now show that in a labelled open graph which has gflow and which satisfies  $|I| = |O|$ , any internal XY vertex must have more than one neighbour.<sup>4</sup>

**Proposition 3.16.** Let  $(G, I, O, \lambda)$  be a labelled open graph which has gflow and for which  $|I| = |O|$ . Suppose  $v \in \bar{O}$  satisfies  $\lambda(v) = XY$ . Then either  $v \in I$  and  $|N_G(v)| \geq 1$ , or  $v \notin I$  and  $|N_G(v)| \geq 2$ .

*Proof.* Consider some  $v \in \bar{O}$  such that  $\lambda(v) = XY$ . Note that a vertex with no neighbours is in the even neighbourhood of any set of vertices. Therefore we must have  $|N_G(v)| \geq 1$ , since  $(G, I, O, \lambda)$  has gflow and  $v$  must be in the odd neighbourhood of its correction set by (g3).

Now suppose for a contradiction that  $v \notin I$  and  $|N_G(v)| = 1$ . Denote by  $u$  the single neighbour of  $v$ .

If the labelled open graph contains any vertices measured in the XZ or YZ planes, by Lemma 3.15, we can remove those vertices while preserving the property of having gflow. Since  $\lambda(v) = XY$ , the removal process preserves  $v$ . The new labelled open graph has gflow and  $v$  cannot have gained any new neighbours, so  $u$  must also be preserved by the argument of the first paragraph above. Thus, without loss of generality, we will assume that all non-outputs of  $(G, I, O, \lambda)$  are measured in the XY plane.

The labelled open graph  $(G, I, O, \lambda)$  has gflow and satisfies  $\lambda(w) = XY$  for all  $w \in \bar{O}$ , so by Theorem 2.45 it has a focused gflow  $(g, \prec)$ . To satisfy the gflow condition (g3) for  $v$ , that is, to satisfy  $v \in \text{Odd}_G(g(v))$ , we must have  $u \in g(v)$ . This then implies  $v \prec u$  by (g1).

Since  $|I| = |O|$ , the focused gflow  $(g, \prec)$  can be reversed in the sense of Corollary 2.47. Denote by  $(G, O, I, \lambda')$  the reversed labelled open graph (cf. Definition 2.46) and by  $(g', \prec')$  the corresponding reversed focused gflow. Since  $v \notin I$ ,  $v$  remains a non-output in the reversed graph, so it has a correction set. But  $g'(v) = \{w \in \bar{O} \mid v \in g(w)\}$ , so it cannot contain  $u$  because  $v \notin g(u)$  by  $v \prec u$ . Thus,  $v \notin \text{Odd}_G(g'(v))$ , contradicting (g3).

Therefore, the initial assumption must be wrong, i.e. if  $v \notin I$  then  $|N_G(v)| \geq 2$ .  $\square$

**Remark 3.17.** This implies that in any unitary MBQC-form ZX-diagram with gflow, any vertex measured in the XY-plane has at least two incident wires (plus the wire leading to the measurement effect), since being an input vertex of the labelled open graph implies being connected to an input wire in the ZX-diagram.

## 4 Simplifying measurement patterns

In the previous section, we saw several ways in which labelled open graphs can be modified while preserving the existence of gflow. In this section, we will see how these modifications can be done on measurement patterns in a way that preserves the computation being performed. The goal of the simplifications in this section is to reduce the number of qubits needed to implement the computation. Since we are specifically interested in patterns with

---

<sup>4</sup>The condition  $|I| = |O|$  is necessary: consider the labelled open graph  $(G, \emptyset, \{o\}, \lambda)$ , where  $G$  is the connected graph on two vertices  $\{v, o\}$ , and  $\lambda(v) = XY$ . Then  $v$  is internal and has only a single neighbour, yet the labelled open graph has gflow with  $g(v) = \{o\}$  and  $v \prec o$ .

gflow, we will represent a pattern by a ZX-diagram in MBQC+LC form, which carries essentially the same information.



We show how local complementations affect a measurement pattern. The relation between local complementations and graph states was first considered in Ref. [50], and the combined action of a local complementation or pivot followed by a vertex deletion on a measurement pattern consisting of just XY-plane measurements was considered in Ref. [26]. We here show the effect of a local complementation on a generic measurement pattern containing measurements in three possible planes. We combine these results with ways to delete qubits measured in a Clifford angle. That qubits measured in a Clifford angle can be treated in special ways is well-known. For instance, Hein et al. showed how graph states are changed under Pauli measurements [30]. More specific to MBQC, Browne et al. found gflow conditions specific to Pauli observables [9] and Di Matteo and Mosca showed that all Clifford measurements (and their corrections) in a pattern containing only XY-plane measurements can be done simultaneously, reducing the depth of a computation [22]. Our results go beyond this in three ways. First, by explicitly showing how to find an equivalent measurement pattern where Clifford qubits are actually removed, instead of just ‘compressed’ into a single layer. Second, by doing this for measurement patterns containing measurements in three planes instead of just one, and third, by showing how these operations preserve the existence of a gflow, and hence preserve deterministic realisability. This last point was also considered in Ref. [26], restricted to XY-plane measurements, where they showed this property for the combination of a local complementation and a qubit deletion. This result actually becomes simpler for generic patterns, as the local complementation and deletion can be done separately.

Before we find qubit-removing rewrite rules however, we establish how local Cliffords in an MBQC+LC form diagram can be changed into measurements in Section 4.1 and how local complementations affect a pattern in Section 4.2. We use local complementations to remove Clifford vertices from a pattern in Section 4.3, and to change a pattern so that only two measurement planes are necessary in Section 4.4. Finally, in Section 4.5 we find some further simplifications that allow the removal of additional qubits.

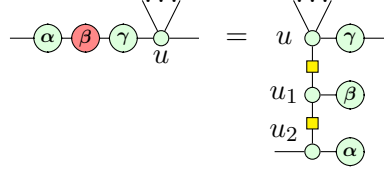
#### 4.1 Transforming local Cliffords into measurements

We used MBQC+LC diagrams as an extension of MBQC form diagrams. In this section we will see that we can always convert the local Clifford gates into measurements to turn the diagram into MBQC form.

**Lemma 4.1.** Any ZX-diagram  $D$  which is in MBQC+LC form can be brought into MBQC form. Moreover, if the MBQC-form part of  $D$  involves  $n$  qubits, of which  $p$  are inputs and  $q$  are outputs, then the resulting MBQC-form diagram contains at most  $n + 2p + 4q$  qubits.

*Proof.* Any single-qubit Clifford unitary can be expressed as a composite of three phase shifts [4, Lemma 3]. Note that this result holds with either choice of colours, i.e. any single-qubit Clifford unitary can be expressed as  or .

Now, with the green-red-green version, for any Clifford operator on an input, we can ‘push’ the final green phase shift through the graph state part onto the outgoing wire. There, it will either merge with the measurement effect or with the output Clifford unitary:



If  $\gamma \in \{0, \pi\}$ , merging the phase shift with a measurement effect may change the angle but not the phase label, e.g. if  $\gamma = \pi$ :

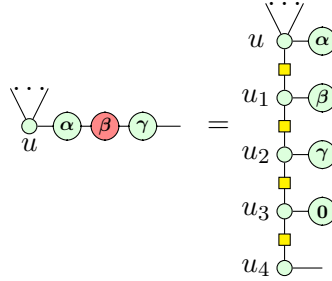
$$\text{---} \circ_{\pi} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{\alpha + \pi} \text{---} \quad \text{---} \circ_{\pi} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{-\alpha} \text{---} \quad \text{---} \circ_{\pi} \circ_{\frac{\pi}{2}} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{\frac{\pi}{2}} \text{---} \circ_{-\alpha} \text{---}$$

If  $\gamma \in \{\frac{\pi}{2}, -\frac{\pi}{2}\}$ , merging the phase shift with a measurement effect will flip the phase labels XZ and YZ, e.g. if  $\gamma = -\frac{\pi}{2}$ :

$$\text{---} \circ_{-\frac{\pi}{2}} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{\alpha - \frac{\pi}{2}} \text{---} \quad \text{---} \circ_{-\frac{\pi}{2}} \circ_{\frac{\pi}{2}} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{0} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{\alpha} \text{---} \quad \text{---} \circ_{-\frac{\pi}{2}} \text{---} \circ_{\alpha} \text{---} = \text{---} \circ_{\frac{\pi}{2}} \text{---} \circ_{-\alpha} \text{---}$$

Thus we need to add at most two new qubits to the MBQC-form part when removing a Clifford unitary on the input.

For a Clifford unitary on the output, we have



Thus we add at most four new qubits.

Combining these properties, we find that rewriting to MBQC form adds at most  $2p+4q$  new qubits to the pattern.  $\square$

**Proposition 4.2.** Suppose  $D$  is a ZX-diagram in MBQC+LC form and that its MBQC part has gflow. Let  $D'$  be the ZX-diagram that results from bringing  $D$  into MBQC form as in Lemma 4.1. Then  $D'$  has gflow.

*Proof.* By applying Lemma 4.1 repeatedly, we can incorporate any local Clifford operators into the MBQC form part of the diagram. Lemmas 3.7 and 3.8 ensure that each step preserves the property of having gflow.  $\square$

## 4.2 Local complementation and pivoting on patterns

Lemmas 2.31 and 2.32 showed how to apply a local complementation and pivot on a ZX-diagram by introducing some local Clifford spiders. In this section we will show how these rewrite rules can be used on MBQC+LC diagrams.

**Lemma 4.3.** Let  $D$  be an MBQC+LC diagram and suppose  $u \in G(D)$  is not an input vertex. Then the diagram resulting from applying Lemma 2.31 on  $u$  (i.e. locally complementing), can be turned back into an MBQC+LC diagram  $D'$  with  $G(D') = G(D) \star u$ . If  $D$  had gflow, then  $D'$  will also have gflow.

*Proof.* Suppose  $D$  is an MBQC+LC diagram,  $\Gamma = (G, I, O, \lambda)$  the corresponding labelled open graph, and  $\alpha : \overline{O} \rightarrow [0, 2\pi)$  gives the associated measurement angles. By assumption,  $u \notin I$ , so – with the exception of the output wire or the edge to the measurement effect – all edges incident on  $u$  connect to neighbouring vertices in the graph. The input wires

on the other qubits can be safely ignored. To get back an MBQC+LC diagram after Lemma 2.31 is applied to  $u$ , we only need to rewrite the measurement effects, and hence we need to construct new  $\lambda'$  and  $\alpha'$  for these measurement effects. We do that as follows.

First of all, there are no changes to the measurement effects on vertices  $v \notin N(u) \cup \{u\}$ , and hence for those vertices we have  $\lambda'(v) = \lambda(v)$  and  $\alpha'(v) = \alpha(v)$ .

The vertex  $u$  gets a red  $\frac{\pi}{2}$  phase from the application of Lemma 2.31. If  $u \in O$ , then it has no associated measurement plane or angle. In this case, this red  $\frac{\pi}{2}$  simply stays on the output wire, as allowed in an MBQC+LC diagram. When  $u \notin O$ , there are three possibilities, depending on  $\lambda(u)$ :

- If  $\lambda(u) = XY$ , then the new measurement effect is

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

i.e.  $\lambda'(u) = XZ$  and  $\alpha'(u) = \frac{\pi}{2} - \alpha(u)$ .

- If  $\lambda(u) = XZ$ , then the new measurement effect is

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

i.e.  $\lambda'(u) = XY$  and  $\alpha'(u) = \alpha(u) - \frac{\pi}{2}$ .

- If  $\lambda(u) = YZ$ , then the new measurement effect is

$$\text{---} \left( \frac{\pi}{2} \right) \alpha \text{---} = \text{---} \left( \alpha + \frac{\pi}{2} \right) \text{---}$$

i.e.  $\lambda'(u) = YZ$  and  $\alpha'(u) = \alpha(u) + \frac{\pi}{2}$ .

The vertices  $v$  that are neighbours of  $u$  get a green  $-\frac{\pi}{2}$  phase. Again, if such a  $v$  is an output, this phase can be put as a local Clifford on the output. If it is not an output, then there are also three possibilities depending on  $\lambda(v)$ :

- If  $\lambda(v) = XY$ , then the new measurement effect is

$$\text{---} \left( -\frac{\pi}{2} \right) \alpha \text{---} = \text{---} \left( \alpha - \frac{\pi}{2} \right) \text{---}$$

i.e.  $\lambda'(v) = XY$  and  $\alpha'(v) = \alpha(v) - \frac{\pi}{2}$ .

- If  $\lambda(v) = XZ$ , then the new measurement effect is

$$\text{---} \left( -\frac{\pi}{2} \right) \left( \frac{\pi}{2} \right) \alpha \text{---} = \text{---} 0 \alpha \text{---} = \text{---} \alpha \text{---}$$

i.e.  $\lambda'(v) = YZ$  and  $\alpha'(v) = \alpha(v)$ .

- If  $\lambda(v) = YZ$ , then the new measurement effect is

$$\text{---} \left( -\frac{\pi}{2} \right) \alpha \text{---} = \text{---} \left( \frac{\pi}{2} \right) -\alpha \text{---}$$

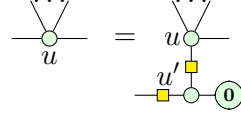
i.e.  $\lambda'(v) = XZ$  and  $\alpha'(v) = -\alpha(v)$ .

With these changes, we see that the resulting diagram  $D'$  is indeed in MBQC+LC form. The underlying graph  $G(D')$  results from the local complementation about  $u$  of the original graph  $G(D)$ . Furthermore, the measurement planes changed in the same way as in Lemma 3.1, and hence if  $D$  had gflow, then  $D'$  will also have gflow.  $\square$

**Proposition 4.4.** Let  $D$  be an MBQC+LC diagram and suppose  $u$  is an arbitrary vertex. Then we can find an MBQC+LC diagram  $D'$  such that  $D'$  is equivalent to  $D$  and  $G(D) \star u \subseteq G(D')$ . Furthermore, if  $D$  has gflow, then  $D'$  does too.

*Proof.* If  $u$  is not an input vertex, the result is immediate from Lemma 4.3.

If instead  $u$  is an input vertex, we modify  $D$  by replacing the input wire incident on  $u$  by an additional graph vertex  $u'$  measured in the XY-plane at angle 0, and a Hadamard unitary on the input wire:



Throughout this process, the measurement effect on  $u$  (if any) does not change, so it is left out of the above equation. In the resulting diagram  $D''$ ,  $u$  is no longer an input. It is straightforward to see that  $D''$  is equivalent to  $D$ . Furthermore,  $D''$  is an MBQC+LC diagram with  $G(D) \subseteq G(D'')$ , and  $D''$  has gflow if  $D$  does. Thus, the desired result follows by applying Lemma 4.3 to  $D''$  and noting that local complementations commute with subgraph relationships.  $\square$

A pivot is just a sequence of three local complementations. Thus, the previous lemma already implies that when Lemma 2.32 is applied to an MBQC+LC diagram the resulting diagram can also be brought back into MBQC+LC form. Nevertheless, it will be useful to explicitly write out how the measurement planes of the vertices change.

**Lemma 4.5.** Let  $D$  be an MBQC+LC diagram and suppose  $u$  and  $v$  are neighbouring vertices in the graph state and are not input vertices of the underlying labelled open graph. Then the diagram resulting from applying Lemma 2.32 to  $u$  and  $v$  (i.e. a pivot about  $u \sim v$ ) can be brought back into MBQC+LC form. The resulting ZX-diagram  $D'$  satisfies  $G(D') = G(D) \wedge uv$ . If  $D$  had gflow, then  $D'$  will also have gflow.

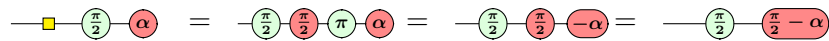
*Proof.* Suppose  $\Gamma = (G, I, O, \lambda)$  is the labelled open graph underlying  $D$  and suppose  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  gives the measurement angles. We will denote the measurement planes after pivoting by  $\lambda' : \bar{O} \rightarrow \{XY, XZ, YZ\}$  and the measurement angles after pivoting by  $\alpha' : \bar{O} \rightarrow [0, 2\pi)$ . Let  $a \in \{u, v\}$ , then:

- If  $a$  is an output, we consider the Hadamard resulting from the pivot operation as a Clifford operator on the output.
- If  $\lambda(a) = XY$  then  $\lambda'(a) = YZ$  and if  $\lambda(a) = YZ$  then  $\lambda'(a) = XY$ :



In both cases, the measurement angle stays the same:  $\alpha'(a) = \alpha(a)$ .

- If  $\lambda(a) = XZ$ , then



i.e.  $\lambda'(a) = XZ$  and  $\alpha'(a) = \frac{\pi}{2} - \alpha(a)$ .



The only other changes are new green  $\pi$  phases on vertices  $w \in N(u) \cap N(v)$ . For measured (i.e. non-output) vertices, these preserve the measurement plane and are absorbed into the measurement angle in all three cases:

$$(\lambda'(w), \alpha'(w)) = \begin{cases} (XY, \alpha(w) + \pi) & \text{if } \lambda(w) = XY \\ (YZ, -\alpha(w)) & \text{if } \lambda(w) = YZ \\ (XZ, -\alpha(w)) & \text{if } \lambda(w) = XZ \end{cases} \quad \begin{array}{l} \text{---} \pi \text{---} \alpha \text{---} = \text{---} \alpha + \pi \text{---} \\ \text{---} \pi \text{---} \frac{\pi}{2} \text{---} \alpha \text{---} = \text{---} \frac{\pi}{2} \text{---} -\alpha \text{---} \\ \text{---} \pi \text{---} \alpha \text{---} = \text{---} -\alpha \text{---} \end{array}$$

If instead  $w$  is an output vertex, we consider the green  $\pi$  phase shift as a Clifford operator on the output wire. The measurement planes and the graph change exactly like in Corollary 3.3 and hence  $D'$  has gflow when  $D$  had gflow.  $\square$

### 4.3 Removing Clifford vertices

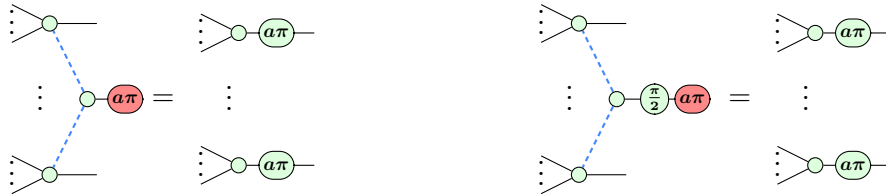
In this section, we show that if a qubit is measured in one of the Pauli bases, i.e. at an angle which is an integer multiple of  $\frac{\pi}{2}$ , it can be removed from a pattern while preserving the semantics as well as the property of having gflow.

**Definition 4.6.** Let  $D$  be a ZX-diagram in MBQC+LC form, with underlying labelled open graph  $(G, I, O, \lambda)$ . Let  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  be the corresponding set of measurement angles. We say a measured vertex  $u \in G$  is *Clifford* when  $\alpha(u) = k\frac{\pi}{2}$  for some  $k$ .

Our goal will be to remove as many internal Clifford vertices as possible. We make a key observation for our simplification scheme: a YZ-plane measurement with a 0 or  $\pi$  phase can be removed from the pattern by modifying its neighbours in a straightforward manner.

**Lemma 4.7.** Let  $D$  be a ZX-diagram in MBQC+LC form with vertices  $V$ . Suppose  $u \in V$  is a non-input vertex measured in the YZ or XZ plane with an angle of  $a\pi$  where  $a = 0$  or  $a = 1$ . Then there is an equivalent diagram  $D'$  with vertices  $V \setminus \{u\}$ . If  $D$  has gflow, then  $D'$  does as well.

*Proof.* Using the axioms of the ZX-calculus, it is straightforward to show that:



These  $a\pi$  phase shifts on the right-hand side can be absorbed into the measurement effects of the neighbouring vertices (or, for output vertices, considered as a local Clifford operator). Absorbing an  $a\pi$  phase shift into a measurement effect does not change the measurement plane, only the angle. The resulting diagram  $D'$  is then also in MBQC+LC form. Since  $G(D')$  is simply  $G(D)$  with a YZ or XZ plane vertex removed,  $D'$  has gflow if  $D$  had gflow by Lemma 3.4.  $\square$

We can combine this observation with local complementation and pivoting to remove vertices measured in other planes or at other angles.

**Lemma 4.8.** Let  $D$  be a ZX-diagram in MBQC+LC form with vertices  $V$ . Suppose  $u \in V$  is a non-input vertex measured in the YZ or XY plane with an angle of  $\pm\frac{\pi}{2}$ . Then there is an equivalent diagram  $D'$  with vertices  $V \setminus \{u\}$ . If  $D$  has gflow, then  $D'$  does as well.

*Proof.* We apply a local complementation about  $u$  using Lemma 2.31 and reduce the diagram to MBQC+LC form with Lemma 4.3. By these lemmas, if the original diagram had gflow, this new diagram will also have gflow. As can be seen from Lemma 4.3, if  $u$  was in the XY plane, then it will be transformed to the XZ plane and will have a measurement angle of  $\frac{\pi}{2} \mp \frac{\pi}{2}$ . As a result its measurement angle is of the form  $a\pi$  for  $a \in \{0, 1\}$ . If instead it was in the YZ plane, then it stays in the YZ plane, but its angle is transformed to  $\frac{\pi}{2} \pm \frac{\pi}{2}$  in which case it will also be of the form  $a\pi$  for  $a \in \{0, 1\}$ . In both cases we can remove the vertex  $u$  using Lemma 4.7 while preserving semantics and the property of having gflow.  $\square$

**Lemma 4.9.** Let  $D$  be a ZX-diagram in MBQC+LC form with vertices  $V$ , and let  $u, v \in V$  be two non-input measured vertices which are neighbours. Suppose that either  $\lambda(u) = XY$  with  $\alpha(u) = a\pi$  for  $a \in \{0, 1\}$  or  $\lambda(u) = XZ$  with  $\alpha(u) = (-1)^a \frac{\pi}{2}$ . Then there is an equivalent diagram  $D'$  with vertices  $V \setminus \{u\}$ . Moreover, if  $D$  has gflow, then  $D'$  also has gflow.

*Proof.* We apply a pivot to  $uv$  using Lemma 2.32 and reduce the diagram to MBQC+LC form with Lemma 4.5. If the original diagram had gflow, this new diagram will also have gflow. As can be seen from Lemma 4.5, if  $\lambda(u) = XY$  then  $\lambda'(u) = YZ$  with  $\alpha'(u) = \alpha(u) = a\pi$ . If instead we had  $\lambda(u) = XZ$  (and thus  $\alpha(u) = (-1)^a \frac{\pi}{2}$ ), then  $\lambda'(u) = XZ$ , but  $\alpha'(u) = \frac{\pi}{2} - \alpha(u) = \frac{\pi}{2} - (-1)^a \frac{\pi}{2} = a\pi$ . In both cases, using Lemma 4.7,  $u$  can be removed while preserving semantics and the existence of gflow.  $\square$

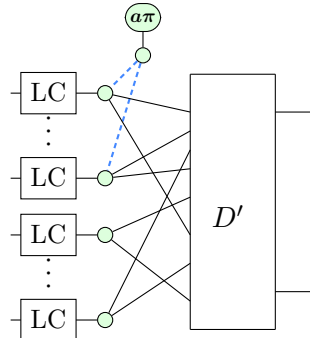
**Remark 4.10.** The ‘graph-like’ ZX-diagrams studied in Ref. [26] are MBQC+LC form diagrams where every vertex is measured in the XY plane. Our Lemmas 4.8 and 4.9 are generalisations of the work found in Ref. [26, Lemmas 5.2 and 5.3] and Ref. [39, (P2) and (P3)].

Combining the three previous lemmas we can remove most non-input Clifford vertices. The exceptions are some non-input Clifford vertices which are only connected to input and output vertices. While it might not always be possible to remove such vertices, when the diagram has a gflow, we can find an equivalent smaller diagram:

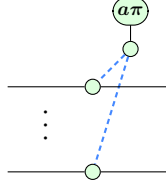
**Lemma 4.11.** Let  $D$  be a ZX-diagram in MBQC+LC form with vertices  $V$  that has a gflow. Let  $u$  be a non-input measured vertex that is only connected to input and output vertices. Suppose that either  $\lambda(u) = XY$  with  $\alpha(u) = a\pi$  for  $a \in \{0, 1\}$  or  $\lambda(u) = XZ$  with  $\alpha(u) = (-1)^a \frac{\pi}{2}$ . Then there exists an equivalent diagram  $D'$  which has gflow and has vertices  $V \setminus \{u\}$ .

*Proof.* We prove the result for  $\lambda(u) = XY$  and  $\alpha(u) = a\pi$ . The other case is similar.

We claim that  $u$  is connected to at least one output that is not also an input. In order to obtain a contradiction suppose otherwise. The diagram then looks as follows:



Here ‘LC’ indicates that there are local Clifford operators on the inputs. Since  $D$  has gflow, the entire diagram must be (proportional to) an isometry, and hence it must still be an isometry if we remove the local Cliffords on the inputs. But we then have the map



as the first operation in the diagram. This map is a projector, and it is not invertible. This is a contradiction, as the entire diagram cannot then be an isometry.

Therefore,  $u$  must be connected to some output vertex  $v$ , which is not an input. We can thus perform a pivot about  $uv$ . This adds a Hadamard operator after  $v$ , and changes the label of  $u$  to YZ. We can then remove  $u$  using Lemma 4.7. As all these operations preserve gflow, the resulting diagram still has gflow.  $\square$

The following result about removing Pauli measurements (i.e. Clifford vertices) from patterns while preserving semantics is already contained in Ref. [30, Section III.A] (if outside the context of MBQC), and is also mentioned in Ref. [33]. Nevertheless, we are the first to explicitly state the effects of this process on the measurement pattern and the gflow.

**Theorem 4.12.** Let  $D$  be a ZX-diagram in MBQC+LC form that has gflow. Then we can find an equivalent ZX-diagram  $D'$  in MBQC+LC form, which also has gflow and which contains no non-input Clifford vertices. The algorithm uses a number of graph operations that is polynomial in the number of vertices of  $D$ .

*Proof.* Starting with  $D$  we simplify the diagram step by step using the following algorithm:

1. Using Lemma 4.8 repeatedly, remove any non-input YZ or XY measured vertex with a  $\pm\frac{\pi}{2}$  phase.
2. Using Lemma 4.7 repeatedly, remove any non-input vertex with measured in the YZ or XZ plane with angle  $a\pi$ .
3. Using Lemma 4.9 repeatedly, remove any XY vertex with an  $a\pi$  phase and any XZ vertex with a  $\pm\frac{\pi}{2}$  phase that are connected to any other internal vertex. If any have been removed, go back to step 1.
4. If there are non-input measured Clifford vertices that are only connected to boundary vertices, use Lemma 4.11 to remove them. Then go back to step 1. Otherwise we are done.

By construction there are no internal Clifford vertices left at the end. Every step preserves gflow, so the resulting diagram still has gflow. As every step removes a vertex, this process terminates in at most  $n$  steps, where  $n$  is the number of vertices in  $D$ . Each of the steps possibly requires doing a pivot or local complementation requiring  $O(n^2)$  elementary graph operations. Hence, the algorithm takes at most  $O(n^3)$  elementary graph operations.  $\square$

**Theorem 4.13.** Suppose  $(G, I, O, \lambda, \alpha)$  is a uniformly deterministic MBQC pattern representing a unitary operation. Assume the pattern involves  $q$  inputs and  $n$  qubits measured at non-Clifford angles, i.e.  $q := |I|$  and  $n := |\{u \in \bar{O} \mid \alpha(u) \neq k\frac{\pi}{2} \text{ for any } k \in \mathbb{Z}\}|$ . Then we can find a uniformly deterministic MBQC pattern that implements the same unitary

and uses at most  $(n + 8q)$  measurements. This process finishes in a number of steps that is polynomial in the number of vertices of  $G$ .

*Proof.* Let  $D$  be the ZX-diagram in MBQC form from Lemma 2.16 that implements the same unitary as the MBQC pattern  $\mathfrak{P} := (G, I, O, \lambda, \alpha)$ . Since  $\mathfrak{P}$  is uniformly deterministic, it has gflow, and hence  $D$  also has gflow by Definition 2.41. Let  $D'$  be the ZX-diagram in MBQC+LC form produced by Theorem 4.12. Since  $D'$  has no internal Clifford vertices, its MBQC-form part can have at most  $n$  internal vertices. It may still have boundary Clifford vertices, and by unitarity  $|O| = |I| = q$ , so the MBQC-form part contains at most  $(n + 2q)$  vertices.

Denote by  $D''$  the MBQC-form diagram produced by applying Lemma 4.1 to  $D'$ . Then  $D''$  has at most  $((n + 2q) + 6q)$  vertices in its MBQC-form part.

We can construct a new pattern  $\mathfrak{P}'$  from  $D''$  using Lemma 2.21. As  $D''$  has gflow,  $\mathfrak{P}'$  also has gflow, and hence is uniformly deterministic. The new pattern  $\mathfrak{P}'$  involves at most  $(n + 8q)$  qubits.

For the complexity of these operations:

- Constructing  $D$  from  $\mathfrak{P}$  takes  $O(|G|)$  operations
- Constructing  $D'$  from  $D$  takes  $O(|G|^3)$  operations
- Constructing  $D''$  from  $D$  takes  $O(|G|)$  operations
- Constructing  $\mathfrak{P}'$  from  $D''$  takes  $O(|G|)$  operations

So the entire process uses  $O(|G|^3)$  graph operations. □

#### 4.4 Phase-gadget form

Using the local complementation and pivoting rules of Section 4.2 we can transform the geometry of MBQC+LC form diagrams so that they no longer contain any vertices measured in the XZ plane, nor will YZ vertices be connected.

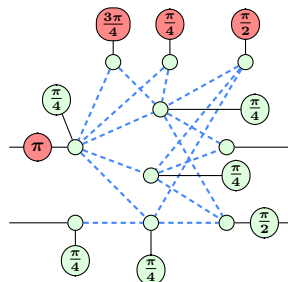
**Definition 4.14.** An MBQC+LC diagram is in *phase-gadget form* if

- there does not exist any  $v \in \bar{O}$  such that  $\lambda(v) = \text{XZ}$ , and
- there does not exist any pair of neighbours  $v, w \in \bar{O}$  such that  $\lambda(v) = \lambda(w) = \text{YZ}$ .

The name ‘phase gadget’ refers to a particular configuration of spiders in the ZX-calculus, which, in our setting, corresponds to spiders measured in the YZ plane. Phase gadgets have been used particularly in the study of circuit optimisation [13, 21, 39].

In Section 4.5 we introduce another form, called *reduced* (Definition 4.20), which requires the pattern to be in phase-gadget form.

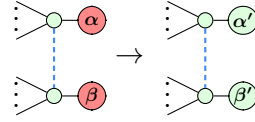
**Example 4.15.** The following MBQC+LC diagram is in phase-gadget form.



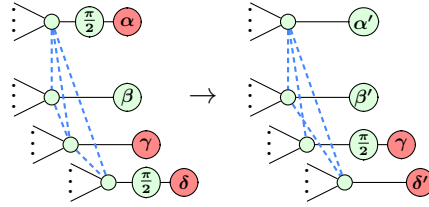
**Proposition 4.16.** Let  $D$  be a ZX-diagram in MBQC+LC form with gflow. Then we can find an equivalent ZX-diagram  $D'$  in MBQC+LC form that has gflow and is in phase-gadget form. This process takes a number of steps that is polynomial in the number of vertices of  $D$ .

*Proof.* Set  $D_0 := D$  and iteratively construct the diagram  $D_{k+1}$  based on  $D_k$ .

- If the diagram  $D_k$  contains a pair of vertices  $u \sim v$  that are both measured in the YZ-plane: First, note that any input vertex  $w$  has  $\lambda(w) = XY$ , as otherwise  $w \in g(w)$  contradicting the co-domain of  $g$  as given in Definition 2.36. Therefore  $u, v \notin I$ . Let  $D_{k+1}$  be the diagram that results from pivoting about the edge  $u \sim v$  (Lemma 2.32) and then transforming to MBQC+LC form (Lemma 4.5.) This changes the measurement plane for  $u$  and  $v$  from YZ to XY and it does not affect the measurement planes for any other vertices:



- If there is no such connected pair but there is some vertex  $u$  that is measured in the XZ-plane: Note that  $u$  cannot be an input vertex by the same reasoning as in the first subcase. Let  $D_{k+1}$  be the diagram that results from applying a local complementation on  $u$  and transforming back to MBQC+LC form (Lemmas 2.31 and 4.3).



As can be seen from Lemma 4.3, this process changes the measurement plane of  $u$  from XZ to YZ and it does not affect the labels of any vertices that are measured in the XY-plane.

- If there is no such connected pair, nor any vertex that is measured in the XZ-plane then  $D_k$  is already in the desired form, so halt.

The number of vertices not measured in the XY-plane decreases with each step, and no vertices are added, so this process terminates in at most  $n$  steps, where  $n$  is the number of vertices in  $D$ . Each step requires checking every pair of vertices, or performing local complementation, each of which have complexity  $O(n^2)$ , so the total complexity is  $O(n^3)$ . Since a pivot is just a sequence of local complementations,  $D_{k+1}$  has gflow if  $D_k$  had gflow (Proposition 4.4). Finally every step preserves equivalence, so  $D_{k+1}$  is equivalent to  $D_k$ .  $\square$

Proposition 4.16 finds a phase-gadget form for an MBQC+LC diagram, but note that the phase-gadget form is not guaranteed to be unique.

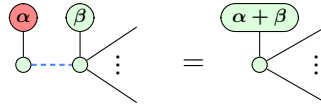
## 4.5 Further pattern optimisation

In Section 4.3 we saw that we can remove all non-input Clifford qubits from a pattern while preserving determinism and the computation implemented by the pattern, up to local Clifford transformations. We will show in this section that it is also possible to remove certain qubits measured in non-Clifford angles.

These measurement pattern rewrite rules, seen then as transformations of ZX-diagrams, were used in Ref. [39] to reduce the T-count of circuits. We will see how in our context they can be used to remove a qubit from a pattern, again while preserving determinism. First of all, any internal YZ vertex with just one neighbour can be fused with this neighbour, resulting in the removal of the YZ vertex:

**Lemma 4.17.** Let  $D$  be an MBQC+LC diagram with an interior vertex  $u$  measured in the YZ plane, and suppose it has a single neighbour  $v$ , which is measured in the XY plane. Then there is an equivalent MBQC+LC diagram  $D'$  with  $G(D') = G(D) \setminus \{u\}$ . If  $D$  has gflow, then  $D'$  also has gflow.

*Proof.* We apply the following rewrite:



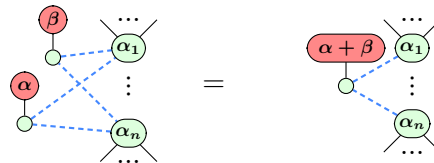
The resulting diagram is again an MBQC+LC diagram. The change to the labelled open graph comes down to deleting a YZ vertex. By Lemma 3.4 this preserves gflow.  $\square$

Note that, by Proposition 3.16, if the diagram has gflow and an equal numbers of inputs and outputs, then it has no internal XY vertices with just one neighbour. Thus, if the diagram is in phase-gadget form (cf. Definition 4.14), the above lemma allows us to remove all internal vertices which have a single neighbour.

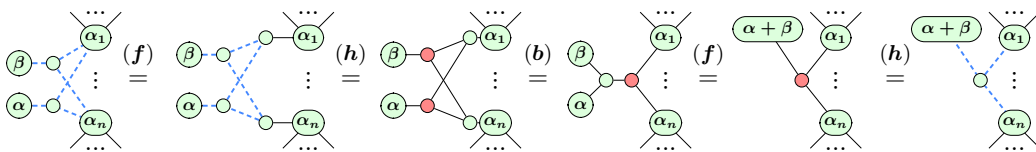
Our second rewrite rule allows us to also ‘fuse’ YZ vertices that have the same set of neighbours:

**Lemma 4.18.** Let  $D$  be an MBQC+LC diagram with two distinct interior vertices  $u$  and  $v$ , both measured in the YZ plane and with  $N(u) = N(v)$ . Then there is an equivalent diagram  $D'$  with  $G(D') = G(D) \setminus \{u\}$ . If  $D$  has gflow, then  $D'$  also has gflow.

*Proof.* We apply the following rewrite:



A straightforward sequence of ZX-calculus transformations shows this rewrite preserves semantics:



The new diagram is still an MBQC+LC diagram, and the only change in the underlying labelled open graph is the deletion of a YZ vertex. Hence, by Lemma 3.4, this rewrite preserves gflow.  $\square$

The analogous result is not true for a pair of XY vertices. However, when the diagram has gflow, such pairs cannot exist to begin with:

**Lemma 4.19.** Let  $G$  be a labelled open graph with gflow and distinct interior vertices  $u$  and  $v$  both measured in the XY plane. Then  $N(u) \neq N(v)$ .

*Proof.* Assume for a contradiction that  $N(u) = N(v)$  and that the diagram has gflow. Note that, for any subset of vertices  $S$ , we have  $u \in \text{Odd}(S) \iff v \in \text{Odd}(S)$ . In particular, as  $u \in \text{Odd}(g(u))$  by (g3), we have  $v \in \text{Odd}(g(u))$  and thus  $u \prec v$  by (g2). Yet, swapping  $u$  and  $v$  in the above argument, we also find  $v \prec u$ , a contradiction. Thus, if the diagram has gflow, distinct vertices  $u$  and  $v$  must have distinct neighbourhoods  $N(u) \neq N(v)$ .  $\square$

We can now combine these rewrite rules with our previous results to get a more powerful rewrite strategy:

**Definition 4.20.** Let  $D$  be an MBQC+LC diagram. We say  $D$  is *reduced* when:

- It is in phase-gadget form (see Definition 4.14).
- It has no internal Clifford vertices.
- Every internal vertex has more than one neighbour.
- If two distinct vertices are measured in the same plane, they have different sets of neighbours.

**Theorem 4.21.** Let  $D$  be an MBQC+LC diagram with gflow and equal numbers of inputs and outputs. Then we can find an equivalent diagram  $D'$  that is reduced and has gflow. This process finishes in a number of steps that is polynomial in the number of vertices of  $D$ .

*Proof.* Starting with  $D$ , we simplify the diagram step by step with the following algorithm:

1. Apply Theorem 4.12 to remove all interior Clifford vertices.
2. Apply Proposition 4.16 to bring the diagram into phase-gadget form. Then every vertex is of type YZ or XY, and the YZ vertices are only connected to XY vertices.
3. Apply Lemma 4.17 to remove any YZ vertex that has a single neighbour.
4. Apply apply Lemma 4.18 to merge any pair of YZ vertices that have the same set of neighbours.
5. If the application of these lemmas resulted in any new internal Clifford vertices, go back to step 1. Otherwise we are done.

Each of the steps preserves gflow, and hence at every stage of the algorithm the diagram has gflow. By construction, when the algorithm has terminated, every vertex is now of type YZ or XY, and YZ vertices are only connected to XY vertices. Furthermore, every YZ vertex must have more than one neighbour and have a different set of neighbours than any other YZ vertex. This is also true for the XY vertices by the existence of gflow and the requirement that the number of inputs match the number of outputs (using Lemma 4.19 and Proposition 3.16). Hence, the resulting diagram has all the properties needed for it to be reduced.

To show this process terminates consider the lexicographic order:

- Number of vertices in the diagram
- Number of vertices measured in the XZ or YZ planes
- Number of vertices measured in the XY plane

The result of each step of the algorithm on this order is:

- Applying Theorem 4.12 reduces the number of vertices in the diagram, while possibly increasing the number of vertices in any given plane.
- Applying Proposition 4.16 reduces the number of vertices in the XZ or YZ planes, while possibly increasing the number of vertices in the XY plane.
- Applying Lemmas 4.17 and 4.18 reduces the number of vertices in the diagram, and the number of vertices measured in the YZ plane.

Therefore each step in the algorithm reduces our order, so the process terminates. Writing  $n$  for the number of vertices in  $D$  we see that the algorithmic loop can be called at most  $n$  times (since we remove vertices each iteration), and each of the steps in the loop take at most  $O(n^3)$  operations, giving a total complexity of  $O(n^4)$ .  $\square$

**Remark 4.22.** The algorithm described above uses the same idea as that described in Ref. [39]. But while they describe the procedure in terms of modifying a graph-like ZX-diagram, we describe it for MBQC+LC diagrams, a more general class of diagrams. Furthermore, we prove that the procedure preserves the existence of gflow. The existence of gflow is used in the next section to show how to recover a circuit from an MBQC+LC diagram.

## 5 Circuit extraction

In this section we will see that we can extract a circuit from a measurement pattern whose corresponding labelled open graph has a gflow.

There are several results known about extracting circuits from measurement patterns. In the original paper defining the flow condition for measurement patterns [17], an ancilla-free circuit extraction algorithm for such patterns was presented. A number of procedures have also been found to extract circuits containing ancillae from XY-plane patterns [10, 15, 22]. In Ref. [24] an algorithm was sketched to extract an ancilla-free circuit from a XY-plane pattern with gflow. A version of this algorithm was further worked out in [45]. Independently, an algorithm for the same task was found in Ref. [26] that uses the ZX-calculus to derive correctness. Our algorithm modifies that of Ref. [26] so that it can handle measurements in multiples planes (and not just the XY plane). While the algorithm itself is a relatively straightforward extension of that of Ref. [26], its proof of correctness relies on the new notion of focused gflow that we found in Section 3.3 and several other lemmas regarding extended gflow that are not obvious.

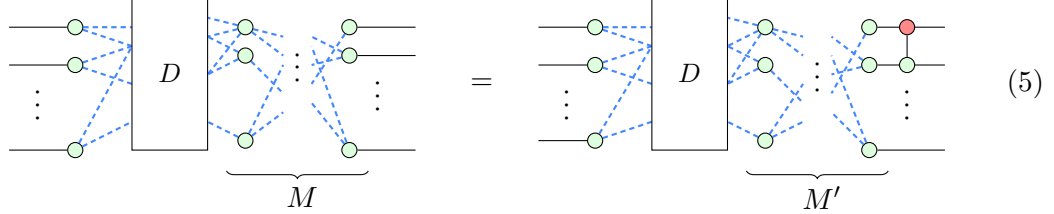
Instead of describing the algorithm for measurement patterns, we describe it for the more convenient form of MBQC+LC diagrams. The general idea is that we modify the diagram vertex by vertex to bring it closer and closer to resembling a circuit. We start at the outputs of the diagram and work our way to the inputs. The gflow informs the choice of which vertex is next in line to be ‘extracted’ (specifically, this will always be a vertex maximal in the gflow partial order). By applying various transformations to the diagram, we change it so that the targeted vertex can easily be pulled out of the MBQC-form part



of the diagram and into the circuit-like part. The remaining MBQC-form diagram is then one vertex smaller. Since all the transformations preserve gflow, we can then repeat the procedure on this smaller diagram until we are finished.

Before we explain the extraction algorithm in detail in Section 5.1, we state some relevant lemmas.

**Lemma 5.1.** The following equation holds:



where  $M$  is the biadjacency matrix of the output vertices to the neighbours of  $D$ , and  $M'$  is the matrix produced from  $M$  by adding row 1 to row 2, modulo 2. If the full diagram on the LHS has gflow, then so does the RHS.

*Proof.* The equality is proved in Ref. [26, Proposition 6.2]. There it is also shown that this preserves gflow when all measurements are in the XY plane, but the same proof works when measurement in all three planes are present.  $\square$

**Lemma 5.2.** Suppose  $(G, I, O, \lambda)$  is a labelled open graph with gflow. Let  $G'$  be the graph containing the same vertices as  $G$  and the same edges except those for which both endpoints are output vertices. Formally, if  $G = (V, E)$ , then  $G' = (V, E')$ , where

$$E' = \{v \sim w \in E \mid v \in \bar{O} \text{ or } w \in \bar{O}\}.$$

Then  $(G', I, O, \lambda)$  also has gflow.

*Proof.* We claim that if  $(g, \prec)$  is a gflow for  $G$ , then it is also a gflow for  $G'$ . Note that  $\text{Odd}_{G'}(g(v)) \cap \bar{O} = \text{Odd}_G(g(v)) \cap \bar{O}$  as the only changes to neighbourhoods are among the output vertices. It is thus easily checked that all properties of Definition 2.36 remain satisfied.  $\square$

**Lemma 5.3.** Let  $(G, I, O, \lambda)$  be a labelled open graph with a gflow and the same number of inputs as outputs:  $|I| = |O|$ . Let  $v \in O \cap \bar{I}$  be an output which is not an input. Suppose  $v$  has a unique neighbour  $u \in \bar{O}$ . Then  $\lambda(u) = XY$ .

*Proof.* Suppose, working towards a contradiction, that  $\lambda(u) \neq XY$ . Form the labelled open graph  $(G', I, O, \lambda')$  by removing from  $G$  all vertices  $w$  such that  $w \in \bar{O}$  and  $\lambda(w) \neq XY$ , restricting  $\lambda$  accordingly. By Lemma 3.15 the labelled open graph  $(G', I, O, \lambda')$  also has a gflow. Note that  $G'$  does contain  $v$ , which is still an output vertex in  $G'$ , but does not contain  $u$ , and hence  $v$  has no neighbours in  $G'$ . By Theorem 2.45,  $G'$  has a focused gflow, and because  $G'$  has the same number of inputs as outputs, its reversed graph also has a gflow  $(g, \prec)$  by Corollary 2.47. In this reversed graph  $v$  is an input and, since it is not an output, it is measured in the XY plane. It therefore has a correction set  $g(v)$  so that  $v \in \text{Odd}(g(v))$ . But because  $v$  has no neighbours, this is a contradiction. We conclude that indeed  $\lambda(u) = XY$ .  $\square$

For any set  $A \subseteq V$ , let  $N_G(A) = \bigcup_{v \in A} N_G(v)$ . Recall the partition of vertices according to the partial order of the gflow into sets  $V_k^{\prec}$ , which is introduced in Definition 3.9.

**Lemma 5.4.** Let  $(G, I, O, \lambda)$  be a labelled open graph in phase-gadget form, which furthermore satisfies  $\overline{O} \neq \emptyset$ . Suppose  $(G, I, O, \lambda)$  has a gflow. Then the maximally delayed gflow,  $(g, \prec)$ , constructed in Proposition 3.14 exists and moreover  $N_G(V_1^{\prec}) \cap O \neq \emptyset$ , i.e. the gflow has the property that, among the non-output vertices, there is a vertex which is maximal with respect to the gflow order and also connected to an output vertex.

*Proof.* By Proposition 3.14, there exists a maximally delayed gflow of  $(G, I, O, \lambda)$  such that no element of a correction set (other than possibly the vertex being corrected) is measured in the YZ plane.

Since the open graph does not consist solely of outputs, the set  $V_1^{\prec}$  (as defined in Definition 3.9) is non-empty, so the following arguments are non-trivial. For any  $v \in V_1^{\prec}$  we must have  $g(v) \subseteq O \cup \{v\}$ . Now if there is a  $v \in V_1^{\prec}$  with  $\lambda(v) = XY$ , then  $v \in \text{Odd}(g(v))$ . There are no self-loops, hence this  $v$  must be connected to at least one output, and we are done. As the graph is in phase-gadget form, there are no vertices labelled XZ and hence from now on assume that  $\lambda(v) = YZ$  for all  $v \in V_1^{\prec}$ . We distinguish two cases.

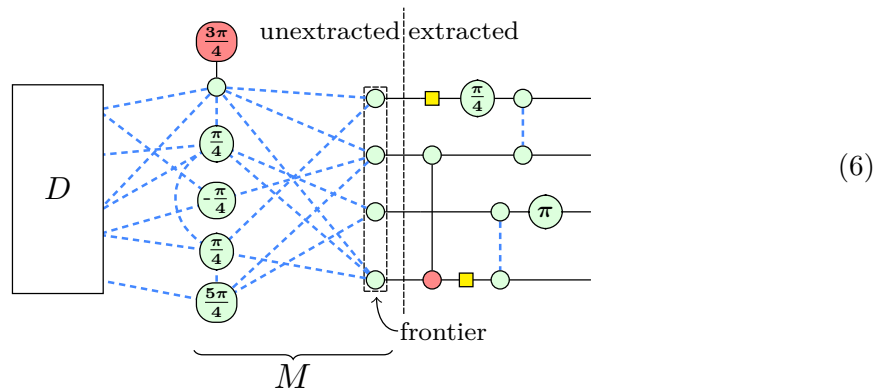
1. If  $V_2^{\prec} = \emptyset$ , then the only non-output vertices are in  $V_1^{\prec}$ . Now, any connected component of the graph  $G$  must contain an input or an output. The vertices in  $V_1^{\prec}$  are all labelled YZ and thus appear in their own correction sets; this means they cannot be inputs because inputs do not appear in correction sets. The vertices in  $V_1^{\prec}$  are not outputs either, so each of them must have at least one neighbour. Yet the labelled open graph is in phase-gadget form. This implies that two vertices both labelled YZ cannot be adjacent, and all vertices in  $V_1^{\prec}$  are labelled YZ. Thus any vertex  $v \in V_1^{\prec}$  must have a neighbour in  $O$ , and we are done.
2. So now assume there is some vertex  $w \in V_2^{\prec}$ . Then, regardless of  $\lambda(w)$ , we have  $g(w) \subseteq V_1^{\prec} \cup O \cup \{w\}$  and  $\text{Odd}(g(w)) \subseteq V_1^{\prec} \cup O \cup \{w\}$ . We distinguish three subcases according to whether one of  $g(w)$  or  $\text{Odd}(g(w))$  intersects  $V_1^{\prec}$ .
  - Suppose  $g(w) \cap V_1^{\prec} = \text{Odd}(g(w)) \cap V_1^{\prec} = \emptyset$ , i.e.  $g(w) \subseteq O \cup \{w\}$  and  $\text{Odd}(g(w)) \subseteq O \cup \{w\}$ . Let  $\prec' = \prec \setminus \{(w, u) : u \in V_1^{\prec}\}$ . Then  $(g, \prec')$  is a gflow: dropping the given inequalities from the partial order does not affect the gflow properties since  $u \in V_1^{\prec}$  implies  $w \notin g(u)$  and  $w \notin \text{Odd}(g(u))$ . Furthermore,  $(g, \prec')$  is more delayed than  $(g, \prec)$  because  $w$  (and potentially some of its predecessors) moves to an earlier layer, contradicting the assumption that  $(g, \prec)$  is maximally delayed. Hence this case cannot happen.
  - Suppose  $g(w) \cap V_1^{\prec} \neq \emptyset$ , then there exists a YZ vertex in the correction set of  $w$  since all elements of  $V_1^{\prec}$  are measured in the YZ plane. But our gflow satisfies the properties of Proposition 3.14, and hence this cannot happen.
  - Suppose  $\text{Odd}(g(w)) \cap V_1^{\prec} \neq \emptyset$  and  $g(w) \cap V_1^{\prec} = \emptyset$ , then there is a  $v \in V_1^{\prec}$  such that  $v \in \text{Odd}(g(w))$ . There are two further subcases.
    - If  $\lambda(w) = XY$ , we have  $w \notin g(w)$  and hence  $g(w) \subseteq O$  so that there must be some  $o \in O$  that is connected to  $v$  and we are done.
    - Otherwise, if  $\lambda(w) = YZ$ , then  $w \in g(w)$ . Yet both  $v$  and  $w$  are measured in the YZ plane, so they are not neighbours, and hence there still must be an  $o \in O$  that is connected to  $v$  to have  $v \in \text{Odd}(g(w))$ .

Thus, the gflow  $(g, \prec)$  has the desired property in all possible cases.  $\square$

## 5.1 General description of the algorithm

We first walk through a high-level description of how to extract a circuit from a diagram in MBQC+LC form with gflow, explaining why every step works. After that, we present a more practical algorithm in Section 5.2. As we wish the output to be a unitary circuit, we will assume that the diagram has an equal number of inputs and outputs.

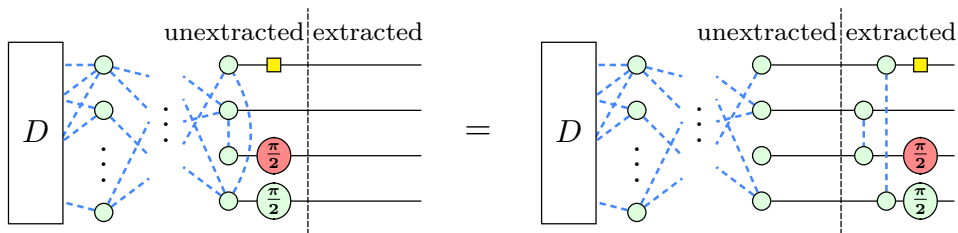
The process will be to make sequential changes to the ZX-diagram that make the diagram look progressively more like a circuit. During the process, there will be a ‘frontier’: a set of green spiders such that everything to their right looks like a circuit, while everything to their left (and including the frontier vertices themselves) is an MBQC+LC form diagram equipped with a gflow. We will refer to the MBQC-form diagram on the left as the *unextracted* part of the diagram, and to the circuit on the right as the *extracted* part of the diagram. For example:



In this diagram, we have merged the XY measurement effects with their respective vertices, in order to present a tidier picture. The matrix  $M$  is the biadjacency matrix between the vertices on the frontier and all their neighbours to the left of the frontier. For the purposes of the algorithm below, we consider the extracted circuit as no longer being part of the diagram, and hence the frontier vertices are the outputs of the labelled open graph of the unextracted diagram.

**Step 0:** First, we transform the pattern into phase-gadget form using Proposition 4.16, ensuring that all vertices are measured in the XY or YZ planes, and that vertices measured in the YZ plane are only connected to vertices measured in the XY plane. This can be done in polynomial time, and preserves the interpretation of the diagram. Furthermore, the resulting diagram still has gflow.

**Step 1:** We unfuse any connection between the frontier vertices as a CZ gate into the extracted circuit, and we consider any local Clifford operator on the frontier vertices as part of the extracted circuit. For example:



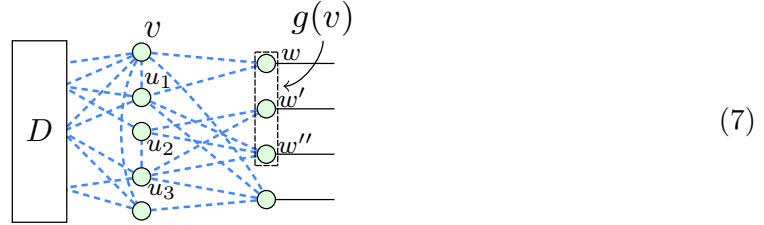
This process changes the unextracted diagram in two ways: by removing local Clifford operators and by removing connections among the frontier vertices. The former does not affect the underlying labelled open graph and the latter preserves gflow by Lemma 5.2.

Thus, the unextracted diagram continues to be in MBQC form and it continues to have gflow. If the only unextracted vertices are on the frontier, go to step 5, otherwise continue to step 2.

**Step 2:** The unextracted diagram is in phase-gadget form and has gflow. Thus, by Lemma 5.4, it has a maximally delayed gflow  $(g, \prec)$  such that  $N_G(V_1^{\prec}) \cap O \neq \emptyset$ , where  $V_1^{\prec}$  is the ‘most delayed’ layer before the frontier vertices, which are the outputs of the labelled open graph (see Definition 3.9). Such a gflow can be efficiently determined by first finding any maximally delayed gflow using the algorithm of Theorem 3.11 and then following the procedure outlined in Proposition 3.14.

Now, if any of the vertices in  $V_1^{\prec}$  are labelled XY, pick one of these vertices and go to step 3. Otherwise, all the maximal non-output vertices (with respect to  $\prec$ ) must have label YZ; go to step 4.

**Step 3:** We have a maximal non-output vertex  $v$  labelled XY, which we want to extract. Since it is maximal in  $\prec$ , we know that  $g(v) \subseteq O$  by Definition 3.9. As the gflow is maximally delayed, we have  $\text{Odd}(g(v)) \cap \bar{O} = \{v\}$ . We now follow the strategy used in Ref. [26] for the ‘XY-plane only’ case, illustrating it with an example. Consider the following diagram, in which the vertex  $v$  and its correction set  $g(v)$  are indicated:



For clarity, we are ignoring the measurement effects on the left-hand-side spiders, and we are not showing any frontier vertices that are inputs (although note that by definition of a gflow, the vertices of  $g(v)$  cannot be inputs). In the above example, the biadjacency matrix of the bipartite graph between the vertices of  $g(v)$  on the one hand, and their neighbours in the unextracted part on the other hand, is

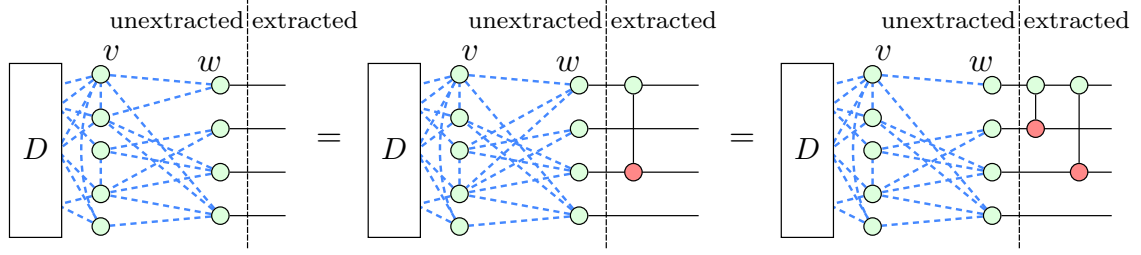
$$\begin{array}{c} \\ w \\ w' \\ w'' \end{array} \begin{array}{cccc} v & u_1 & u_2 & u_3 \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \quad (8)$$

where the rows correspond to vertices of  $g(v)$ , and vertices are ordered top-to-bottom. We do not include the bottom-most frontier vertex in the biadjacency matrix, as it is not part of  $g(v)$ , and we do not include the bottom left spider, as it is not connected to any vertex in  $g(v)$ .

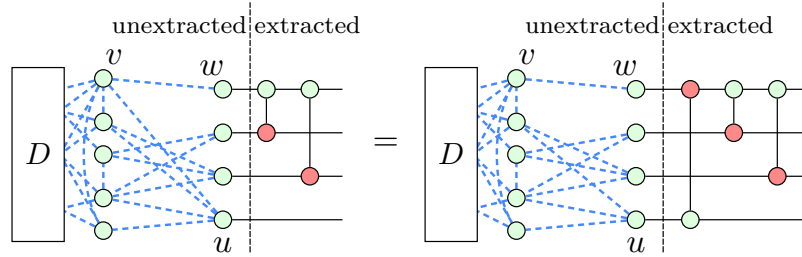
The property that  $\text{Odd}(g(v)) \cap \bar{O} = \{v\}$  now corresponds precisely to the following property of the matrix: if we sum up all the rows of this biadjacency matrix modulo 2, the resulting row vector contains a single 1 corresponding to the vertex  $v$  and zeroes everywhere else. It is straightforward to see that this is indeed the case for the matrix of Eq. (8).

Now pick any frontier vertex  $w \in g(v)$ . Lemma 5.1 shows that the application of a CNOT to two outputs corresponds to a row operation on the biadjacency matrix, which adds the row corresponding to the target to the row corresponding to the control. Hence if, for each  $w' \in g(v) \setminus \{w\}$ , we apply a CNOT with control and target on the output wires of  $w$  and  $w'$ , the effect on the biadjacency matrix is to add all the other rows of the

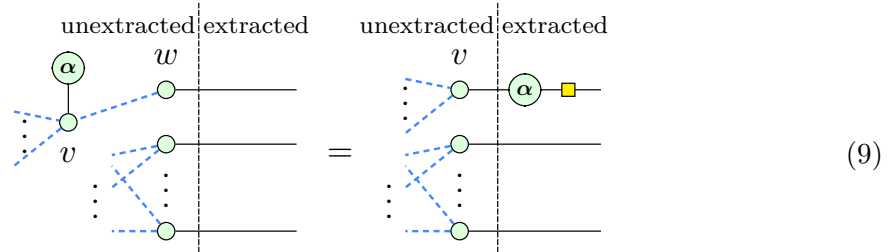
vertices of  $g(v)$  to that of  $w$ :



As a result,  $w$  is now only connected to  $v$ , but  $v$  may still be connected to other vertices in  $O \setminus g(v)$ . For each such vertex  $u$ , applying a CNOT with control  $u$  and target  $w$  removes the connection between  $u$  and  $v$ :



Now we can move  $v$  to the frontier by removing  $w$  from the diagram, adding a Hadamard to the circuit (this comes from the Hadamard edge between  $v$  and  $w$ ), adding the measurement angle of  $v$  to the circuit as a Z-phase gate, and adding  $v$  to the set of outputs of the graph (i.e. the frontier):



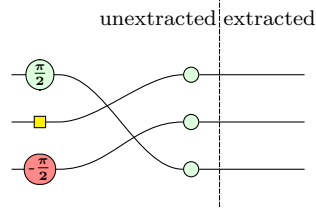
On the underlying labelled open graph this corresponds to removing  $w$  and adding  $v$  to the list of outputs. We need to check that this preserves the existence of a gflow. The only change we need to make is that for all  $v' \neq v$  with  $w \in g(v')$  we set  $g'(v') = g(v') \setminus \{w\}$ . As  $w$ 's only neighbour is  $v$ , removing  $w$  from  $g(v')$  only toggles whether  $v \in \text{Odd}(g'(v'))$ . Since  $v$  is a part of the outputs in the new labelled open graph, this preserves all the properties of being a gflow.

Now that the vertex  $w$  has been removed, the number of vertices in the unextracted part of the diagram is reduced by 1. We now go back to step 1.

**Step 4:** All the maximal vertices are labelled YZ. Since we chose our gflow according to Lemma 5.4, we know that at least one of these vertices is connected to an output, and hence a frontier vertex. Pick such a vertex  $v$ , and pick a  $w \in O \cap N_G(v)$  (this set is non-empty). Pivot about  $vw$  using Lemma 2.32 and reduce the resulting diagram to MBQC form with Lemma 4.5. Afterwards,  $v$  has label XY and  $w$  has a new Hadamard gate on its output wire (which will be dealt with in the next iteration of step 1).

We have changed one vertex label in the unextracted part of the diagram from YZ to XY. Since no step introduces new YZ vertices, step 4 can only happen as many times as there are YZ vertices at the beginning. Go back to step 1.

**Step 5:** At this point, there are no unextracted vertices other than the frontier vertices, all of which have arity 2 and can be removed using rule (i1) of Figure 2. Yet the remaining frontier vertices might be connected to the inputs in some permuted manner and the inputs might carry some local Cliffords:



This is easily taken care of by decomposing the permutation into a series of SWAP gates, at which point the entire diagram is in circuit form.

**Correctness and efficiency:** Since step 3 removes a vertex from the unextracted diagram, and step 4 changes a measurement plane from YZ to XY (and no step changes measurement planes in the other direction), this algorithm reduces the lexicographic order (# unextracted vertices, # unextracted vertices with  $\lambda = YZ$ ) each time we repeat an iteration of the process, so that the algorithm terminates. Each step described above takes a number of graph-operations polynomial in the number of unextracted vertices, and therefore this entire algorithm takes a number of steps polynomial in the number of vertices. All steps correspond to ZX-diagram rewrites, so the resulting diagram is a circuit that implements the same unitary as the pattern we started with.

## 5.2 A more practical algorithm

Now we know that the algorithm above is correct and will always terminate, we can take a couple of short-cuts that will make it more efficient.

In step 2, instead of using the gflow to find a maximal vertex, we do the following: Write down the biadjacency matrix of the bipartite graph consisting of frontier vertices on one side and all their neighbours on the other side. For example, the Diagram (7) would give the matrix:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \quad (10)$$

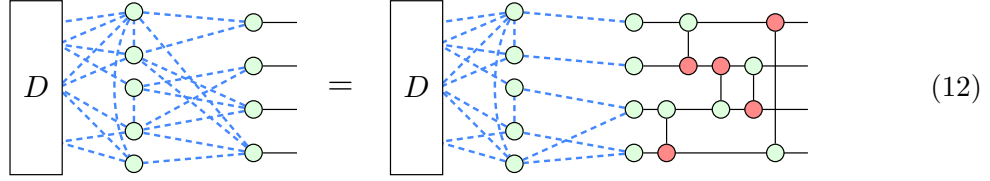
Now perform a full Gaussian elimination on this  $\mathbb{Z}_2$  matrix. In the above case, this results in the matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad (11)$$

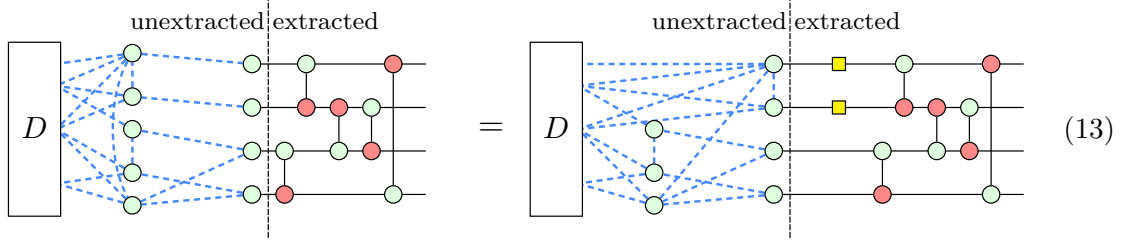
Any row in this matrix containing a single 1 corresponds to an output vertex with a single neighbour. By Lemma 5.3, this neighbour is of type XY. As an example, in the matrix in Eq. (11), the first row has a single 1 in the first column, and hence the top-left spider of Diagram (7) is the unique XY neighbour to the first output. Similarly, the second row has a single 1, appearing in column 2, and hence the second spider from the top on the left in Diagram (7) is the unique neighbour to the second output.

If we found at least one row with a single 1 with this method, we implement the row operations corresponding to the Gaussian elimination procedure as a set of CNOT gates

using Lemma 5.1. Doing this with Diagram (7) gives:



We see that every row which had a single 1 now corresponds to a frontier spider with a single neighbour, and hence we can extract vertices using the technique of Eq. (9):



As we now extract multiple vertices at a time, there could be connections between the new frontier vertices (for instance between the top two frontier spiders in Eq. (13)). These are taken care of in the next iteration of step 1, turning those into CZ gates.

If the Gaussian elimination does not reveal a row with a single 1, then we are in the situation of step 4. We perform pivots involving a vertex with label YZ and an adjacent frontier vertex until there is no vertex with a label YZ which is connected to a frontier vertex. We then go back to step 1.

With these short-cuts, it becomes clear that we do not need an explicitly calculated gflow in order to extract a circuit. The fact that there is a gflow is only used to argue that the algorithm is indeed correct and will always terminate. Pseudocode for this algorithm can be found in Appendix D.

With the results of this section we have then established the following theorem.

**Theorem 5.5.** Let  $\mathfrak{P}$  be a measurement pattern with  $n$  inputs and outputs containing a total of  $k$  qubits, and whose corresponding labelled open graph has a gflow. Then there is an algorithm running in time  $O(n^2k^2 + k^3)$  that converts  $\mathfrak{P}$  into an equivalent  $n$ -qubit circuit that contains no ancillae.

*Proof.* The runtime for the extraction algorithm is dominated by Gaussian elimination of the biadjacency matrices which has complexity  $O(n^2m)$ , where  $n$  is the number of rows, corresponding to the number of outputs, and  $m$  is the number of neighbours these output vertices are connected to. In principle  $m$  could be as large as the number of vertices in the graph and hence could be as large as  $k$  (although in practice it will be much smaller than that). In the worst case, performing a pivot operation also requires toggling the connectivity of almost the entire graph, which requires  $k^2$  elementary graph operations. Since we might have to apply a pivot and a Gaussian elimination process for every vertex in the graph, the complexity for the entire algorithm is bounded above by  $O(k(n^2k + k^2)) = O(n^2k^2 + k^3)$ .  $\square$

Note that if  $k \geq O(n^2)$ , which will be the case for most useful computation, the bound becomes  $O(k^3)$ . In practice however we would not expect to see this worst-case complexity as it would only be attained if everything is almost entirely fully connected all the time. This does not seem possible because the pivots and Gaussian elimination always toggle connectivity, and hence a highly connected graph in one step will become less connected in the following step.

## 6 Conclusions and Future Work

We have given an algorithm which extracts a circuit from any measurement pattern whose underlying labelled open graph has extended gflow. This is the first algorithm which works for patterns with measurements in multiple planes, and does not use ancillae. Simultaneously, it is the most general known algorithm for extracting quantum circuits from ZX-calculus diagrams.

We have also developed a set of rewrite rules for measurement patterns containing measurements in all three planes. For each of these rewrite rules, we have established the corresponding transformations of the extended gflow. The rewrite rules can be used to reduce the number of qubits in a measurement pattern, in particular eliminating all qubits measured in a Pauli basis. Additionally, we have generalised the notions of focused gflow and maximally delayed gflow to labelled open graphs with measurements in multiple planes, and we have described algorithms for finding such gflows.

The pattern optimisation algorithm of Theorem 4.21 and the circuit extraction algorithm of Section 5 have been implemented in the ZX-calculus rewriting system *PyZX*<sup>5</sup> [38]. The reduction in non-Clifford gates using this method matches the state-of-the-art for ancillae-free circuits [39] at the time of development.

Our circuit extraction procedure resynthesises the CNOT gates in the circuit. Depending on the input circuit this can lead to drastic decreases in the 2-qubit gate count of the circuit [26, 38], but in many cases it can also lead to an *increase* of the CNOT count. Such increases are avoided in the procedure of Ref. [39], where two-qubit gates are not resynthesised.

Yet re-synthesis of two-qubit gates may be necessary anyway in many applications: current and near-term quantum devices do not allow two-qubit gates to be applied to arbitrary pairs of qubits. Thus, general circuits need to be adapted to the permitted connections; this is called routing. Our extraction algorithm currently uses a standard process of Gaussian elimination to produce the two-qubit gates required to implement the circuit, which implicitly assumes that two-qubit gates can be applied between any pair of qubits in the circuit. It may be useful to replace this procedure with one incorporating routing, such as the ZX-calculus-based routing approach by Kissinger and Meijer-van de Griend [36]. This would allow circuits to be adapted to the desired architecture as they are being extracted.

It would also be interesting to consider whether these routing algorithms can be used more abstractly to transform general measurement patterns to more regular patterns with restricted connectivity.

**Acknowledgements.** Many thanks to Fatimah Ahmadi for her contributions in the earlier stages of this project. We would also like to thank Will Simmons for pointing out an error in Lemma C.4 in a previous version of this paper. The majority of this work was developed at the Applied Category Theory summer school during the week of 22–26 July 2019; we thank the organisers of this summer school for bringing us together and making this work possible. JvdW is supported in part by AFOSR grant FA2386-18-1-4028 and is supported by a Rubicon fellowship financed by the Dutch Research Council (NWO). HJM-B is supported by the EPSRC.

---

<sup>5</sup>PyZX is available at <https://github.com/Quantomatic/pyzx>. A Jupyter notebook demonstrating the T-count optimisation is available at <https://github.com/Quantomatic/pyzx/blob/5d409a246857b7600cc9bb0fbc13043d54fb9449/demos/T-count%20Benchmark.ipynb>.



## References

- [1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004. DOI: [10.1103/PhysRevA.70.052328](https://doi.org/10.1103/PhysRevA.70.052328).
- [2] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. 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): 818–830, 2013. DOI: [10.1109/TCAD.2013.2244643](https://doi.org/10.1109/TCAD.2013.2244643).
- [3] Matthew Amy, Dmitri Maslov, and Michele Mosca. 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):1476–1489, 2014. DOI: [10.1109/TCAD.2014.2341953](https://doi.org/10.1109/TCAD.2014.2341953).
- [4] Miriam Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014. ISSN 1367-2630. DOI: [10.1088/1367-2630/16/9/093021](https://doi.org/10.1088/1367-2630/16/9/093021).
- [5] Miriam Backens. Making the stabilizer ZX-calculus complete for scalars. In Chris Heunen, Peter Selinger, and Jamie Vicary, editors, *Proceedings of the 12th International Workshop on Quantum Physics and Logic (QPL 2015)*, volume 195 of *Electronic Proceedings in Theoretical Computer Science*, pages 17–32, 2015. DOI: [10.4204/EPTCS.195.2](https://doi.org/10.4204/EPTCS.195.2).
- [6] Gregory Bard. Achieving a  $\log(n)$  speed up for boolean matrix operations and calculating the complexity of the dense linear algebra step of algebraic stream cipher attacks and of integer factorization methods. *IACR Cryptology ePrint Archive*, 2006:163, 01 2006.
- [7] Xiaoning Bian and Peter Selinger. Relations for 2-qubit Clifford+T operator group, 2015. URL [https://mathstat.dal.ca/~xbian/talks/slide\\_cliffordt2.pdf](https://mathstat.dal.ca/~xbian/talks/slide_cliffordt2.pdf).
- [8] Anne Broadbent and Elham Kashefi. Parallelizing quantum circuits. *Theoretical Computer Science*, 410(26):2489–2510, 2009. DOI: [10.1016/j.tcs.2008.12.046](https://doi.org/10.1016/j.tcs.2008.12.046).
- [9] Daniel E Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9(8):250, 2007. DOI: [10.1088/1367-2630/9/8/250](https://doi.org/10.1088/1367-2630/9/8/250).
- [10] Daniel E. Browne, Elham Kashefi, and Simon Perdrix. Computational depth complexity of measurement-based quantum computation. In *Theory of Quantum Computation, Communication, and Cryptography (TQC'10)*, volume 6519, pages 35–46. LNCS, 2011. DOI: [10.1007/978-3-642-18073-6\\_4](https://doi.org/10.1007/978-3-642-18073-6_4).
- [11] B. Coecke and R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13:043016, 2011. DOI: [10.1088/1367-2630/13/4/043016](https://doi.org/10.1088/1367-2630/13/4/043016). arXiv:quant-ph/09064725.
- [12] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. ISBN 9781107104228. DOI: [10.1017/9781316219317](https://doi.org/10.1017/9781316219317).
- [13] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase Gadget Synthesis for Shallow Circuits. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–228. Open Publishing Association, 2020. DOI: [10.4204/EPTCS.318.13](https://doi.org/10.4204/EPTCS.318.13).
- [14] Alexander Cowtan, Will Simmons, and Ross Duncan. A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz. *arXiv preprint arXiv:2007.10515*, 2020.

- [15] Raphael Dias da Silva and Ernesto F Galvão. Compact quantum circuits from one-way quantum computation. *Physical Review A*, 88(1):012319, 2013. DOI: [10.1103/physreva.88.012319](https://doi.org/10.1103/physreva.88.012319).
- [16] Raphael Dias da Silva, Einar Pius, and Elham Kashefi. Global quantum circuit optimization. *arXiv:1301.0351*, 2013.
- [17] V. Danos and E. Kashefi. Determinism in the one-way model. *Phys. Rev. A*, 74(052310), 2006. DOI: [10.1103/PhysRevA.74.052310](https://doi.org/10.1103/PhysRevA.74.052310).
- [18] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *J. ACM*, 54(2), April 2007. DOI: [10.1145/1219092.1219096](https://doi.org/10.1145/1219092.1219096).
- [19] Vincent Danos, Elham Kashefi, Prakash Panangaden, and Simon Perdrix. *Extended Measurement Calculus*, pages 235–310. Cambridge University Press, 2009. DOI: [10.1017/CBO9781139193313.008](https://doi.org/10.1017/CBO9781139193313.008).
- [20] Niel de Beaudrap. Unitary-circuit semantics for measurement-based computations. *International Journal of Quantum Information*, 8(01n02):1–91, 2010. DOI: [10.1142/s0219749910006113](https://doi.org/10.1142/s0219749910006113).
- [21] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to Reduce  $\pi/4$ -Parity-Phase Circuits, Motivated by the ZX Calculus. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 131–149. Open Publishing Association, 2020. DOI: [10.4204/EPTCS.318.9](https://doi.org/10.4204/EPTCS.318.9).
- [22] Olivia Di Matteo and Michele Mosca. Parallelizing quantum circuit synthesis. *Quantum Science and Technology*, 1(1):015003, 2016. DOI: [10.1088/2058-9565/1/1/015003](https://doi.org/10.1088/2058-9565/1/1/015003).
- [23] Ross Duncan and Simon Perdrix. Graph states and the necessity of Euler decomposition. *Mathematical Theory and Computational Practice*, pages 167–177, 2009. DOI: [10.1007/978-3-642-03073-4\\_18](https://doi.org/10.1007/978-3-642-03073-4_18).
- [24] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In *International Colloquium on Automata, Languages, and Programming*, pages 285–296. Springer, 2010. DOI: [10.1007/978-3-642-14162-1\\_24](https://doi.org/10.1007/978-3-642-14162-1_24).
- [25] Ross Duncan and Simon Perdrix. Pivoting makes the ZX-calculus complete for real stabilizers. In Bob Coecke and Matty Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic (QPL)*, volume 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 50–62. Open Publishing Association, 2014. DOI: [10.4204/EPTCS.171.5](https://doi.org/10.4204/EPTCS.171.5).
- [26] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, 6 2020. ISSN 2521-327X. DOI: [10.22331/q-2020-06-04-279](https://doi.org/10.22331/q-2020-06-04-279).
- [27] Maryam Eslamy, Mahboobeh Houshmand, Morteza Saheb Zamani, and Mehdi Sedighi. Optimization of one-way quantum computation measurement patterns. *International Journal of Theoretical Physics*, 57(11):3296–3317, 2018. DOI: [10.1007/s10773-018-3844-x](https://doi.org/10.1007/s10773-018-3844-x).
- [28] Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two Complete Axiomatisations of Pure-state Qubit Quantum Computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 502–511, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5583-4. DOI: [10.1145/3209108.3209128](https://doi.org/10.1145/3209108.3209128).
- [29] Nidhal Hamrit and Simon Perdrix. Reversibility in extended measurement-based

- quantum computation. In Jean Krivine and Jean-Bernard Stefani, editors, *Reversible Computation*, pages 129–138. Springer International Publishing, 2015. ISBN 978-3-319-20860-2. DOI: [10.1007/978-3-319-20860-2\\_8](https://doi.org/10.1007/978-3-319-20860-2_8).
- [30] M. Hein, J. Eisert, and H. J. Briegel. Multiparty entanglement in graph states. *Physical Review A*, 69(6):062311, 2004. DOI: [10.1103/PhysRevA.69.062311](https://doi.org/10.1103/PhysRevA.69.062311).
- [31] Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, M Nest, and H-J Briegel. Entanglement in graph states and its applications. In G. Casati, D. L. Shepelyansky, P. Zoller, and G. Benenti, editors, *Quantum Computers, Algorithms and Chaos*, volume 162 of *Proceedings of the International School of Physics "Enrico Fermi"*, pages 115–218. IOS Press Books, 2006. DOI: [10.3254/978-1-61499-018-5-115](https://doi.org/10.3254/978-1-61499-018-5-115).
- [32] Mahboobeh Houshmand, Mehdi Sedighi, Morteza Saheb Zamani, and Kourosh Marjoei. Quantum circuit synthesis targeting to improve one-way quantum computation pattern cost metrics. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(4):55, 2017. DOI: [10.1145/3064834](https://doi.org/10.1145/3064834).
- [33] Monireh Houshmand, Mahboobeh Houshmand, and Joseph F Fitzsimons. Minimal qubit resources for the realization of measurement-based quantum computation. *Physical Review A*, 98(1):012318, 2018. DOI: [10.1103/physreva.98.012318](https://doi.org/10.1103/physreva.98.012318).
- [34] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Diagrammatic Reasoning Beyond Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 569–578, New York, NY, USA, 2018. ACM. DOI: [10.1145/3209108.3209139](https://doi.org/10.1145/3209108.3209139).
- [35] 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, LICS '18*, pages 559–568, New York, NY, USA, 2018. ACM. DOI: [10.1145/3209108.3209131](https://doi.org/10.1145/3209108.3209131).
- [36] Aleks Kissinger and Arianne Meijer-van de Griend. CNOT circuit extraction for topologically-constrained quantum memories. *Quantum Information and Computation*, 20:581–596, 2020. DOI: [10.26421/QIC20.7-8](https://doi.org/10.26421/QIC20.7-8).
- [37] Aleks Kissinger and John van de Wetering. Universal MBQC with generalised parity-phase interactions and Pauli measurements. *Quantum*, 3:134, 2019. DOI: [10.22331/q-2019-04-26-134](https://doi.org/10.22331/q-2019-04-26-134).
- [38] Aleks Kissinger and John van de Wetering. PyZX: Large Scale Automated Diagrammatic Reasoning. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 229–241. Open Publishing Association, 2020. DOI: [10.4204/EPTCS.318.14](https://doi.org/10.4204/EPTCS.318.14).
- [39] Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Physical Review A, Vol.102-2*, 102:022406, 8 2020. DOI: [10.1103/PhysRevA.102.022406](https://doi.org/10.1103/PhysRevA.102.022406).
- [40] Vadym Kliuchnikov and Dmitri Maslov. Optimization of Clifford circuits. *Phys. Rev. A*, 88:052307, 2013. DOI: [10.1103/PhysRevA.88.052307](https://doi.org/10.1103/PhysRevA.88.052307).
- [41] Anton Kotzig. Eulerian lines in finite 4-valent graphs and their transformations. In *Colloquium on Graph Theory Tihany 1966*, pages 219–230. Academic Press, 1968.
- [42] Mehdi Mhalla and Simon Perdrix. Finding optimal flows efficiently. In *the 35th International Colloquium on Automata, Languages and Programming (ICALP), LNCS*, volume 5125, pages 857–868, 2008. DOI: [10.1007/978-3-540-70575-8\\_70](https://doi.org/10.1007/978-3-540-70575-8_70).
- [43] Mehdi Mhalla and Simon Perdrix. Graph states, pivot minor, and universality of

- (X, Z)-measurements. *International Journal of Unconventional Computing*, 9(1–2): 153–171, 2012.
- [44] Mehdi Mhalla, Mio Murao, Simon Perdrix, Masato Someya, and Peter S Turner. Which graph states are useful for quantum information processing? In *Conference on Quantum Computation, Communication, and Cryptography*, pages 174–187. Springer, 2011. DOI: [10.1007/978-3-642-54429-3\\_12](https://doi.org/10.1007/978-3-642-54429-3_12).
- [45] Jisho Miyazaki, Michal Hajdušek, and Mio Murao. Analysis of the trade-off between spatial and temporal resources for measurement-based quantum computation. *Physical Review A*, 91(5):052302, 2015. DOI: [10.1103/physreva.91.052302](https://doi.org/10.1103/physreva.91.052302).
- [46] Kang Feng Ng and Quanlong Wang. A universal completion of the ZX-calculus. *arXiv:1706.09877*, 2017.
- [47] M. A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. DOI: [10.1119/1.1463744](https://doi.org/10.1119/1.1463744).
- [48] R. Raussendorf, D.E. Browne, and H.J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(2):22312, 2003. DOI: [10.1103/physreva.68.022312](https://doi.org/10.1103/physreva.68.022312).
- [49] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, 2001. DOI: [10.1103/PhysRevLett.86.5188](https://doi.org/10.1103/PhysRevLett.86.5188).
- [50] M. Van den Nest, J. Dehaene, and B. De Moor. Graphical description of the action of local Clifford transformations on graph states. *Physical Review A*, 69(2):9422, 2004. DOI: [10.1103/physreva.69.022316](https://doi.org/10.1103/physreva.69.022316).
- [51] Renaud Vilmart. 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)*, pages 1–10, 2019. DOI: [10.1109/LICS.2019.8785765](https://doi.org/10.1109/LICS.2019.8785765).

## A Gflow and determinism

Here, we introduce the notation and concepts needed to state and prove the fact that gflow is a sufficient condition for uniform, strong and stepwise determinism of a measurement pattern. Then we give the detailed statement along with the proof.

Throughout this appendix,  $\simeq$  shall denote equality up to a global phase; that is, for linear maps  $L$  and  $K$ , we have  $L \simeq K$  iff there is an angle  $\alpha \in [0, 2\pi)$  such that  $L = e^{i\alpha}K$ .

Recall from Definition 2.33 that the linear map implemented by a particular set of measurement outcomes in a pattern is called a branch of the pattern. We begin by making this more precise.

**Definition A.1.** Let  $\mathfrak{P}$  be a measurement pattern with  $n$  measurement commands. A *branch* of the pattern is a sequence  $p$  of length  $n$  whose entries are binary digits 0 and 1 together with a linear map  $[[\mathfrak{P}]]_p$ , where each measurement is replaced with an outcome corresponding to either 0 or 1 as indicated by the corresponding entry in  $p$ .

If  $s$  is a binary sequence with  $k$  entries where  $k \leq n$ , we may also define an ‘intermediate pattern’  $[[\mathfrak{P}]]_s$  by replacing the first  $k$  measurements in  $\mathfrak{P}$  by an outcome corresponding to the entries in  $s$ .

We may now state various forms of determinism (Definition 2.33) using the notation of the above definition. A pattern  $\mathfrak{P}$  with  $n$  measurement commands is *deterministic* if for any branches  $p$  and  $q$  there is a complex number  $c$  such that  $[[\mathfrak{P}]]_p = c [[\mathfrak{P}]]_q$ . It is *strongly deterministic* if for any branches  $p$  and  $q$  we have  $[[\mathfrak{P}]]_p \simeq [[\mathfrak{P}]]_q$ . It is *stepwise deterministic* if the pattern  $[[\mathfrak{P}]]_s$  is deterministic for any binary sequence  $s$  with at most

$n$  entries. Recall that a pattern is *uniformly deterministic* if it is deterministic for any choice of measurement angles in the measurement commands appearing in  $\mathfrak{P}$ .

It will be useful to encode the measurement planes using following functions for any labelled open graph  $(G, I, O, \lambda)$ .

$$\begin{aligned}\beta_X, \beta_Z : \overline{O} &\rightarrow \{0, 1\} \\ \beta_X(i) &= \begin{cases} 1 & \text{if } \lambda(i) \in \{XY, XZ\} \\ 0 & \text{if } \lambda(i) = YZ \end{cases} \\ \beta_Z(i) &= \begin{cases} 1 & \text{if } \lambda(i) \in \{XZ, YZ\} \\ 0 & \text{if } \lambda(i) = XY \end{cases}\end{aligned}$$

The following observation will be the starting point for our proof of the main theorem.

**Lemma A.2.** Let  $s_i$  be the binary digit indicating the measurement outcome of qubit  $i$ . Then<sup>6</sup>

$$\left[ \left[ M_i^{\lambda(i), \alpha(i)} [X_i]^{\beta_Z(i)s_i} [Z_i]^{\beta_X(i)s_i} \right] \right]_p \simeq \left\langle +_{\lambda(i), \alpha(i)} \right\rangle_i$$

for both branches  $p$ .

*Proof.* If  $p = (0)$ , so that  $s_i = 0$ , this is clear, as the corrections vanish while there is nothing to correct. Hence suppose  $p = (1)$ . We have three cases:

1.  $\lambda(i) = XY$ , so that  $\beta_Z(i) = 0$  and  $\beta_X(i) = 1$ ,
2.  $\lambda(i) = XZ$ , so that  $\beta_Z(i) = \beta_X(i) = 1$ ,
3.  $\lambda(i) = YZ$ , so that  $\beta_Z(i) = 1$  and  $\beta_X(i) = 0$ .

Recalling that the Pauli operators act on the basis qubits as

$$\begin{aligned}X|0\rangle &= |1\rangle & X|1\rangle &= |0\rangle \\ Z|0\rangle &= -|0\rangle & Z|1\rangle &= |1\rangle\end{aligned}$$

we compute in each case, starting from the (dual of the) left-hand side:

1. 
$$\begin{aligned}Z_i(|0\rangle - e^{i\alpha(i)}|1\rangle) &= -(|0\rangle + e^{i\alpha(i)}|1\rangle) \\ &\simeq \left| +_{XY, \alpha(i)} \right\rangle_i,\end{aligned}$$
2. 
$$\begin{aligned}Z_i X_i \left( \sin\left(\frac{\alpha(i)}{2}\right)|0\rangle - \cos\left(\frac{\alpha(i)}{2}\right)|1\rangle \right) &= \sin\left(\frac{\alpha(i)}{2}\right)|1\rangle + \cos\left(\frac{\alpha(i)}{2}\right)|0\rangle \\ &= \left| +_{XZ, \alpha(i)} \right\rangle_i,\end{aligned}$$
3. 
$$\begin{aligned}X_i \left( \sin\left(\frac{\alpha(i)}{2}\right)|0\rangle - i \cos\left(\frac{\alpha(i)}{2}\right)|1\rangle \right) &= \sin\left(\frac{\alpha(i)}{2}\right)|1\rangle - i \cos\left(\frac{\alpha(i)}{2}\right)|0\rangle \\ &\simeq \cos\left(\frac{\alpha(i)}{2}\right)|0\rangle + i \sin\left(\frac{\alpha(i)}{2}\right)|1\rangle \\ &= \left| +_{YZ, \alpha(i)} \right\rangle_i. \quad \square\end{aligned}$$

---

<sup>6</sup>Note that the pattern below is not runnable, as the corrections depend on a future measurement outcome.

**Definition A.3.** Let  $(G, I, O)$  be an open graph. View  $(V, I, O)$  as the register of a measurement pattern as in Definition 2.7. Define the following command for each  $i \in \bar{I}$ :

$$K_i := X_i \prod_{j \in N_G(i)} Z_j.$$

We call  $K_i$  the *graph stabiliser* at qubit  $i$  for reasons explained below. Observe that the commands  $Z_j$  in the product act on different qubits and hence commute with each other. Thus the product does not depend on the order, so that  $K_i$  is well-defined.

The reason for calling the  $K_i$  graph stabilisers is the following lemma.

**Lemma A.4** ([9, pp. 6-7]). For any open graph  $(G, I, O)$  and for any  $i \in \bar{I}$  with graph stabiliser  $K_i$  we have

$$K_i E_G N_{\bar{I}} = E_G N_{\bar{I}}.$$

**Notation:** Given a command  $C$  in a measurement pattern with register  $V$ , for any subset of qubits  $S \subseteq V$  we write  $C_S := \prod_{i \in S} C_i$  if the right-hand side is well-defined (in particular, if it does not depend on the order in which the different operations are applied). If all the  $C_i$ 's are moreover Pauli  $X$  or  $Z$  operators, and  $s_i$  is a binary digit corresponding to the measurement outcome of qubit  $i$ , we may treat  $C_S$  as a joint correction by writing  $C_S^{s_i} := \prod_{i \in S} [C_i]^{s_i}$ . Note that the graph stabiliser may now be written as  $K_i = X_i Z_{N_G(i)}$ .

We are now ready to state and prove the central theorem about gflow.

**Theorem A.5.** Let  $\Gamma = (G, I, O, \lambda)$  be a labelled open graph. If  $\Gamma$  has a gflow  $\mathfrak{g} = (g, \prec)$ , then for any set of measurement angles  $\alpha : \bar{O} \rightarrow [0, 2\pi)$  the following pattern is runnable as well as strongly and stepwise deterministic:

$$\mathfrak{P}_{\Gamma, \mathfrak{g}, \alpha} := \prod_{i \in \bar{O}}^{\prec} \left( X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i)) \setminus \{i\}}^{s_i} M_i^{\lambda(i), \alpha(i)} \right) E_G N_{\bar{I}}, \quad (14)$$

where  $\prod^{\prec}$  denotes concatenation of commands following the order  $\prec$ . Moreover,  $\mathfrak{P}_{\Gamma, \mathfrak{g}, \alpha}$  realizes the linear map associated with  $\Gamma$  at angles  $\alpha$  (Definition 2.12) in the sense that for any branch  $p$  we have

$$\llbracket \mathfrak{P}_{\Gamma, \mathfrak{g}, \alpha} \rrbracket_p \simeq \prod_{i \in \bar{O}} \left\langle +_{\lambda(i), \alpha(i)} \middle|_i E_G N_{\bar{I}}. \quad (15)$$

**Remark A.6.** Note that the pattern in equation (14) is uniformly deterministic, as it is (strongly) deterministic independently of the choice of  $\alpha$ . It can be shown [19, Proposition 7.3.4] that any strongly deterministic pattern implements a unitary embedding. Thus the map in equation (15) is guaranteed to be one. In particular, if  $|I| = |O|$ , then the map is unitary.

*Proof.* First observe that  $\mathfrak{P}_{\Gamma, \mathfrak{g}, \alpha}$  is indeed runnable; the conditions (g1) and (g2) ensure that no command acts on a qubit that has been measured<sup>7</sup>, and all the other conditions in Definition 2.8 are easily verified from the shape of the pattern.

For strong determinism, it is sufficient to show the claimed equality (15), as this shows that any two branches are equal (up to a global phase). The proof of the equality is based

<sup>7</sup>We remark that the condition (4) would fail to guarantee this.

on Lemma A.2 and the following observation. Using the fact that the Pauli matrices are involutive, we show that for all  $i \in \bar{O}$ ,

$$\begin{aligned} \prod_{u \in g(i)} Z_{\text{Even}(g(i)) \cap N_G(u)} &= I, \\ \prod_{u \in g(i)} Z_{\text{Odd}(g(i)) \cap N_G(u)} &= Z_{\text{Odd}(g(i))}, \end{aligned} \quad (16)$$

where  $I$  is the identity matrix. The first equality follows by taking  $u \in g(i)$  and  $j \in N_G(u)$  such that  $j$  is in the even neighbourhood of  $g(i)$ . Then the number of neighbours of  $j$  in  $g(i)$  is even, so that  $Z_j$  appears in the product an even number of times. For the second equality, take  $u \in g(i)$  and  $j \in N_G(u)$  such that  $j$  is in the odd neighbourhood of  $g(i)$ . Then the number of neighbours of  $j$  in  $g(i)$  is odd, so that  $Z_j$  appears in the product an odd number of times.

Thus we compute from equation (14):

$$\begin{aligned} & \llbracket \mathfrak{P}_{\Gamma, \mathfrak{g}, \alpha} \rrbracket_p \\ &= \prod_{i \in \bar{O}} \left[ \left[ M_i^{\lambda(i), \alpha(i)} X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i)) \setminus \{i\}}^{s_i} \right] \right]_p E_G N_{\bar{I}} \end{aligned} \quad (\text{s1})$$

$$= \prod_{i \in \bar{O}} \left[ \left[ M_i^{\lambda(i), \alpha(i)} [X_i]^{2\beta_Z(i)s_i} [Z_i]^{2\beta_X(i)s_i} X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i)) \setminus \{i\}}^{s_i} \right] \right]_p E_G N_{\bar{I}} \quad (\text{s2})$$

$$\simeq \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i [X_i]^{\beta_Z(i)s_i} [Z_i]^{\beta_X(i)s_i} X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i)) \setminus \{i\}}^{s_i} \right\rangle E_G N_{\bar{I}} \right) \quad (\text{s3})$$

$$\simeq \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i X_{g(i)}^{s_i} Z_{\text{Odd}(g(i))}^{s_i} \right\rangle E_G N_{\bar{I}} \right) \quad (\text{s4})$$

$$= \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i \prod_{u \in g(i)} [X_u]^{s_i} \prod_{v \in g(i)} Z_{\text{Odd}(g(i)) \cap N_G(v)}^{s_i} \prod_{w \in g(i)} Z_{\text{Even}(g(i)) \cap N_G(w)}^{s_i} \right\rangle E_G N_{\bar{I}} \right) \quad (\text{s5})$$

$$\simeq \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i \prod_{u \in g(i)} \left( [X_u]^{s_i} Z_{\text{Odd}(g(i)) \cap N_G(u)}^{s_i} Z_{\text{Even}(g(i)) \cap N_G(u)}^{s_i} \right) \right\rangle E_G N_{\bar{I}} \right) \quad (\text{s6})$$

$$= \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i \prod_{u \in g(i)} \left( [X_u]^{s_i} Z_{N_G(u)}^{s_i} \right) \right\rangle E_G N_{\bar{I}} \right)$$

$$= \prod_{i \in \bar{O}} \left( \left\langle +_{\lambda(i), \alpha(i)} \middle|_i \prod_{u \in g(i)} K_u^{s_i} \right\rangle E_G N_{\bar{I}} \right)$$

$$= \prod_{i \in \bar{O}} \left\langle +_{\lambda(i), \alpha(i)} \middle|_i E_G N_{\bar{I}} \right\rangle. \quad (\text{s7})$$

Here we used:

(s1) The fact that the correction terms do not act on the vertex  $i$ .

(s2) Involutivity of the Pauli matrices.

(s3) Lemma A.2 and that  $X_i$  and  $Z_i$  anti-commute.

- (s4) Conditions (g3)-(g5). For example, if  $\lambda(i) = YZ$ , then  $\beta_Z(i) = 1$  and by (g5)  $i \in g(i)$ , so that  $[X_i]^{s_i} X_{g(i)\setminus\{i\}}^{s_i} = X_{g(i)}^{s_i}$ , while  $\beta_X(i) = 0$  and again by (g5)  $i \notin \text{Odd}(g(i))$ , so that  $Z_{\text{Odd}(g(i))\setminus\{i\}} = Z_{\text{Odd}(g(i))}$ . The cases of other measurement planes are similar.
- (s5) Equations (16) and the definition of  $X_{g(i)}^{s_i}$ .
- (s6) The facts that  $X_i$  and  $Z_i$  anti-commute, and that commands acting on different qubits commute.
- (s7) Lemma A.4 and (g1) in order to commute the graph stabilisers past the linear maps.

Thus the pattern is strongly deterministic.

It remains to show that  $\mathfrak{P}_{\Gamma, g, \alpha}$  is stepwise deterministic. Thus consider a subgraph of  $\Gamma$  (with  $I$ ,  $O$  and  $\lambda$  restricted appropriately) obtained by removing some minimal vertices in the order  $\prec$ . The interpretation is that those vertices have already been measured. By conditions (g1) and (g2), for any vertex  $i$  in the subgraph, the correction set  $g(i)$  and its odd neighbourhood do not contain any of the removed vertices, and are thus themselves contained in the subgraph. Therefore we may restrict  $g$  to the subgraph, and the odd neighbourhood of each  $g(i)$  in the subgraph will be equal to its odd neighbourhood in the full graph. Thus the conditions (g3), (g4) and (g5) are satisfied by the subgraph. If we also restrict the partial order to the subgraph, we find that the conditions (g1) and (g2) are satisfied. Hence the gflow of the full graph restricts to a gflow on the subgraph. Therefore, since the measurements in  $\mathfrak{P}_{\Gamma, g, \alpha}$  follow the order  $\prec$ , we may think of each measurement as removing a minimal vertex in  $\Gamma$ . As we have just seen, the graph with some minimal vertices removed has a gflow given by restriction of the original gflow, so that repeating the preceding parts of the proof will yield another (strongly) deterministic pattern. This pattern is equal to the proper sub-pattern of  $\mathfrak{P}_{\Gamma, g, \alpha}$  which does not involve the already-performed measurement and its corresponding corrections. Therefore, the original pattern is stepwise deterministic.  $\square$

## B Proofs for Section 3

The following lemma records how the odd neighbourhood of an arbitrary set of vertices changes under local complementation.

**Lemma B.1.** ([26, Lemma B.4]) Given a graph  $G = (V, E)$ , a subset  $A \subseteq V$  and a vertex  $u \in V$ , we have

$$\text{Odd}_{G \star u}(A) = \begin{cases} \text{Odd}_G(A) \Delta (N_G(u) \cap A) & \text{if } u \notin \text{Odd}_G(A) \\ \text{Odd}_G(A) \Delta (N_G(u) \setminus A) & \text{if } u \in \text{Odd}_G(A). \end{cases}$$

Now we can prove the main lemma.

**Lemma 3.1 (restated).** Let  $(g, \prec)$  be a gflow for  $(G, I, O, \lambda)$  and let  $u \in \overline{O} \cap \overline{I}$ . Then  $(g', \prec)$  is a gflow for  $(G \star u, I, O, \lambda')$ , where

$$\lambda'(u) := \begin{cases} XZ & \text{if } \lambda(u) = XY \\ XY & \text{if } \lambda(u) = XZ \\ YZ & \text{if } \lambda(u) = YZ \end{cases}$$



and for all  $v \in \overline{O} \setminus \{u\}$

$$\lambda'(v) := \begin{cases} \text{YZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{XZ} \\ \text{XZ} & \text{if } v \in N_G(u) \text{ and } \lambda(v) = \text{YZ} \\ \lambda(v) & \text{otherwise.} \end{cases}$$

Furthermore,

$$g'(u) := \begin{cases} g(u) \Delta \{u\} & \text{if } \lambda(u) \in \{\text{XY}, \text{XZ}\} \\ g(u) & \text{if } \lambda(u) = \text{YZ} \end{cases}$$

and for all  $v \in \overline{O} \setminus \{u\}$ ,

$$g'(v) := \begin{cases} g(v) & \text{if } u \notin \text{Odd}_G(g(v)) \\ g(v) \Delta g'(u) \Delta \{u\} & \text{if } u \in \text{Odd}_G(g(v)). \end{cases}$$

*Proof.* We divide the proof into three parts, first proving that  $g'(u)$  satisfies the required properties, then proving that  $g'(v)$  for  $v \neq u$  satisfies (g1) and (g2), and finally showing that  $g''(v)$  satisfies the conditions (g3)–(g5). Each part is further subdivided based on the values of  $\lambda(u)$  or  $\lambda(v)$ .

**Part 1:** Note that condition (g1) is trivially satisfied by  $g'(u)$ . We now show that  $g'(u)$  satisfies the remaining conditions, subdividing into two cases based on  $\lambda(u)$ .

**Case 1a:** If  $\lambda(u) \in \{\text{XY}, \text{XZ}\}$ , then  $u \in \text{Odd}_G(g(u))$  and  $\lambda'(u) \in \{\text{XY}, \text{XZ}\}$  with  $\lambda'(u) \neq \lambda(u)$ . We have  $\text{Odd}_{G^*u}(\{u\}) = N_{G^*u}(u) = N_G(u)$ . Thus

$$\begin{aligned} \text{Odd}_{G^*u}(g'(u)) &= \text{Odd}_{G^*u}(g(u) \Delta \{u\}) \\ &= \text{Odd}_{G^*u}(g(u)) \Delta \text{Odd}_{G^*u}(\{u\}) \\ &= \text{Odd}_G(g(u)) \Delta (N_G(u) \setminus g(u)) \Delta N_G(u) \\ &= \text{Odd}_G(g(u)) \Delta (N_G(u) \cap g(u)), \end{aligned} \tag{17}$$

where the third equality uses Lemma B.1. Hence we have

- $u \in \text{Odd}_{G^*u}(g'(u))$  since  $u \in \text{Odd}_G(g(u))$  and  $u \notin N_G(u)$ ,
- $u \in g'(u)$  if and only if  $u \notin g(u)$  (this is desired because  $\lambda'(u) \neq \lambda(u)$ ); together with the previous item this yields Condition (g3) or (g4), as appropriate,
- if  $v \in \text{Odd}_{G^*u}(g'(u))$ , then either  $v \in \text{Odd}_G(g(u))$  or  $v \in g(u)$ , so either  $u = v$  or  $u \prec v$ ; this is Condition (g2).

So  $g'(u)$  indeed satisfies all the required conditions.

**Case 1b:** If  $\lambda(u) = \text{YZ}$ , then  $u \notin \text{Odd}_G(g(u))$ , hence

$$\text{Odd}_{G^*u}(g'(u)) = \text{Odd}_{G^*u}(g(u)) = \text{Odd}_G(g(u)) \Delta (N_G(u) \cap g(u)), \tag{18}$$

again using Lemma B.1. This implies

- $u \notin \text{Odd}_{G^*u}(g'(u))$  because  $u \notin \text{Odd}_G(g(u))$  and  $u \notin N_G(u)$ ,
- $u \in g'(u)$  because  $u \in g(u)$ ; together with the previous item this yields Condition (g5),
- if  $v \in \text{Odd}_{G^*u}(g'(u))$ , then either  $v \in \text{Odd}_G(g(u))$  or  $v \in g(u)$ , so either  $u = v$  or  $u \prec v$ ; this is Condition (g2).

So  $g'(u)$  indeed satisfies all the required conditions.

**Part 2:** We show that the correction set  $g(v)$  for any  $v \neq u$  satisfies Conditions (g1) and (g2) of the gflow definition. The proof is split into subcases depending on whether  $u \in \text{Odd}_G(g(v))$ .

**Case 2a:** If  $u \notin \text{Odd}_G(g(v))$ , first note that

$$\text{Odd}_{G^{\star u}}(g'(v)) = \text{Odd}_{G^{\star u}}(g(v)) = \text{Odd}_G(g(v)) \Delta (N_G(u) \cap g(v))$$

Hence if  $w \in \text{Odd}_{G^{\star u}}(g(v))$ , then either  $w \in \text{Odd}_G(g(v))$  or  $w \in g(v)$ , so  $v = w$  or  $v \prec w$ ; this is Condition (g2). Condition (g1) is trivially still satisfied.

**Case 2b:** If  $u \in \text{Odd}_G(g(v))$ , first note that  $v \prec u$ . If  $w \in g'(v)$ , then  $w \in g(v)$  or  $w \in g'(u)$  or  $w = u$ , and in each case either  $w = v$  or  $v \prec w$ , this is Condition (g1). Furthermore, we have

$$\begin{aligned} \text{Odd}_{G^{\star u}}(g'(v)) &= \text{Odd}_{G^{\star u}}(g(v) \Delta g'(u) \Delta \{u\}) \\ &= \text{Odd}_{G^{\star u}}(g(v)) \Delta \text{Odd}_{G^{\star u}}(g'(u)) \Delta N_G(u) \\ &= \text{Odd}_G(g(v)) \Delta (N_G(u) \setminus g(v)) \Delta \text{Odd}_G(g(u)) \Delta (N_G(u) \cap g(u)) \Delta N_G(u) \\ &= \text{Odd}_G(g(v)) \Delta (N_G(u) \cap g(v)) \Delta \text{Odd}_G(g(u)) \Delta (N_G(u) \cap g(u)) \end{aligned}$$

where the third step uses the property that  $\text{Odd}_{G^{\star u}}(g'(u)) = \text{Odd}_G(g(u)) \Delta (N_G(u) \cap g(u))$  for any  $\lambda(u)$ , which follows from combining (17) and (18). Hence if  $w \in \text{Odd}_{G^{\star u}}(g'(v))$ , then at least one of the following holds:

- $w \in \text{Odd}_G(g(v))$ , so  $v = w$  or  $v \prec w$
- $w \in g(v)$ , so  $v = w$  or  $v \prec w$
- $w \in \text{Odd}_G(g(u))$ , so  $u = w$  or  $u \prec w$ ; in both cases  $v \prec w$  since  $v \prec u$
- $w \in g(u)$ , so  $u = w$  or  $u \prec w$ ; in both cases  $v \prec w$  since  $v \prec u$ .

In each case, Condition (g2) is satisfied.

**Part 3:** Finally we show that the correction set  $g(v)$  for any  $v \neq u$  satisfies Conditions (g3), (g4) or (g5) of gflow. The proof is split into subcases depending on whether  $v \in N_G(u)$ .

**Case 3a:** Suppose  $v \in N_G(u)$  and distinguish cases according to  $\lambda(v)$ .

- Suppose  $\lambda(v) = XY$ , then  $v \notin g(v)$  and  $v \in \text{Odd}_G(g(v))$ . Furthermore,  $\lambda'(v) = XY$ . We have
  - $v \in \text{Odd}_{G^{\star u}}(g(v))$  since  $v \in \text{Odd}_G(g(v))$  and  $v \notin g(v)$ ,
  - $v \notin g'(v)$ ,

which together give Condition (g3).

- Suppose  $\lambda(v) = XZ$ , then  $v \in g(v)$  and  $v \in \text{Odd}_G(g(v))$ . Furthermore,  $\lambda'(v) = YZ$ . We have
  - $v \notin \text{Odd}_{G^{\star u}}(g(v))$  since  $v \in \text{Odd}_G(g(v))$  and  $v \in N_G(u) \cap g(v)$ ,
  - $v \in g'(v)$ ,

which together give Condition (g5).

- Suppose  $\lambda(v) = YZ$ , then  $v \in g(v)$  and  $v \notin \text{Odd}_G(g(v))$ . Furthermore,  $\lambda'(v) = XZ$ . We have

- $v \in \text{Odd}_{G \star u}(g(v))$  since  $v \notin \text{Odd}_G(g(v))$  and  $v \in N_G(u) \cap g(v)$ ,
- $v \in g'(v)$ ,

which together give Condition (g4).

**Case 3b:** Suppose  $v \notin N_G(u)$  and distinguish cases according to  $\lambda(v)$ .

- Suppose  $\lambda(v) = XY$ , this case is analogous to the corresponding one in Subcase 3a, so Condition (g3) is satisfied.
- Suppose  $\lambda(v) = XZ$ , then  $v \in g(v)$  and  $v \in \text{Odd}_G(g(v))$ . Furthermore,  $\lambda'(v) = XZ$ . We have

- $v \in \text{Odd}_{G \star u}(g(v))$  since  $v \in \text{Odd}_G(g(v))$  and  $v \notin N_G(u) \cap g(v)$ ,
- $v \in g'(v)$ ,

which together give Condition (g4).

- Suppose  $\lambda(v) = YZ$ , then  $v \in g(v)$  and  $v \notin \text{Odd}_G(g(v))$ . Furthermore,  $\lambda'(v) = YZ$ . We have

- $v \notin \text{Odd}_{G \star u}(g(v))$  since  $v \notin \text{Odd}_G(g(v))$  and  $v \notin N_G(u) \cap g(v)$ ,
- $v \in g'(v)$ ,

which together give Condition (g5). □

**Corollary 3.3 (restated).** Let  $(G, I, O, \lambda)$  be a labelled open graph which has a gflow, and let  $u, v \in \overline{O} \cap \overline{I}$  be connected by an edge. Then  $(G \wedge uv, I, O, \hat{\lambda})$ , where

$$\hat{\lambda}(a) = \begin{cases} YZ & \text{if } \lambda(a) = XY \\ XZ & \text{if } \lambda(a) = XZ \\ XY & \text{if } \lambda(a) = YZ \end{cases}$$

for  $a \in \{u, v\}$ , and  $\hat{\lambda}(w) = \lambda(w)$  for all  $w \in \overline{O} \setminus \{u, v\}$  also has a gflow.

*Proof.* Recall that  $G \wedge uv = G \star u \star v \star u$ . Hence by applying Lemma 3.1 three times we get the correct underlying graph, which has a gflow. It remains to show that the labels change in the manner stated. Let us denote the labelling function associated with  $G \star u$  by  $\lambda'$  (as in Lemma 3.1). Similarly, we denote the labelling and correction set functions for  $G \star u \star v$  by  $\lambda''$  and  $g''$ . We then compute:

$$\lambda''(v) = \begin{cases} XZ & \text{if } \lambda'(v) = XY \text{ iff } \lambda(v) = XY \\ XY & \text{if } \lambda'(v) = XZ \text{ iff } \lambda(v) = YZ \\ YZ & \text{if } \lambda'(v) = YZ \text{ iff } \lambda(v) = XZ, \end{cases}$$

so that

$$\hat{\lambda}(v) = \begin{cases} XY & \text{if } \lambda''(v) = XY \text{ iff } \lambda(v) = YZ \\ YZ & \text{if } \lambda''(v) = XZ \text{ iff } \lambda(v) = XY \\ XZ & \text{if } \lambda''(v) = YZ \text{ iff } \lambda(v) = XZ, \end{cases}$$

and by symmetry the same holds for  $u$ . For  $w \in \overline{O} \setminus \{u, v\}$  we have

$$\lambda''(w) = \begin{cases} YZ & \text{if } w \in N_{G \star u}(v) \text{ and } \lambda'(w) = XZ \\ XZ & \text{if } w \in N_{G \star u}(v) \text{ and } \lambda'(w) = YZ \\ \lambda'(w) & \text{otherwise.} \end{cases}$$

Note that  $w \in N_{G \star u}(v)$  iff  $w \in N_G(v) \Delta N_G(u)$ , whence, denoting complementation in the vertex set by an overline, we obtain

$$\lambda''(w) = \begin{cases} \text{YZ} & \text{if } \begin{array}{l} w \in \overline{N_G(v)} \cap N_G(u), \lambda(w) = \text{YZ}, \text{ or} \\ w \in N_G(v) \cap \overline{N_G(u)}, \lambda(w) = \text{XZ}, \text{ or} \\ w \in N_G(v) \cap N_G(u), \lambda(w) = \text{XZ} \end{array} \\ \text{XZ} & \text{if } \begin{array}{l} w \in \overline{N_G(v)} \cap N_G(u), \lambda(w) = \text{XZ}, \text{ or} \\ w \in N_G(v) \cap \overline{N_G(u)}, \lambda(w) = \text{YZ}, \text{ or} \\ w \in N_G(v) \cap N_G(u), \lambda(w) = \text{YZ} \end{array} \\ \lambda(w) & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \text{YZ} & \text{if } w \in N_G(v) \text{ and } \lambda(w) = \text{XZ} \\ \text{XZ} & \text{if } w \in N_G(v) \text{ and } \lambda(w) = \text{YZ} \\ \lambda(w) & \text{otherwise.} \end{cases}$$

Similarly, we have

$$\hat{\lambda}(w) = \begin{cases} \text{YZ} & \text{if } w \in N_{G \star u \star v}(u) \text{ and } \lambda''(w) = \text{XZ} \\ \text{XZ} & \text{if } w \in N_{G \star u \star v}(u) \text{ and } \lambda''(w) = \text{YZ} \\ \lambda''(w) & \text{otherwise.} \end{cases}$$

Noting that  $w \in N_{G \star u \star v}(u)$  iff  $w \in N_{G \star u}(u) \Delta N_{G \star u}(v)$  iff  $w \in N_G(v)$  we get

$$\hat{\lambda}(w) = \begin{cases} \text{YZ} & \text{if } w \in N_G(v) \text{ and } \lambda(w) = \text{YZ} \\ \text{XZ} & \text{if } w \in N_G(v) \text{ and } \lambda(w) = \text{XZ} \\ \lambda(w) & \text{otherwise,} \end{cases}$$

which reduces to  $\hat{\lambda}(w) = \lambda(w)$ . □

## C Finding maximally delayed gflow in multiple measurement planes

The original algorithm for finding a maximally delayed gflow from Ref. [42] is restricted to patterns where all measurements are in the XY-plane. Here we extend this procedure to measurements in the YZ and XZ-planes, in the manner of Ref. [9]. We will always assume that any connected component of a labelled open graph contains an input or an output. This is because any connected component containing neither inputs nor outputs contributes only an irrelevant scalar factor to a measurement-based computation. In the presence of gflow, this assumption implies the following stronger property.

**Lemma C.1.** Let  $(G, I, O, \lambda)$  be a labelled open graph with the property that any connected component of  $G$  intersects  $I \cup O$ . Suppose furthermore that  $(G, I, O, \lambda)$  has gflow. Then any isolated vertex is an output.

*Proof.* Let  $(g, \prec)$  be a gflow on  $(G, I, O, \lambda)$ , and let  $v \in V$  be arbitrary. We will show that either

- $v \in O$ , or
- $v \in \overline{O}$  and there exists  $w \in V$  with  $w \sim v$ .

In the first case, there is nothing to prove. In the second case, consider the different measurement planes.

- Suppose  $\lambda(v) = XY$ , then  $v \notin g(v)$  and  $v \in \text{Odd}(g(v))$  by (g3). This implies there exists  $w \in g(v) \cap N_G(v)$ .
- Suppose  $\lambda(v) = XZ$ , then  $v \in g(v)$  and  $v \in \text{Odd}(g(v))$  by (g4). This again implies that there exists  $w \in g(v) \cap N_G(v)$ .
- Finally, suppose  $\lambda(v) = YZ$ , then  $v \in g(v)$  and  $v \notin \text{Odd}(g(v))$  by (g5). Note that  $v \notin I$  since inputs cannot appear in correction sets. Also,  $v$  is not an output, but its connected component must intersect  $I \cup O$ , so  $v$  must have a neighbour  $w$ .  $\square$

**Lemma C.2** (Generalisation of [42, Lemma 1] to multiple measurement planes). If  $(g, \prec)$  is a maximally delayed gflow of  $(G, I, O, \lambda)$ , then  $V_0^\prec = O$ .

*Proof.* To show that  $V_0^\prec \subseteq O$ : Assume towards a contradiction that there exists  $u \in V_0^\prec \setminus O$ . Then  $u \in \max_\prec V$  by Definition 3.9 and  $g(u)$  is either  $\{u\}$  or  $\emptyset$  by (g1) of Definition 2.36.

- If  $g(u) = \emptyset$  then  $u \notin \text{Odd}(g(u))$  and  $u \notin g(u)$ , contradicting all of (g3)–(g5), so the labelled open graph could not have gflow.
- Thus  $g(u) = \{u\}$ . Now, since  $u$  is not an output, Lemma C.1 implies  $u$  has a neighbour; call this neighbour  $v$ . But  $u \in g(u)$  and  $v \sim u$  imply  $v \in \text{Odd}(g(u))$  and therefore  $u \prec v$  by (g2).

This contradicts the property that  $u \in \max_\prec V$ . Therefore such a  $u$  cannot exist and instead  $V_0^\prec \subseteq O$ .

To show that  $O \subseteq V_0^\prec$  we assume that  $O \setminus V_0^\prec \neq \emptyset$ . Let  $\prec' = \prec \setminus (O \times V)$ , which considers all outputs to be maximal. Then  $(g, \prec')$  is a gflow of  $(G, I, O, \lambda)$ :

- Conditions (g1) and (g2) of Definition 2.36 are satisfied by  $\prec'$  because elements of  $O$  are not in the domain of  $g$
- Conditions (g3)–(g5) are unaffected because the image of  $g$  has not changed.

Thus,  $(g, \prec')$  is a gflow of  $(G, I, O, \lambda)$ . Moreover, for any  $k$ ,  $\bigcup_{i=1}^k V_i^\prec \subseteq \bigcup_{i=1}^k V_i^{\prec'}$ , and  $|V_0^\prec| < |V_0^{\prec'}|$ , thus  $(g, \prec')$  is more delayed than  $(g, \prec)$ , which is a contradiction.  $\square$

**Lemma C.3** (Generalisation of [42, Lemma 2] to multiple measurement planes). If  $(g, \prec)$  is a maximally delayed gflow of  $(G, I, O, \lambda)$ , then  $(\tilde{g}, \tilde{\prec})$  is a maximally delayed gflow of  $(G, I, O \cup V_1^\prec, \lambda)$ , where  $\tilde{g}$  is the restriction of  $g$  to the domain  $V \setminus (V_0^\prec \cup V_1^\prec)$  and  $\tilde{\prec} = \prec \setminus (V_1^\prec \times V_0^\prec)$ .

*Proof.* First show that  $(\tilde{g}, \tilde{\prec})$  is a gflow:

- (g1) for  $\tilde{\prec}$ : If  $v \in V \setminus (V_0^\prec \cup V_1^\prec)$  and  $w \in \tilde{g}(v)$ , then  $w \in g(v)$  and thus  $v = w$  or  $v \prec w$  ((g1) for  $\prec$ .) Therefore  $v = w$  or  $v \tilde{\prec} w$ .
- (g2) for  $\tilde{\prec}$ : If  $v \in V \setminus (V_0^\prec \cup V_1^\prec)$  and  $w \in \text{Odd}(\tilde{g}(v))$ , then  $w \in \text{Odd}(g(v))$  and thus  $v = w$  or  $v \prec w$  ((g2) for  $\prec$ .) Again  $v = w$  or  $v \tilde{\prec} w$ .
- (g3), (g4), and (g5) for  $\tilde{\prec}$ : Since  $\tilde{g} = g$  on  $V \setminus (V_0^\prec \cup V_1^\prec)$ , these conditions still hold for all  $v \in V \setminus (V_0^\prec \cup V_1^\prec)$ .

Now to show that  $(\tilde{g}, \tilde{\prec})$  is maximally delayed: Assume for a contradiction that there exists a gflow  $(g', \prec')$  which is more delayed than  $(\tilde{g}, \tilde{\prec})$ , i.e. that  $|\bigcup_{i=0}^k V_i^{\prec'}| \geq |\bigcup_{i=0}^k V_i^{\tilde{\prec}}|$ , where the inequality is strict at some  $k = j$ . Extend  $(g', \prec')$  on  $(G, I, O \cup V_1^{\prec}, \lambda)$  to  $(g'', \prec'')$  on  $(G, I, O, \lambda)$  where

$$g''(v) = \begin{cases} g'(v) & \text{if } v \in V \setminus (V_0^{\prec} \cup V_1^{\prec}) \\ g(v) & \text{if } v \in V_1^{\prec} \end{cases} \quad (19)$$

$$\prec'' = \prec' \cup \{(u, v) \mid u \in V_1^{\prec} \wedge u \prec v\} \quad (20)$$

We can relate  $(g', \prec')$  to  $(g'', \prec'')$  and  $(\tilde{g}, \tilde{\prec})$  to  $(g, \prec)$  via

$$\bigcup_{i=0}^k V_i^{\prec'} = \bigcup_{i=0}^{k+1} V_i^{\prec''} \quad \bigcup_{i=0}^k V_i^{\tilde{\prec}} = \bigcup_{i=0}^{k+1} V_i^{\prec} \quad V_0^{\prec} = V_0^{\prec''} \quad (21)$$

Therefore  $|\bigcup_{i=0}^k V_i^{\prec''}| \geq |\bigcup_{i=0}^k V_i^{\prec}|$  and the inequality is strict at  $k = j + 1$ . This is a contradiction of maximality of  $(g, \prec)$ , so  $(\tilde{g}, \tilde{\prec})$  must be maximally delayed.  $\square$

**Lemma C.4** (Generalisation of [42, Lemma 3] to multiple measurement planes). If  $(g, \prec)$  is a maximally delayed gflow of a labelled open graph  $(G, I, O, \lambda)$ , then  $V_1^{\prec} = V_1^{\prec, XY} \cup V_1^{\prec, XZ} \cup V_1^{\prec, YZ}$ , where

$$V_1^{\prec, XY} = \{u \in \bar{O} \mid \lambda(u) = XY \wedge \exists K \subseteq O \cap \bar{I} \text{ s.t. } \text{Odd}(K) \cap \bar{O} = \{u\}\}$$

$$V_1^{\prec, XZ} = \{u \in \bar{O} \mid \lambda(u) = XZ \wedge \exists K \subseteq O \cap \bar{I} \text{ s.t. } \text{Odd}(K \cup \{u\}) \cap \bar{O} = \{u\}\}$$

$$V_1^{\prec, YZ} = \{u \in \bar{O} \mid \lambda(u) = YZ \wedge \exists K \subseteq O \cap \bar{I} \text{ s.t. } \text{Odd}(K \cup \{u\}) \cap \bar{O} = \emptyset\}.$$

*Proof.* We first show that if  $(g, \prec)$  is a maximally delayed gflow, then for any  $u \in V_1^{\prec}$ , we have  $g(u) \subseteq O \cup \{u\}$ . This is because if  $v \in g(u)$  and  $v \neq u$ , then  $u \prec v$  by property (g1). Furthermore, by the definition of  $V_1^{\prec}$ , if  $u \prec v$  then  $v \in V_0^{\prec}$ , and therefore  $v \in O$  by Lemma C.2. Combined with properties (g2) and (g3)–(g5), this shows that  $\text{Odd}(g(u)) \cap \bar{O} = \{u\}$  if  $\lambda(u) \in \{XY, XZ\}$ , and  $\text{Odd}(g(u)) \cap \bar{O} = \emptyset$  if  $\lambda(u) = YZ$ . Therefore  $V_1^{\prec} \subseteq V_1^{\prec, XY} \cup V_1^{\prec, XZ} \cup V_1^{\prec, YZ}$ .

To prove that any vertex satisfying one of the three conditions

1.  $\lambda(u) = XY$  and  $\exists K \subseteq O \cap \bar{I}$  s.t.  $\text{Odd}(K) \cap \bar{O} = \{u\}$
2.  $\lambda(u) = XZ$  and  $\exists K \subseteq O \cap \bar{I}$  s.t.  $\text{Odd}(K \cup \{u\}) \cap \bar{O} = \{u\}$
3.  $\lambda(u) = YZ$  and  $\exists K \subseteq O \cap \bar{I}$  s.t.  $\text{Odd}(K \cup \{u\}) \cap \bar{O} = \emptyset$

must be in  $V_1^{\prec}$ , proceed by contradiction. As in the proof of Ref. [42, Lemma 3], we show that a gflow cannot be maximally delayed if there exists some vertex  $w$  which satisfies one of conditions 1–3 but is not in  $V_1^{\prec}$ .

Suppose  $(g, \prec)$  is a maximally delayed gflow and there exists  $w \in V \setminus V_0^{\prec}$  such that  $w$  satisfies condition  $j$ , where  $j \in \{1, 2, 3\}$ , and let  $K$  be the set identified in that condition. Define<sup>8</sup>

$$g'(u) := \begin{cases} K & \text{if } u = w \text{ and } \lambda(w) = XY \\ K \cup \{w\} & \text{if } u = w \text{ and } \lambda(w) \in \{XZ, YZ\} \\ g(u) & \text{otherwise.} \end{cases} \quad (22)$$

<sup>8</sup>The statement of Ref. [42, Lemma 3] is missing the condition that the elements of  $K$  are not inputs, however this error does not appear in that paper's pseudo-code for finding gflow.

Let  $\prec'$  be the partial order defined by<sup>9</sup>

$$u \prec' v \text{ if } \begin{cases} u \neq w \text{ and } u \prec v \\ u = w \text{ and } v \in K \\ u = w \text{ and } v \in \text{Odd}(g'(w)) \setminus \{w\} \end{cases} \quad (23)$$

Then property (g1) of the gflow definition holds by the properties of  $(g, \prec)$  and the second case of the definition of  $\prec'$ . Property (g2) holds by the properties of  $(g, \prec)$  and the third case of the definition of  $\prec'$ . Properties (g3)–(g5) are satisfied for all  $v \in V \setminus \{w\}$  since  $g$  has not changed for those vertices. For  $w$  itself, only one of properties (g3)–(g5) is relevant, and it is satisfied by condition  $j$ .

Now, assume towards a contradiction that  $w \notin V_1^{\prec}$ , i.e. assume that there exists  $k > 1$  such that  $w \in V_k^{\prec}$ . Our construction of  $\prec'$  in conjunction with condition  $j$  implies that  $w \in V_1^{\prec'}$ . For any  $u \in V$  define the *depth* of  $u$ , written  $d(u)$ , as the index  $k$  such that  $u \in V_k^{\prec}$ , and define  $d'(u)$  as the index  $k$  such that  $u \in V_k^{\prec'}$ . Our assumption implies that  $d'(w) < d(w)$ , and our construction of  $\prec'$  implies that  $d'(u) \leq d(u)$  for all vertices  $u$ . This shows that:

$$V_0^{\prec} = V_0^{\prec'} \quad |V_1^{\prec}| < |V_1^{\prec'}| \quad \left| \bigcup_{i=0}^k V_i^{\prec} \right| \leq \left| \bigcup_{i=0}^k V_i^{\prec'} \right| \quad (24)$$

Therefore  $(g', \prec')$  is more delayed than  $(g, \prec)$ , contradicting the assumption that  $(g, \prec)$  is maximally delayed. Hence any vertex  $w$  satisfying one of conditions 1–3 must be in  $V_1^{\prec}$ , i.e.  $V_1^{\prec,XY} \cup V_1^{\prec,XZ} \cup V_1^{\prec,YZ} \subseteq V_1^{\prec}$ .  $\square$

We combine Lemma C.4 with Lemma C.2 to construct a partial converse to Lemma C.3; if we know that  $(\tilde{g}, \tilde{\prec})$  is maximally delayed on  $(G, I, O \cup V_1^{\prec}, \lambda)$ , (with  $V_1^{\prec}$  defined as in Lemma C.4) we want to show that  $(g, \prec)$  is maximally delayed on  $(G, I, O, \lambda)$ .

**Lemma C.5.** Suppose  $(g, \prec)$  is a gflow of  $(G, I, O, \lambda)$  such that  $V_0^{\prec} = O$  and  $V_1^{\prec} = V_1^{\prec,XY} \cup V_1^{\prec,XZ} \cup V_1^{\prec,YZ}$ . Now if  $(\tilde{g}, \tilde{\prec})$  defined as in Lemma C.3 is a maximally delayed gflow for  $(G, I, O \cup V_1^{\prec}, \lambda)$ , then  $(g, \prec)$  is also maximally delayed.

*Proof.* Recall from Lemma C.3 that  $\tilde{g}$  is the restriction of  $g$  to the domain  $V \setminus (V_0^{\prec} \cup V_1^{\prec})$  and  $\tilde{\prec} = \prec \setminus (V_1^{\prec} \times V_0^{\prec})$ . Working towards a contradiction, suppose that  $(g', \prec')$  is a gflow of  $(G, I, O, \lambda)$  which is more delayed than  $(g, \prec)$ . By Lemmas C.2 and C.4 we know that  $V_0^{\prec} = V_0^{\prec'}$  and that  $V_1^{\prec} = V_1^{\prec'}$ . Let  $j$  be such that:

$$\left| V_j^{\prec'} \right| < \left| V_j^{\prec} \right| \quad \forall k \quad \left| \bigcup_{i=0}^k V_i^{\prec'} \right| \leq \left| \bigcup_{i=0}^k V_i^{\prec} \right| \quad (25)$$

Such a  $j$  must exist, and be greater than 1, because  $(g', \prec')$  is more delayed than  $(g, \prec)$ . Define  $(\tilde{g}', \tilde{\prec}')$  analogously to  $(\tilde{g}, \tilde{\prec})$ :

$$\tilde{g}' = \text{the restriction of } g' \text{ to } V \setminus (V_0^{\prec'} \cup V_1^{\prec'}) \quad (26)$$

$$\tilde{\prec}' = \prec' \setminus (V_1^{\prec'} \times V_0^{\prec'}) \quad (27)$$

---

<sup>9</sup>The proof of Ref. [42, Lemma 3] is missing the third case of the definition of  $\prec'$ , but it is actually required in that proof as well.

By Lemma C.3  $(\tilde{g}', \tilde{z}')$  is maximally delayed for  $(G, I, O \cup V_1^{\prec}, \lambda)$ , and by our assumption so is  $(\tilde{g}, \tilde{z})$ . Our contradiction will be shown by noting that  $(\tilde{g}', \tilde{z}')$  is more delayed than  $(\tilde{g}, \tilde{z})$ :

$$V_j^{\prec} = V_{j-1}^{\tilde{z}} \quad V_j^{\prec'} = V_{j-1}^{\tilde{z}'} \quad |V_j^{\prec'}| < |V_j^{\prec}| \quad \implies \quad |V_{j-1}^{\tilde{z}'}| < |V_{j-1}^{\tilde{z}}| \quad (28)$$

and

$$\forall k \quad \left| \bigcup_{i=0}^k V_i^{\prec'} \right| \leq \left| \bigcup_{i=0}^k V_i^{\prec} \right| \quad \implies \quad \forall k \quad \left| \bigcup_{i=0}^{k-1} V_i^{\tilde{z}'} \right| \leq \left| \bigcup_{i=0}^{k-1} V_i^{\tilde{z}} \right| \quad (29)$$

This contradicts  $(\tilde{g}, \tilde{z})$  being maximally delayed, and therefore  $(g, \prec)$  must be maximally delayed.  $\square$

We now adapt the main gflow theorem and proof from Ref. [42, Theorem 2] to our multi-planar setting. The idea is to iteratively construct the layers  $V_k^{\prec}$ , finding candidate correction sets via matrix manipulation in  $\mathbb{F}_2$ .

**Theorem C.6** (Generalisation of [42, Theorem 2] to multiple measurement planes). There exists an algorithm that decides whether a given labelled open graph has an (extended) gflow, and that outputs such a gflow if it exists. Moreover, this output gflow is maximally delayed, and the algorithm takes a number of steps that is polynomial in the number of vertices of the labelled open graph.

*Proof.* Let  $(G, I, O, \lambda)$  be a labelled open graph. The algorithm  $\text{GFLOW}(M, I, O, \lambda)$  (Algorithm 1), where  $M$  is the adjacency matrix of  $G$ , finds a maximally delayed gflow and returns true if one exists and returns false otherwise.

- **Correctness of gflow:** At the  $k^{\text{th}}$  recursive call, the set  $C$  constructed by the algorithm in the **for** loop at line 9 corresponds to the layer  $V_k^{\prec}$  of the partition induced by the returned strict partial order (Definition 3.9). The construction of  $K'$  corresponds to finding a correcting set in  $\bigcup_{j \leq k} V_j^{\prec}$  that has the desired properties for its odd neighbourhood (as specified in conditions (g2)–(g5) of the gflow definition, see Lemma C.4.) Furthermore the restriction  $K' \subseteq O'$  ensures condition (g1), thus if the algorithm returns a solution, then the solution satisfies the conditions of a gflow.
- **Maximality of gflow:** By induction on the number of non-output vertices, we prove that if the given labelled open graph has a gflow, then the algorithm outputs a maximally delayed gflow. Assume that the given labelled open graph has a gflow.
  - The base case is when there is no non-output vertex, and here there is no correction needed: The empty flow  $(g, \emptyset)$  (where  $g$  is a function with an empty domain) is a maximally delayed gflow.
  - For the inductive step suppose that there exist some non-output vertices; according to Lemma C.4 the elements of  $V_1^{\prec}$  are those that satisfy one of the tests inside the loop at line 9. Thus, after the loop (line 20),  $C = V_1^{\prec}$ . Since the existence of a gflow is assumed,  $V_1^{\prec}$  cannot be empty (all non-output vertices have to be corrected), thus the algorithm is called recursively on the labelled open graph  $(G, I, O \cup V_1^{\prec}, \lambda)$  which has fewer non-output vertices. Lemma C.3 ensures the existence of a gflow in  $(G, I, O \cup V_1^{\prec}, \lambda)$ , and by induction we know that the recursive call returns a maximally delayed gflow for  $(G, I, O \cup V_1^{\prec}, \lambda)$ . By Lemma C.5 we know that this construction for  $V_0^{\prec}$  and  $V_1^{\prec}$  yields a maximally delayed gflow for  $(G, I, O, \lambda)$ .



- **Complexity:** First form the matrix  $A$  from the adjacency matrix  $M$  by removing those rows that correspond to output vertices, and keeping all columns that correspond to outputs that are not also inputs (i.e. keep candidate vertices for  $K$ ). Notice that finding a solution for  $K$  inside the `for` loop at line 9 consists of solving a system  $Ax = b_j$  in  $\mathbb{F}_2$ , where  $b_j$  is either a column vector with a 1 corresponding to the vertex  $u$  (in the XY and XZ cases), or  $b_j$  is the column vector of all 0s (in the YZ case.) The resulting column vector  $x$  has a 1 in position  $x_v$  if  $v \in K$  and a 0 otherwise.

Set  $n = |V|$ ,  $\ell = |O|$ ,  $\ell' = |O \setminus I|$ , and  $j$  to range over the  $n - \ell$  vertices in  $\bar{O}$ . In order to solve these  $n - \ell$  equations the block matrix

$$A' = \left[ \begin{array}{c|c|c|c} A & b_1 & \dots & b_{n-\ell} \end{array} \right] \quad (30)$$

of dimensions  $(n - \ell) \times (\ell' + n - \ell)$  is transformed into an upper triangular form, for instance using Gaussian elimination, within  $O(n^3)$  operations. Then for each  $b_j$  a back substitution within  $O(n^2)$  operations is used to find  $x_i$ , if it exists, such that  $Ax_j = b_j$  (see Ref. [6]). The back substitutions costs  $O(n^3)$  operations at each call of the function. Since there are at most  $n$  recursive calls, the overall complexity is  $O(n^4)$ .

□

## D Pseudocode for algorithms

---

### Algorithm 1 Extended Generalized Flow

---

```

1: procedure GFLOW( $M, I, O, \lambda$ )
2:   initialise functions  $g$  and  $d$ 
3:   for all  $v \in O$  do
4:      $d(v) \leftarrow 0$   $\triangleright$  Outputs have depth 0
5:   return GFLOWAUX( $M, I, O, \lambda, 1, d, g$ )  $\triangleright$  Start the recursive process of finding  $V_j^x$ 

6: procedure GFLOWAUX( $M, I, O, \lambda, k, d, g$ )
7:    $O' \leftarrow O \setminus I$ 
8:    $C \leftarrow \emptyset$ 
9:   for all  $u \in \bar{O}$  do
10:    if  $\lambda(u) = XY$  then
11:       $K' \leftarrow$  Solution for  $K \subseteq O'$  where  $\text{Odd}(K) \cap \bar{O} = \{u\}$ 
12:    else if  $\lambda(u) = XZ$  then
13:       $K' \leftarrow \{u\} \cup$  (Solution for  $K \subseteq O'$  where  $\text{Odd}(K \cup \{u\}) \cap \bar{O} = \{u\}$ )
14:    else  $\lambda(u) = YZ$ 
15:       $K' \leftarrow \{u\} \cup$  (Solution for  $K \subseteq O'$  where  $\text{Odd}(K \cup \{u\}) \cap \bar{O} = \emptyset$ )
16:    if  $K'$  exists then
17:       $C \leftarrow C \cup \{u\}$ 
18:       $g(u) \leftarrow K'$   $\triangleright$  Assign a correction set for  $u$ 
19:       $d(u) \leftarrow k$   $\triangleright$  Assign a depth value for  $u$ 
20:    if  $C = \emptyset$  then
21:      if  $O = V$  then
22:        return (true,  $g, d$ )  $\triangleright$  Halt, returning maximally delayed  $g$  and depth values  $d$ 
23:      else
24:        return (false,  $\emptyset, \emptyset$ )  $\triangleright$  Halt if no gflow exists
25:    else
26:      return GFLOWAUX( $M, I, O \cup C, \lambda, k + 1, d, g$ )

```

---

---

**Algorithm 2** Circuit Extraction

---

```
1: procedure EXTRACT( $D$ ) ▷ input is MBQC diagram  $D$ 
2:   Init empty circuit  $C$ 
3:    $G, I, O \leftarrow \text{Graph}(D)$  ▷ get the underlying graph of  $D$ 
4:    $F \leftarrow O$  ▷ initialise the frontier
5:    $D, C \leftarrow \text{ProcessFrontier}(D, F, C)$ 
6:   while  $\exists v \in D \setminus F$  do ▷ there are still vertices to be processed
7:      $D, F, C \leftarrow \text{UpdateFrontier}(D, F, C)$ 
8:     for  $v \in F$  do ▷ the only vertices still in  $D$  are in  $F$ 
9:       if  $v$  connected to input via Hadamard then
10:         $C \leftarrow \text{Hadamard}(\text{Qubit}(v))$ 
11:   Perm  $\leftarrow$  Permutation of Qubits of  $F$  to qubits of inputs ▷ step 5 of Section 5.1
12:   for swap( $q_1, q_2$ ) in PermutationAsSwaps(Perm) do
13:      $C \leftarrow \text{swap}(q_1, q_2)$ 
14:   return  $C$ 
15: procedure PROCESSFRONTIER( $D, F, C$ ) ▷ Corresponds to step 1 of Section 5.1
16:   for  $v \in F$  do
17:     if  $v$  has local Cliffords then
18:        $C \leftarrow \text{Cliffords}(\text{Qubit}(v))$ 
19:       Remove Cliffords from  $v$  to output
20:   for edge between  $v$  and  $w$  in  $F$  do
21:      $C \leftarrow \text{CZ}(\text{Qubit}(v), \text{Qubit}(w))$ 
22:     Remove edge between  $v$  and  $w$ 
23:   return  $D, C$ 
24: procedure UPDATEFRONTIER( $D, F, C$ ) ▷ Corresponds to steps 3,4 of Section 5.1
25:    $N \leftarrow \text{Neighbours}(F)$ 
26:    $M \leftarrow \text{Biadjacency}(F, N)$ 
27:    $M' \leftarrow \text{GaussReduce}(M)$ 
28:   Init  $vs$  ▷ initialise empty set  $vs$ 
29:   for row  $r$  in  $M'$  do
30:     if sum( $r$ ) == 1 then ▷ there is a single 1 on row  $r$ 
31:       Set  $v$  to vertex corresponding to non-zero column of  $r$ 
32:       Add  $v$  to  $vs$  ▷  $v$  will be part of the new frontier
33:   if  $vs$  is empty then
34:     while There is YZ vertex connected to  $F$  do
35:        $v \leftarrow$  YZ vertex connected to  $F$ 
36:        $w \leftarrow$  a neighbour of  $v$  in  $F$ 
37:        $D \leftarrow \text{Pivot}(D, v, w)$ 
38:        $D, C \leftarrow \text{ProcessFrontier}(D, F, C)$ 
39:     return  $D, F, C$ 
40:    $M \leftarrow \text{Biadjacency}(F, vs)$  ▷ smaller biadjacency matrix
41:   for ( $r_1, r_2$ )  $\in$  GaussRowOperations( $M$ ) do
42:      $C \leftarrow \text{CNOT}(\text{Qubit}(r_1), \text{Qubit}(r_2))$ 
43:     Update  $D$  based on row operation
44:   for  $v \in vs$  do ▷ all  $v$  now have a unique neighbour in  $F$ 
45:      $w \leftarrow$  Unique neighbour of  $v$  in  $F$ 
46:      $C \leftarrow \text{Hadamard}(\text{Qubit}(w))$ 
47:      $C \leftarrow \text{Phase-gate}(\text{Phase}(v), \text{Qubit}(w))$ 
48:     Remove  $w$  from  $F$ 
49:     Add  $v$  to  $F$ 
50:    $D, C \leftarrow \text{ProcessFrontier}(D, F, C)$ 
51:   return  $D, F, C$ 
```

---