The ZX calculus is a language for surface code lattice surgery

Niel de Beaudrap[1] and Dominic Horsman[2]

[1]Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD

[2]Department of Physics, Durham University, South Road, Durham, DH1 1LE

Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD

2 September, 2019

A leading choice of error correction for scalable quantum computing is the surface code with lattice surgery. The basic lattice surgery operations, the merging and splitting of logical qubits, act non-unitarily on the logical states and are not easily captured by standard circuit notation. This raises the question of how best to design, verify, and optimise protocols that use lattice surgery, in particular in architectures with complex resource management issues. In this paper we demonstrate that the operations of the ZX calculus — a form of quantum diagrammatic reasoning based on bialgebras — match exactly the operations of lattice surgery. Red and green "spider" nodes match rough and smooth merges and splits, and follow the axioms of a dagger special associative Frobenius algebra. Some lattice surgery operations require non-trivial correction operations, which are captured natively in the use of the ZX calculus in the form of ensembles of diagrams. We give a first taste of the power of the calculus as a language for lattice surgery by considering two operations (T gates and producing a CNOT) and show how ZX diagram re-write rules give lattice surgery procedures for these operations that are novel, efficient, and highly configurable.

1 Introduction

With the development of (small-scale, noisy) devices, quantum computing is in the midst of moving from concept to mature technology [31, 26]. As we try to determine how best to make quantum technology scalable, it becomes apparent that new theoretical tools would be useful, both to organise the components of quantum computers, and to reason effectively about the way that these systems are constructed.

A key element of any such large-scale fault-tolerant architecture is quantum error correction [46, 44]. Currently the technique of choice for flexibility and efficient use of resources is the surface code with lattice surgery [5, 30, 23]. The surface code, in this case, encodes a single logical qubit in the state of many entangled qubits in a single planar surface (a quantum memory). Error detection and correction are performed by the repeated measurement of stabilizers across the surface, to track how noise affects the encoded data. Lattice surgery operations are performed by splitting surfaces into two or more (using measurements), and by merging two or more surfaces together (by further stabilizer measurements).

An intriguing feature of lattice surgery operations is that they necessarily involve quantum transformations of memories by CPTP maps which are not unitary transformations. Two-qubit operations in the previously-standard circuit model, such as a CNOT , may be realised deterministically in lattice surgery, using classical processing similar to that involved in standard teleportation. However, the basic split and merge operations themselves do not rest easily with the standard unitary circuit model. The absence of a notation or language specifically for lattice surgery has made

Niel de Beaudrap: niel.debeaudrap@cs.ox.ac.uk

Dominic Horsman: dom.horsman@gmail.com

it hard efficiently to design, verify, and compile novel surgery patterns. It has also lead to the occasional misrepresentation of lattice surgery as a method for merely simulating unitary gates, rather than a set of computational primitives in its own right. Previous methods for representing lattice surgery include as square patches [29], moving squares around in a 'game' style [39], and as 3D space-time figures in CAD software [25]. In all cases, these representations get visually and technically unwieldy very quickly, making design, optimisation, and particularly verification (that a lattice surgery procedure is doing what we think) challenging.

In this paper we link the operations of lattice surgery directly to a pre-existing diagrammatic language for quantum computing: the ZX calculus. Developed over the last ten years, the ZX calculus (sometimes referred to simply as "ZX") is an abstract graphical language for tensor networks that is complete for quantum mechanics [12, 28]. We show in this paper how the lattice surgery primitives of split and merge — including their non-deterministic byproduct operations — are precisely captured by some of the simplest diagrams of this calculus. In order to do this, we also give a more fine-grained analysis of lattice surgery operations and corrections than appears in the pre-existing literature. The ZX calculus is therefore a language for lattice surgery, meaning that results of work on the calculus can be imported directly for use with lattice surgery.

Unlike circuit notation, the ZX calculus is indeed a *calculus*: a formal language, with meaning-preserving rules for how those diagrams may be transformed, without the need to transcribe those operations as exponentially large matrices (even for full pure-state QM, that cannot be efficiently simulated using e.g. stabilizer notation [27]). By such transformations of these diagrams — corresponding to different sets of lattice surgery procedures that implement the same operation — new protocols may thus be discovered. We give examples of re-writing diagrams for CNOT and T-gate operations to demonstrate this technique. In the process, we discover six novel CNOT implementations in lattice surgery, and significantly reduce the operational overhead of the T-gate implementation.

The use of the ZX calculus for surface codes with lattice surgery makes the manipulation of error corrected operations visually intuitive, and capable of verification at large scales. As these operations will likely form the basic operations that a fault-tolerant device will use at the logical level, ZX therefore becomes the natural language and logic for programming large-scale quantum computing technologies.

2   The planar surface code

The planar surface code uses a 2D lattice segment, in which every edge of the lattice (between neighbouring vertices) is associated with a qubit [5, 24, 15, 44]. These physical qubits encode a single logical qubit's worth of information in their (generally highly-entangled) joint state. For a lattice of n qubits there are $n - 1$ *stabilizers* (operators for which the state of the lattice is a positive +1 eigenstate), fixing a subspace which encodes a single degree of freedom (the logical qubit) across the physical qubits.

Figure 1(a) shows a distance-3 planar surface code lattice (the distance of a code is the weight of the smallest non-trivial error which cannot be detected by the code). The black circles correspond to physical qubits; everything else in the diagram is an aid to the eye. A planar code lattice has two types of boundary: rough (here, left and right) and smooth (here, top and bottom). The stabilizers are defined in terms of local relationships forming "plaquettes". In this paper we use the convention that the term "plaquette" refers to any set of qubits, either on the boundary of a face or surrounding a vertex of the lattice. Specifically, the operators $Z \otimes Z \otimes Z \otimes Z$ around each face (such as the blue-shaded set of four qubits), and $X \otimes X \otimes X \otimes X$ around each vertex (such as the brown shaded region). For plaquettes on the surface boundary, the corresponding stabilisers are $Z \otimes Z \otimes Z$ or $X \otimes X \otimes X$. Figure 1(a) has 13 physical qubits, and 12 plaquettes/stabilisers. The codespace is the simultaneous +1 eigenstate of all of the stabilisers: taken as multi-qubit observables, these operators can be measured without changing the encoded "lattice state".

The information of the logical qubit is accessed through *logical operators*, as shown in Figure 1(b). Any horizontal chain of Z operators starting and finishing on (separate) rough boundaries is a logical Z_L operator, and any vertical chain of X operators between smooth boundaries is a logical X_L operator. These operations commute with all of the stabilizers, and so preserve
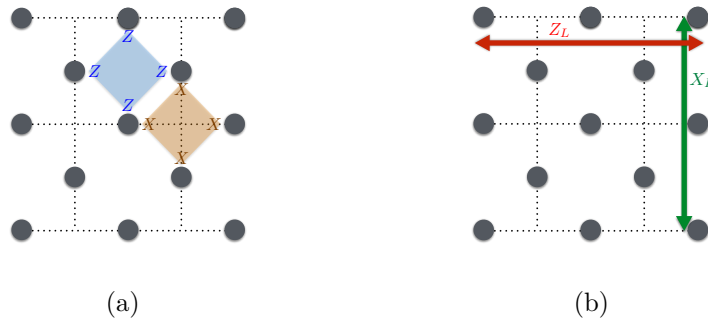
Figure 1: A distance 3 planar surface code. (a) Stabilizers on face and vertex plaquettes. (b) Support of the logical operators: a Z_L operator may be realised by a sequence of Z operators on any horizontal row of qubits connecting the two "rough" boundaries, and similarly an X_L operator my be realised by a sequence of X operators on any vertical column of qubits connecting the "smooth" boundaries.

the code. The significance of the Z_L and X_L operators is two-fold. Firstly, they realise operations on the encoded logical qubit. For instance, a bit-flip of the logical qubit is accomplished via a sequence of physical X operations on qubits forming a chain between smooth boundaries (the X_L operator). The logical operators act secondly as measurable observables of the encoded data itself, encoding specific logical states. For instance, if a chain of qubits between rough boundaries are measured out in the Z basis, then the product of the measurement results ± 1 is the eigenvalue of the logical Z_L. If the result is $+1$ then the logical state was $|0\rangle_L$, and if -1 then it was $|1\rangle_L$.

Regular measurements of the stabilizers allow for error detection and correction by detecting changes to their eigenvalues. A Pauli X or Z error on a physical qubit flips the eigenvalue of the state to -1, for the stabilisers of the opposite type which act on that qubit (Y errors affect both types of stabilisers). A "string" of identical Pauli errors connecting qubits by horizontal and vertical steps will only flip stabilisers at the endpoints of the string. In practise, we avoid explicitly correcting errors (which adds noise owing to imperfections of the correcting gates), and instead keep track of the accumulated errors classically. This information is referred to as a "Pauli frame" [37], and plays the role of a reference frame which indicates how information is stored in the error-affected memory. We refer to the usual reference frame of the surface code, in which the encoded state is a $+1$ eigenstate of every "plaquette operator" (i.e. the state is literally stabilised by those operators) as the *positive (Pauli) frame*. Pauli frames require updating not only owing to error, but also after some lattice surgery operations, described below.

3 Lattice surgery

Lattice surgery is a method of combining or decomposing planar surface-coded memories, to perform multi-qubit operations on the encoded information[1] [30]. There are two types of surgery operation, **split** and **merge**, which can be either "rough" or "smooth". These operations all change the number of memories present, necessarily introducing discontinuities in the Pauli frames of these memories. By accounting appropriately for these changes in the reference frames, we may regard the effect of "split" operations on the encoded state as a single CPTP map from density operators on $\mathcal{H} \cong \mathbb{C}^2$ to operators on $\mathcal{H} \otimes \mathcal{H}$; and "merge" operations as taking a density operator on $\mathcal{H} \otimes \mathcal{H}$ and producing one on $\mathcal{H}$.

---

[1]The other method for planar codes is transversal [15], requiring more operations and/or connectivity. Other procedures for surface codes involve introducing defects into the lattice, and braiding [43, 22] or twisting them [3], or by code deformation [4].
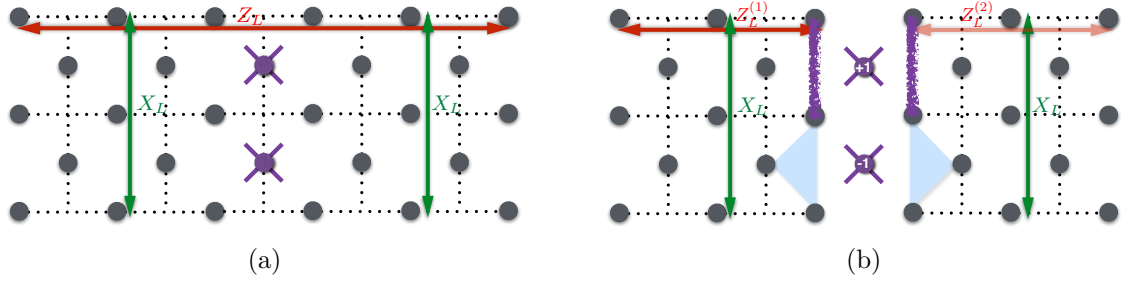
Figure 2: A rough split. (a) Purple qubits are measured out in the Z basis and determine the Pauli frame of the two new surfaces. The split copies X_L, and distributes Z_L across the two resulting surfaces. (b) Correction of a -1 measurement result on the lower qubit: a chain of X operations (purple line) joining the errored syndromes (blue shading) to a boundary.

3.1 Splitting

Figure 2(a) illustrates a **rough split**. For greater clarity, the memory is shown with a greater width than height, though this is not required. We perform Z measurements on the intermediate purple qubits (shown crossed out). The result is two new planar surface memories, each encoding a qubit, with a boundary where the column of measured qubits used to be. Most of the stabilizers in the new "daughter" memories will be the same operators (and have the same eigenvalues) as in the original "mother" memory. However, along the boundary the daughter memories will have modified stabilisers of weight 3 rather than 4, and some of these may end up in the -1 eigenstate, requiring correction to complete the logical operation.

Let $X_L^{(0)}$ represent a logical X operator on the mother memory, and $X_L^{(1)}$ and $X_L^{(2)}$ represent logical X operators on the left and right daughters (similarly for $Z_L^{(0)}$, $Z_L^{(1)}$, and $Z_L^{(2)}$). Note that $X_L^{(1)}$ and $X_L^{(2)}$ both commute with the measurements of the split operation, so the expected value of either $X_L^{(1)}$ or $X_L^{(2)}$ is equal to $X_L^{(0)}$ before the split: $X_L^{(0)} \equiv X_L^{(1)} \equiv X_L^{(2)}$. The $Z_L^{(0)}$ operator, however, is distributed across the two new surfaces. As $Z_L^{(0)}$ commutes with the split procedure and yields Z operators across both memories, it decomposes as $Z_L^{(0)} \equiv Z_L^{(1)} Z_L^{(2)}$. In particular, if subsequently measured, the outcome of measuring $Z_L^{(1)} Z_L^{(2)}$ is the same as measuring $Z_L^{(0)}$ before the split: $\langle Z_L^{(1)} Z_L^{(2)} \rangle = \langle Z_L^{(0)} \rangle$.

If all measurement outcomes of the split are $+1$, then no corrections are applied. However, if one or more measurements give -1, as in Figure 2(b), this produces matching pairs of Z-plaquette stabilizers on the boundaries of each daughter memory that are in the -1 eigenstate (shaded blue). We adapt the Pauli frames (simulating a correction) by a chain of physical X operations linking to the top boundary (purple chain). The lower boundary could also be used; the difference would be equivalent to a logical $X_L^{(i)}$ operation. However, the split has copied $X_L^{(0)}$, so the daughters are in an eigenstate of $X_L^{(1)} \otimes X_L^{(2)}$. As long as the same boundary is chosen for both, the correction strategies differ only by the stabilizer operation $X_L^{(1)} \otimes X_L^{(2)}$ and therefore are equivalent. In this way, we may accommodate the changes in Pauli frame introduced by the measurements, and represent the effect of the split operation on the logical state as a unitary embedding on density operators $\mathsf{S}_R(\rho) = U_R \rho U_R^\dagger$, where $U_R = |{+}{+}\rangle\langle{+}| + |{-}{-}\rangle\langle{-}|$.

A **smooth split** performs the corresponding operation while interchanging Z and X, and the horizontal and vertical axes. Performing X measurements along a row which does not quite reach the two rough boundaries, we obtain two memories where (by adapting the Pauli frames appropriately) the observables on the original and new memories satisfy $Z_L^{(0)} \equiv Z_L^{(1)} \equiv Z_L^{(2)}$ and $X_L^{(0)} \equiv X_L^{(1)} X_L^{(2)}$. The effect on the logical state is then another embedding on density operators $\mathsf{S}_S(\rho) = U_S \rho U_S^\dagger$, where $U_S = |00\rangle\langle 0| + |11\rangle\langle 1|$.

3.2 Merging

A **rough merge** (illustrated in Figure 3) joins two "parent" memories along their rough edges. An intermediate column of qubits (initially here in the $|{+}\rangle$ state, although the $|0\rangle$ state can also be used) is added, and X-plaquette operators are measured across this join. The result is a single
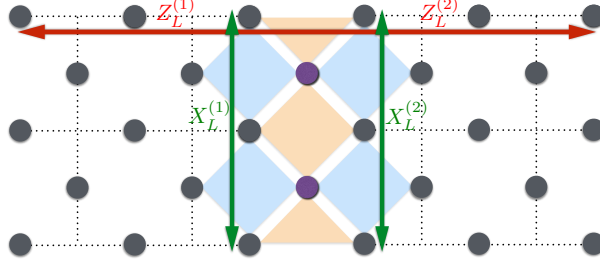
4

Figure 3: A rough merge. Purple qubits are initialised in $|+\rangle$. Measuring X plaquette operators (orange) across the join realises a $X_L^{(1)} \otimes X_L^{(2)}$ measurement.

"child" memory, whose Pauli frame is the union of the frames of the parent surfaces, corrected for the outcomes of the plaquette measurements across the join.

Let us denote the logical X operators on either parent by $X_L^{(1)}$ and $X_L^{(2)}$, and the logical X on the child memory as $X_L^{(3)}$ (similarly for $Z_L^{(i)}$). Measuring the X-plaquette operators across the join and taking the product of the outcomes is equivalent to measuring the two columns of X operators on either side of the divide. These are the $X_L^{(i)}$ logical operators. The action of the rough merge therefore realises a $X_L^{(1)} \otimes X_L^{(2)}$ measurement. This removes a degree of freedom, destroying any information carried by expectation values of the observables $Z_L^{(1)}$ or $Z_L^{(2)}$ alone. However, the new $Z_L^{(3)}$ operator has support on a chain of Z operators the entire width of the surface: $Z_L^{(3)} = Z_L^{(1)} \otimes Z_L^{(2)}$. This commutes with $X_L^{(1)} \otimes X_L^{(2)}$, and so is unchanged by the merge. The new basis states (the positive and negative eigenstates of $Z_L^{(3)}$) arise from the previous ones, $|00\rangle_{12}, |11\rangle_{12} \mapsto |0\rangle_3$ and $|01\rangle_{12}, |10\rangle_{12} \mapsto |1\rangle_3$: this effectively computes the XOR of the labels of the standard basis states (and similarly for superpositions of these basis states).

The action of the rough merge on the $X_L^{(i)}$ operators is more subtle. Consider the case where both parent surfaces are in the positive frame. Measuring $X_L^{(1)} X_L^{(2)}$ by performing the merge tells us whether the parent X_L operators are the same $(+1)$ or different (-1): we call this *the outcome* of the merge. In the "positive branch" $(+1$ outcome), all possible X_L operators on the child surface are identical: $X_L^{(1)} \equiv X_L^{(2)} \equiv X_L^{(3)}$. If, however, the $X_L^{(1)} X_L^{(2)}$ measurement outcome is -1 (the "negative branch"), we have $X_L^{(1)} \equiv -X_L^{(2)}$. It is as if one half of the child memory is subject to a string of Z errors. The choice to correct from the join either to the left or to the right boundary represents a choice of what logic the merge implements. Either $X_L^{(3)} \equiv X_L^{(1)} \equiv -X_L^{(2)}$, or $X_L^{(3)} \equiv X_L^{(2)} \equiv -X_L^{(1)}$, depending on whether we adapt the Pauli frame of the child memory using a chain of Z operations to the right or to the left respectively. These choices differ by a $Z_L^{(3)}$ operation, corresponding to the difference in reference frames.

Described as a CPTP map, the logical transformation of the rough merge is a map from two-qubit density operators to one-qubit density operators $\mathsf{M}_R(\rho) = K_{0,R}\, \rho\, K_{0,R}^\dagger + K_{1,R}\, \rho\, K_{1,R}^\dagger$. Each Kraus operator $K_{b,R}$ is the transformation in the case of a $M = (-1)^b$ measurement outcome of $X_L^{(1)} X_L^{(2)}$. If we represent the classical outcome $M = \pm 1$ itself by a decohered quantum system, the operation is a channel

$$\tilde{\mathsf{M}}_R(\rho) = \left(K_{0,R}\, \rho\, K_{0,R}^\dagger \otimes |{+}1\rangle\langle{+}1|_M \right) + \left(K_{1,R}\, \rho\, K_{1,R}^\dagger \otimes |{-}1\rangle\langle{-}1|_M \right) \tag{1}$$

where the M system heralds the effect on the merged parent memories.

The form of the Kraus operators $K_{0,R}$ and $K_{1,R}$ depends on the choice of reference frame for the child surface. In both cases, we have for the outcome $M = +1$ the Kraus operator $K_{0,R} = |{+}\rangle\langle{++}| + |{-}\rangle\langle{--}|$. Depending on whether one adapts the Pauli frame of the child on the left or the right for the outcome $M = -1$, we respectively obtain the Kraus operator $K_{1,R} = |{+}\rangle\langle{-+}| + |{-}\rangle\langle{+-}|$ or $K_{1,R} = |{+}\rangle\langle{+-}| + |{-}\rangle\langle{-+}|$. This represents the effect of the rough merge for all measurement outcomes, as a CPTP map.

A **smooth merge** performs the corresponding operation, interchanging the horizontal and vertical axes, and also Z and X. An interstitial row of qubits are prepared in the $|0\rangle$ state, and performing Z-plaquette measurements across a horizontal join between two parent memories realises a measurement of $Z_L^{(1)} Z_L^{(2)}$ with outcome $M = \pm 1$. The observables on the original and new memories satisfy $X_L^{(3)} \equiv X_L^{(1)} X_L^{(2)}$, and either $Z_L^{(3)} \equiv Z_L^{(1)} \equiv Z_L^{(2)}$ (if $M = +1$, i.e. the

positive branch) or $Z_L^{(3)} \equiv \pm Z_L^{(1)} \equiv \mp Z_L^{(2)}$ (if $M = -1$, where the signs depend on the choice of Pauli frame for the child memory). The effect on the logical state is a CPTP map $\mathsf{M}_S(\rho) = K_{0,S}\,\rho\,K_{0,S}^\dagger + K_{1,S}\,\rho\,K_{1,S}^\dagger$ where $K_{0,S}$ is realised if $M = +1$ and $K_{1,S}$ if $M = -1$. We have $K_{0,S} = |0\rangle\langle 00| + |1\rangle\langle 11|$ and (depending on how we adapt the Pauli frame of the child memory) one of the two choices $K_{1,S} = |0\rangle\langle 10| + |1\rangle\langle 01|$ or $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$.

The operations of lattice surgery are very different from those usually found in discussions of quantum computing. Describing split and merge operations with standard circuits is unwieldy, as they are explicitly non-unitary; the merge operation in particular is intrinsically non-deterministic. As one may realise a CNOT by a smooth split of the control qubit, followed by a rough merge with the target [30], lattice surgery is usually presented as a way of realising operations in the unitary circuit model. However, a native language of splitting and merging for design, verification, and optimization of lattice surgery protocols would be valuable for the effective management of resources involving these operations [29].

4 The ZX calculus

The ZX calculus is a notation together with a system of transformations, for reasoning about tensors in terms of complementary bases [11, 12]. The ZX calculus can fruitfully be applied to certain processes in quantum information theory: specifically, the standard model of the calculus (using eigenstates of the Pauli X and Z operators) is effective for reasoning about stabilizer-like quantum operations [1], which are useful for describing transformations of Pauli observables. In this section, we provide a brief introduction to the "ZX calculus" suitable for a non-specialist reader who is familiar with the quantum circuit model. (A more complete introduction is available from Refs. [47] and [12].)

4.1 Using ZX diagrams to denote tensors

The notation of the ZX calculus consists of graphs with coloured nodes — which represent operations, including preparations and projections — connected by edges, representing qubits. These graphs denote tensors in the same way that circuit diagrams do, and can often be interpreted as a sequence of linear operators acting on a state-vector. (In the ZX diagrams of this article, the time axis runs left to right.) In its simplest form, the ZX calculus includes nodes of only two colours — conventionally red and green, where 'red' is the darker shade — representing two choices of basis in which information may be stored and transformed. For example, preparations and measurements may be denoted by

$$
\begin{array}{cccccccc}
|0\rangle & R_x(\alpha)\,|0\rangle & |+\rangle & R_z(\alpha)\,|+\rangle & \langle 0| & \langle 0|\,R_x(\alpha) & \langle +| & \langle +|\,R_z(\alpha)
\end{array}
\tag{2}
$$

where each diagram fragment represents the operator written below it.[2] (Setting $\alpha = \pi$ in each case yields the state or projector orthogonal to the corresponding unlabelled node; we omit angles as labels when they are multiples of 2π.) The other basic nodes of the ZX calculus are as follows:

---

[2]More precisely, the diagrams of Eqn. (2) actually represent operators $\sqrt{2}\,|0\rangle$, $\sqrt{2}\langle 0|$, $\sqrt{2}\,|+\rangle$, $\sqrt{2}\langle +|$, and so forth: see *e.g.* Ref. [2]. This distinction is unimportant to our results, and the cumulated scalar $2^{k/2}$ for a diagram can be easily inferred from its topology.

$$|0\rangle\langle 0| + e^{i\alpha}\,|1\rangle\langle 1| \qquad |00\rangle\langle 0| + e^{i\alpha}\,|11\rangle\langle 1| \qquad |0\rangle\langle 00| + e^{i\alpha}\,|1\rangle\langle 11|$$

$$|{+}\rangle\langle {+}| + e^{i\alpha}\,|{-}\rangle\langle {-}| \qquad |{+}{+}\rangle\langle {+}| + e^{i\alpha}\,|{-}{-}\rangle\langle {-}| \qquad |{+}\rangle\langle {+}{+}| + e^{i\alpha}\,|{-}\rangle\langle {-}{-}|$$

$$(3)$$

Note that each of these nodes come in adjoint pairs, as follows (blank nodes denote either red or green, with the same colour throughout each equation):

$$\left[\ \ \right]^{\dagger} = \quad ; \quad \left[\ \ \right]^{\dagger} = \quad ; \quad \left[\ \ \right]^{\dagger} = \quad . \tag{4}$$

We can generalise the nodes above to ones with larger in-degree and out-degree (conventionally known as "spiders" [13]), which for any given angle α corresponds to $|0\rangle^{\otimes k} + e^{i\alpha}\,|1\rangle^{\otimes k}$ for the green nodes, and $|{+}\rangle^{\otimes k} + e^{i\alpha}\,|{-}\rangle^{\otimes k}$ for the red nodes (taking the transpose of those tensor factors representing inputs rather than outputs). Considering nodes merely as tensors, it is not important whether a given wire represents a "bra" or a "ket"; we may be agnostic about their direction, and even allow them to run vertically without ambiguity as to how to evaluate the tensor contraction with an index of another tensor.

Note that the left-hand diagrams of Eqn. (3) are Z and X rotations, respectively. Then nodes of degree 2 suffice to describe arbitrary single-qubit operations by their Euler decomposition. To show how we may represent arbitrary unitary operators with this notation, it suffices to demonstrate a decomposition of a CNOT operator. We may do this up to a scalar factor in the ZX notation as follows:

$$\begin{aligned}
&= \ |0\rangle\langle 0| \otimes \langle 0|{+}\rangle \otimes |{+}\rangle\langle {+}| \ + \ |0\rangle\langle 0| \otimes \langle 0|{-}\rangle \otimes |{-}\rangle\langle {-}| \\
&\quad + \ |1\rangle\langle 1| \otimes \langle 1|{+}\rangle \otimes |{+}\rangle\langle {+}| \ + \ |1\rangle\langle 1| \otimes \langle 1|{-}\rangle \otimes |{-}\rangle\langle {-}| \\[4pt]
&= \ \tfrac{1}{\sqrt{2}}\,|0\rangle\langle 0| \otimes I \ + \ \tfrac{1}{\sqrt{2}}\,|1\rangle\langle 1| \otimes X \ = \ \tfrac{1}{\sqrt{2}}\,\mathrm{CNOT}.
\end{aligned} \tag{5}$$

To represent CNOT exactly using ZX, we would multiply this by a scalar factor of $\sqrt{2}$, *e.g* by including a diagram fragment which is equivalent to the scalar $\sqrt{2}$. Such scalar factors are often omitted as a minor abuse of notation when describing operators using the ZX calculus, but they play an important role for our results, in understanding such diagrams as representing Kraus operators. In similar ways to (5) we may represent any unitary operator with the ZX calculus, using standard results in circuit decomposition.

4.2 Properties of the ZX calculus

As a tensor notation, the ZX calculus enjoys several convenient properties which correspond to abstract representations of equality tests and copying in either in the Z or the X eigenbases. These properties are as follows — where throughout, blank nodes denote either red or green (with the same colour throughout each equation), provided that the sum of the angles in each diagram are equivalent mod 2π:

$$(6a)$$

$$(6b)$$

$$
\text{(6c)}
$$

$$
\text{(6d)}
$$

These properties allow us to reduce nodes of higher in- and out-degree in arbitrary ways, using the nodes of Eqns. (2) and (3), so long as the composite diagram has the correct in- and out-degree and the same total angle [12, Ch. 8.6.1]. For instance, using Eqns. (6c) and (6d), we may without confusion define nodes of degree 4 from compositions of nodes of degree 3 (where again we require that mod 2π sum of the phases remains constant through each equation):

$$
\text{(7a)}
$$

$$
\text{(7b)}
$$

$$
\text{(7c)}
$$

We may recursively define nodes of any degree in this way. This is the so-called *Spider Law*, and also allows us to simplify diagrams by merging nodes of degree 1 and 2 of the same colour by adding their angles [12, Ch. 9.4]. Eqn. (6a) also allows for simplification by reducing the number of nodes: for instance, if the sum of the angles is a multiple of 2π, the resulting operation simply represents the identity operator $\mathbb{1}$, in which case we may remove the node without changing the meaning of the diagram.

Finally, for angles $\alpha = k\pi$ for integer k, we may describe a simple rule for commuting nodes of degree 2, past differently coloured nodes of degree 3:

$$
\text{(8)}
$$

The reflection of Eqn. (8) about the vertical axis also holds, and represents the sense in which the nodes of one colour correspond to simple arithmetic operations on information stored in the "distinguished basis" of the other colour. These rules in effect provide a reduced instruction set for transformation and evaluation of tensors, which is well-suited for automated reasoning about equivalence of quantum procedures [33].

4.3   A simple demonstration of how to use the ZX calculus

We now provide simple illustrations of the way in which the ZX calculus can be used as a tool for computation, by demonstrating an elementary fact about the stabiliser formalism. Our aim here is not to argue that the ZX calculus is in some way a superior tool to prove this result, but instead to *demonstrate* how the ZX calculus produces this result — and in so doing, demonstrate the transformations which we use in our results. (For a more complete introduction to ZX, the interested reader is invited to refer to Refs. [47] or [12].)

Our demonstration is to show how to derive how to commute Pauli operators past CNOT gates, using the "Spider Rule" (the higher-level result following from Eqns. (6) allowing one to merge any two connected nodes of the same colour) together with Eqn. (8). The Spider Rule describes how we can merge, or split, two nodes of the same colour while keeping the total phase the same (mod 2π). For example, with a node of degree 3 and a node of degree 2, where the node of degree

3 happens not to have a phase of zero, we have:

$$(9)$$

where the white nodes again stand for either red or green nodes (the same colour throughout). This effectively allows us to "commute" a degree 2 node past a degree 3 node of the same colour. Furthermore Eqn. (8) essentially describes how X and Z operators propagate between controls and targets of CNOT gates, as degree-2 nodes with a π phase are either X or Z operators:

$$|0\rangle\langle0| + e^{i\pi}|1\rangle\langle1| = Z \qquad\qquad |+\rangle\langle+| + e^{i\pi}|-\rangle\langle-| = X \tag{10}$$

We may then verify, (*i.e.*, compute from first principles) how to commute Z and X operators past a CNOT gate:

$$(11)$$

The equations (expressed as rewritings of diagrams) in the left-hand column represent how Z operators commute with the control of a CNOT , and propagate from the target to the control CNOT ; the equations of the right-hand column represent how X operators commute with the target of a CNOT , and propagate from the control to the target of a CNOT . Thus, in the ZX calculus, the propagation of Pauli operators is a consequence of the rules for how the two colours of nodes interact, and can be demonstrated without matrix calculations.

5   Lattice surgery in the ZX calculus

Having in hand an account of lattice surgery using encoded Kraus operators, and given the presentation above of the tensors involved in the standard model of the ZX calculus, it should be clear that there are considerable similarities between them. A lattice "split" operation copies information represented in one of two bases, as shown in Section 3.1. This mirrors the action of Eqn. (8): for instance, a red 1-to-2 node copies a green π-rotation, representing both how a change in the input maps to an output, and also how an input Z observable is equivalent to a product of Z observables at the output. It has also been previously observed that the red node in the CNOT of Eqn. (5) acts as an XOR [12, Ch. 5.3.5], which is also the action of a rough merge. We thus appear to have an equivalence between red degree-3 nodes in the ZX calculus with rough lattice surgery operations, and between green degree-3 nodes and smooth operations.

  We now make this equivalence precise and explicit, demonstrating that the actions of lattice surgery on encoded data essentially form a model of the ZX calculus. The precise nature of the equivalence requires us to describe the merge operations in terms of *ensembles* of simple diagrams, which yield byproduct operations associated with the heralded outcomes of the merge process. The ZX calculus then provides us with a way to simplify the descriptions of these byproduct operations,

allowing us to account for them in much more complicated procedures than the standard realisation of the CNOT gate.

In what follows, a *lattice surgery procedure* is a composition of the maps M_S and M_R (albeit possibly using different conventions for updating the Pauli frame in each instance) with S_S and S_R, and preparations of fresh qubits prepared in states $|g_\alpha\rangle \propto |0\rangle + e^{i\alpha}|1\rangle$ and $|r_\alpha\rangle \propto |+\rangle + e^{i\alpha}|-\rangle$.

Lemma 1. *The effect on the logical state space of a surface code of a smooth split, and of the positive branch of a smooth merge, are given by the operators associated with the green degree-3 nodes in Eqn. (3) for $\alpha = 0$; and similarly for the rough split/merge operations and the red degree-3 nodes.*

Proof. For the split operations, this follows from the equality of the operators associated to the 1-to-2 nodes in Eqn. (3) for $\alpha = 0$, and the corresponding Kraus operators U_S and U_R (respectively) of the unitary embeddings S_S and S_R of Section 2. Similarly, the Kraus operators $K_{0,S}$ and $K_{0,R}$ (respectively) of the merge operators M_S and M_R of Section 3 are equal to the operators associates to (respectively) the green and the red 2-to-1 nodes in Eqn. (3). □

The preceding observation about identical pairs of linear operators, has the following consequence on how we can reason about compositions of operations in lattice surgery:

Corollary 2. *The positive branches of lattice surgery procedures (i.e. conditioned on +1 outcomes of all merge operations) provide a model for the equational theory of the ZX calculus without projections, in which nodes of degree 2 or 3 have an angle of 0.*

In particular: if we let $\multimap\!\!\!<$ denote the action of the unitary embedding U_S (respectively, U_R) on encoded data realised by a smooth (resp. a rough) split, $\!>\!\!\!\multimap$ denote the action of the Kraus operator $K_{0,S}$ (resp. $K_{0,R}$) on encoded data realised in the positive branch by a smooth (resp. a rough) merge, $\overset{\alpha}{\multimap\!\!\circ\!\!\multimap}$ denote an $R_x(\alpha)$ gate (resp. a $R_z(\alpha)$ gate), and $\overset{\alpha}{\circ\!\!\multimap}$ denote (up to an extra factor of $\sqrt{2}$) preparation of a $|g_\alpha\rangle$ state (resp. a $|r_\alpha\rangle$ state), then these operations satisfy all of the properties of Eqns. (4) and (6). More complicated green or red nodes (including nodes of degree > 1 with non-zero angles) can then be realised using the Spider Law [12, Ch. 9.4], describing them as the effect of compositions of split and merge operations in the positive branch.

The qualification "in the positive branch" in Corollary 2 is significant, and relates to the normalisation of the diagram in Eqn. (5). Omitting the correction operations needed to realise CNOT deterministically, the standard lattice-surgical realisation of CNOT [30] is by the composition $(\mathbf{1}\otimes\mathsf{M}_R)(\mathsf{S}_S\otimes\mathbf{1})$. We may associate to this composition a pair of Kraus operators $(I\otimes K_{0,R})(U_S\otimes I)$ and $(I\otimes K_{1,R})(U_S\otimes I)$, where the positive branch corresponds to the first of these. A simple calculation reveals that $(I\otimes K_{0,R})(U_S\otimes I) = \frac{1}{\sqrt{2}}\,\text{CNOT}$, which reflects the fact that the positive branch occurs with probability $\frac{1}{2}$ and realises a CNOT operation on all input states. From this standpoint, the subnormalisation of Eqn. (5) is a feature of ZX notation, not a bug: it captures not only the way in which states transform in the positive branch of a lattice-surgery procedure, but also the 2-norms of the Kraus operators which govern the transformation.

This motivates a view of simple ZX diagrams as denoting Kraus maps of lattice surgery procedures. It remains to describe how to represent the "negative branch" of rough and smooth merges.

Lemma 3. *For a rough merge, if we adapt the Pauli frame of the child memory to agree with the second parent memory so that $K_{1,R} = |+\rangle\langle-+| + |-\rangle\langle+-|$ — respectively: if we adapt the Pauli frame of the child memory to agree with the first parent memory so that $K_{1,R} = |+\rangle\langle+-| + |-\rangle\langle-+|$ — then*

$$K_{1,R} \;\; = \;\; \overset{\pi}{\underset{\text{\textcolor{red}{●}}}{\circ}}\!\!\!\multimap \qquad \left(\text{respectively,} \quad K_{1,R} \;\; = \;\; \underset{\pi}{\overset{\text{\textcolor{red}{●}}}{\circ}}\!\!\!\multimap\right). \tag{12a}$$

Similarly, for a smooth merge, if we adapt the Pauli frame of the child memory to agree with the second parent memory so that $K_{1,S} = |0\rangle\langle10| + |1\rangle\langle01|$ — respectively: if we adapt the Pauli fram

of the child memory to agree with the first parent memory so that $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$ *— then*

$$K_{1,S} \;=\; \text{<image>} \qquad \left(\text{respectively,} \quad K_{1,S} \;=\; \text{<image>} \right). \tag{12b}$$

Proof. By calculation of the operators, from Eqn. (3). □

Corollary 4. *A rough merge operation realises a logical transformation* M_R *with Kraus operators*

$$\left\{ \text{<image>} \;,\; \text{<image>} \right\} \quad or \quad \left\{ \text{<image>} \;,\; \text{<image>} \right\} \tag{13a}$$

acting on the logical qubit of the surface codes, depending on whether one adopts the convention of correcting either the first or the second parent in the negative branch. Similarly, a smooth merge operation realises a logical transformation M_S *with Kraus operators*

$$\left\{ \text{<image>} \;,\; \text{<image>} \right\} \quad or \quad \left\{ \text{<image>} \;,\; \text{<image>} \right\} \tag{13b}$$

acting on the logical qubit of the surface codes, depending on whether on adopts the convention of correcting either the first or the second parent in the negative branch.

Proof. This follows from the previous Lemmata concerning the equality of the Kraus operators. □

Remark. In practise, it is likely to prove convenient to analyse the operators M_S and M_R with *annotated ZX diagrams* [18], in which we allow nodes whose phases are not necessarily constants, and may include variables whose values may only be determined during the computation (*e.g.* as with the results of a measurement). For instance, we may denote the pairs of logical Kraus operators for M_R by

$$\text{<image>} \qquad or \qquad \text{<image>} \tag{14a}$$

depending on the conventional choice of how one adapts the Pauli frame of the child memory; here $b \in \{0, 1\}$ indicates the measurement outcome $M = (-1)^b$ of that merge, and the Kraus operator $K_{b,R}$ which is realised as a transformation of the quantum state. Similarly, the pairs of logical Kraus operators of M_S may be denoted by

$$\text{<image>} \qquad or \qquad \text{<image>} \tag{14b}$$

depending on the conventional choice of how one adapts the Pauli frame of the child memory. In either case, the bit $b \in \{0, 1\}$ is revealed when the operation is performed, as a result of the stabiliser measurements which are performed during the merge operations; one may reason about the operations performed in general by leaving b as a formal indeterminate, and describing the phase by a polynomial with real coefficients over some set of indeterminates (associated operationally with the outcomes of operations).

The purpose of associating the operations of lattice surgery with simple (annotated) ZX diagrams is two-fold: (a) to allow us to reason about more diverse compositions of lattice surgery

operations using compositions of the diagrams, and (b) to allow us to relate more complex ZX-diagrams to lattice surgery procedures which may be used to realise them. For (b), it suffices to produce a simple ZX diagram which is equivalent, and then consider how this diagram may be realised using (a). We have already glimpsed how (a) might be done in Corollary 2 for the positive branch of any lattice surgery procedure; the following extends this to arbitrary outcomes of the merges:

Theorem 5. *Lattice surgery procedures model randomly-constructed simple ZX diagrams, in which gadgets realising 2-to-1 nodes are selected from either one of the pairs of operations of Eqn. (14a) for rough merges, and from either one of the pairs of operations of Eqn. (14b) for smooth merges; and the probability of each diagram is given by its 2-norm.*

Proof. Each pair of diagrams from Eqns. (13) describes Kraus operators of M_S and M_R (as determined by a conventional choice of procedure to update the Pauli frame). A single gadget selected from one of these pairs is modelled by a lattice merge of the appropriate sort, for which the diagram represents a Kraus operator. Composing such maps (and others) to produce a simple ZX diagram represents the effect of composing the Kraus operators for the lattice merge operations with those of split operations, and preparation of qubits in the $|g_\alpha\rangle$ and $|r_\alpha\rangle$ states. These form Kraus operators for the entire lattice surgery procedure. The 2-norm of the diagram then corresponds to the probability of the Kraus operator (i.e. the particular merge outcomes) being realised. □

6 Consequences of the equivalence

We have described a tight connection between the ZX calculus and lattice surgery. Splitting and merging can be written in terms of degree-3 nodes (including correction operations if necessary); realised as Kraus operators of lattice surgery procedures. The use of ZX as a tool for manipulating lattice surgery diagrams has a huge number of potential applications. To begin with, almost all work on the ZX calculus can now be applied directly to lattice surgery. In particular, known equivalences under re-writing in the calculus now have an interpretation in terms of equivalence of lattice surgery patterns. For example, the Frobenius laws of Eqn. (6) can now be imported and read directly as referring to equivalent set of split and merge operations. The standard re-write axioms of the calculus give further pattern equivalences. To give a flavour of the power of this re-writing system, we here detail two examples where ZX diagrammatic equivalence gives novel lattice surgery procedures: for non-Clifford rotations, and the CNOT (as a subroutine).

6.1 Lattice surgery T gate

Magic states permit the realisation of logical operations outside of the Clifford group on surface codes, allowing for (approximately) universal quantum computing [6]. In the surface code, they may be distilled to high fidelity, injected into a code surface, and then used to perform teleported rotation gates [22], as these cannot be performed transversally [20]. A T gate uses a magic state and teleportation to perform a $\pi/4$ rotation. The usual presentation of the teleported T gate involves a CNOT and measurement (see e.g. [8]). However, the ZX calculus allows us to give a more efficient procedure for lattice surgery.

Consider the state $|A\rangle \propto |0\rangle + e^{i\pi/4}|1\rangle$. In a teleported T gate this is used to perform a Z-rotation $R_z(\pi/4) = |0\rangle\langle 0| + e^{i\pi/4}|1\rangle\langle 1|$. In the ZX calculus the $|A\rangle$ state is represented by a green $\pi/4$ preparation node, and the $R_z(\pi/4)$ operation by a green $\pi/4$ degree-2 node. As an application of Eqn. (6), we have

$$\begin{array}{ccc} & & \\ \overset{\pi/4}{\bullet} & = & \overset{\pi/4}{-\bullet-} \\ \diagdown\!\!-\bullet\!\!- & & \end{array} \tag{15}$$

By Corollary 2, the left-hand diagram represents the positive branch of a smooth merge of an $|A\rangle$ state with an input state. Using Lemma 3 and Eqn. (6), the negative branch of this procedure is

described by

 (16)

As with a standard T gate [22, §6], the rotation to correct this to $+\pi/4$ uses $|Y\rangle \propto |0\rangle + \mathrm{e}^{i\pi/2}|1\rangle$:

 (17)

The negative branch of this operation is:

 (18)

This can now be corrected, if necessary, by a phase flip. The T gate has now become simple merging with magic states.[3] Note that the above ZX analysis also holds if we interchange red and green nodes: we thus also obtain a procedure for $R_x(\pi/4)$ rotations as well. Procedures for smaller-angle rotations can similarly be found, which can then be used for the most efficient magic state compilations [7].

6.2 Realising CNOT gates

The original lattice surgery procedure for the CNOT gate is a smooth split of the control, followed by a rough merge of one of the daughter memories with the target [30, §4.1], followed by appropriate corrections depending on the outcome of the measurements in the merge procedure. Compare the positive branch of this procedure (using Corollary 2) to the (subnormalised) representation of CNOT from Eqn. (5):

 (19)

These two presentations are clearly topologically equivalent. By considering other topologically equivalent presentations of the coloured graph of Eqn. (5), and how those graphs describe operations which may be decomposed as ZX diagrams, we may obtain a further set of procedures, all of which implement the CNOT in the positive branch (in the Appendix we show how we may realise CNOT operations in the negative branch for each of these procedures, as well as for that given in Eqn. (19)):

 (20)

From left to right: the second of these diagrams describes a rough split of the target followed by a smooth merge with the control; the third and fourth show the creation of an intermediary Bell pair $|\Phi^+\rangle \propto |00\rangle + |11\rangle = |{+}{+}\rangle + |{-}{-}\rangle$, the two halves of which are merged with the control and target respectively; and the final two describe splits to both control and target qubits (smooth and rough respectively), with a simulated Bell projection $\langle\Phi^+| \propto \langle 00| + \langle 11| = \langle{+}{+}| + \langle{-}{-}|$ on a pair of the daughter memories.

Thus, by minor variations of ZX diagrams, we obtain a proliferation of different procedures to realise a CNOT by lattice surgery procedures — of which only the one illustrated in Eqn. (19) has previously appeared in the literature. This gives an indication of the breadth of compilation flexibility that the use of the ZX calculus will bring to the surface code with lattice surgery.

---

[3]The alternative convention for the negative branch of merges works equally well (see Section B. of the Appendix).

7   Conclusions

We have demonstrated how the ZX calculus acts as a precise and fundamental description of the splitting and merging operations of lattice surgery on the planar surface code. Merge operations output a bit of information, and this is used to determine which out of a set of possible diagrams describe the post-merge state. Sometimes, as with the case of the constructed CNOT gate, these possible diagrams differ by logical Pauli corrections. However, this is not in general the case. Previously there was no easy, systematic way to describe these Pauli frame updates in the high-level language used (*i.e.*, the circuit model). With the use of the ZX calculus, there is now a high-level notation, together with rules to calculate with that notation. This allows one to straightforwardly describe the Pauli frame information associated with a lattice surgery procedure, as well as how that information propagates under further operations.

Lattice surgery fits precisely the structure of the ZX calculus in the "positive branch", *i.e.*, when merge measurement outcomes are $+1$. Merge and split operations — whether rough (corresponding to red nodes) or smooth (green) — are related through the dagger structure of the calculus. A rough merge is a retraction (inversion by post-composition) of a rough split, and equivalently for smooth merges and splits. Legs of nodes, which represent different parent or child surfaces, may be interchanged; this is the commutative structure. The specialness axioms also follow, as does associativity. Because of this, multi-input/output "spiders" may be defined. In lattice surgery terms, this is a simultaneous splitting or merging into/from multiple surfaces. Multi-surface splitting was described in the original paper on lattice surgery [30]; now, however, we can prove what effect such operations have on the encoded data, and how they interact with each other. In particular, the ability to describe the byproducts in the "negative branches", and when they can be corrected, will make it possible to determine which multi-node operations can be deterministically performed.

We have focussed in this paper on lattice surgery within the *surface* code. The question of a similar representation for other codes is an interesting question. In colour code lattice surgery, for example, Y-basis merging appears [38] (see also [39]): this may motivate the development of a calculus closely related to the ZX calculus in which such merges have a similarly direct representation. In other ways, ZX already has proven its worth as a language for the broad class of codes known as coherent parity check codes (which includes the CSS construction) [10]; and an early use of automated theorem proving in the calculus was verify correctness of the Steane code [17]. Ways in which ZX may be used as a notation to describe the use of stabilizer codes is an important area of ongoing further research.

There are a number of uses for the work presented here; the following is by no means exhaustive. Firstly, an immediate application is the use of ZX as the basis of an *intermediate representation* in compilers for near-term error corrected quantum devices. Such devices, based for example on architectures such as [41, 40], have strictly limited resources. Compiling protocols to CNOT operations wastes valuable operations; using ZX we can now compile to precisely the operations that such a device will in practice implement. The use of ZX, and automated tools [36], for compilation is an ongoing larger research project, including optimisations for Clifford circuits [21], circuit simplification [19] including routing and topological constraints [34, 14], and state-of-the-art T-count reductions at the time of writing [35]. Secondly, the ZX calculus can be used as a quantum protocol design tool. Through lattice surgery, the ZX calculus expresses another circuit-like model of quantum computing which we may attempt to realise through practical operations. The ZX calculus allows us to explore new techniques for quantum protocols, which may not be as naturally expressed by unitary circuits (such as in measurement-based quantum computing [16]), for computation and also networking/communication [32, 45] and large-scale qubit registers [9] There is a large amount of work on the calculus that can now be imported for use with lattice surgery; the new procedures given in this paper will be the first of many examples. With the equivalence between ZX and lattice surgery fixed, it will now be possible to take a diagram in the ZX calculus, re-write it, and then interpret that diagram as a lattice surgery procedure — which will include a definition of determinism in diagrams. The equivalence shown in this paper allows us to analyse lattice surgery procedures, to determine how a ZX transformation may practically be realised. This will include the question of time-ordering (including simultaneity) of compiled operations.

There are further uses of the ZX/lattice surgery equivalence. By producing ZX-compiled lat-

tice surgery procedures for known algorithms and protocols, not only can we potentially simplify procedures significantly, but also a different logic can be applied to analyse how these protocols behave. ZX from its beginnings was introduced as another logic to analyse and design further quantum algorithms and protocols. By expanding the space of design tools, we expand what can be thought of in quantum computing. With a physical model of the ZX calculus in the operations of lattice surgery, this programme has an important role to play in realising near-term error corrected quantum devices.

Acknowledgements

References

[1] M. Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014. DOI: 10.1088/1367-2630/16/9/093021.

[2] 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.

[3] H. Bombin. Topological order with a twist: Ising anyons from an Abelian model. *Physical review letters*, 105(3):030403, 2010. DOI: 10.1103/PhysRevLett.105.030403.

[4] H. Bombin and M. Martin-Delgado. Quantum measurements and gates by code deformation. *Journal of Physics A: Mathematical and Theoretical*, 42(9):095302, 2009. DOI: 10.1088/1751-8113/42/9/095302.

[5] S. Bravyi and A. Kitaev. Quantum codes on a lattice with boundary. *Preprint, arXiv:quant-ph/9811052*, 1998. Translation of Quantum Computers and Computing 2 (1), pp. 43-48. (2001).

[6] S. Bravyi and A. Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005. DOI: 10.1103/PhysRevA.71.022316.

[7] E. Campbell and J. O'Gorman. An efficient magic state approach to small angle rotations. *Quantum Science and Technology*, 1(1):015007, 2016. DOI: 10.1088/2058-9565/1/1/015007.

[8] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172, 2017. DOI: 10.1038/nature23460. arXiv:1612.07330.

[9] T. Carette, D. Horsman, and S. Perdrix. SZX-calculus: Scalable graphical quantum reasoning. *Preprint, arXiv:1905.00041*, 2019.

[10] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical structures for design and verification of quantum error correction. *arXiv:1611.08012*, 2018.

[11] B. Coecke and R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011. DOI: 10.1088/1367-2630/13/4/043016.

[12] B. Coecke and A. Kissinger. *Picturing Quantum Processes: A first course in quantum theory and diagrammatic reasoning.* Cambridge University Press, 2017. DOI: 10.1017/9781316219317.

[13] B Coecke, E. Paquette, and D. Pavlovic. Classical and quantum structuralism. *Semantic Techniques in Quantum Computation, eds. Gay S., Mackie I., Cambridge University Press*, 2008. DOI: 10.1017/CBO9781139193313.003. arXiv:0904.1997.

[14] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the Qubit Routing Problem. In Wim van Dam and Laura Mancinska, editors, *14th Conference on the Theory of Quantum Computation, Communication*

and Cryptography (TQC 2019), volume 135 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:32, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-112-2. DOI: 10.4230/LIPIcs.TQC.2019.5. URL http://drops.dagstuhl.de/opus/volltexte/2019/10397. arXiv:1902.08091.

[15] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002. DOI: 10.1063/1.1499754. arXiv:quant-ph/0110143.

[16] R. Duncan and S. Perdrix. Graph states and the necessity of Euler decomposition. In *Conference on Computability in Europe*, pages 167–177. Springer, 2009. DOI: 10.1007/978-3-642-03073-4.

[17] Ross Duncan and Maxime Lucas. Verifying the steane code with Quantomatic. *Proceedings QPL 2013*, pages 33–49, 2013. DOI: 10.4204/EPTCS.171.4. arXiv:1306.4532.

[18] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 285–296, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14162-1.

[19] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Preprint, arXiv:1902.03178*, 2019.

[20] B. Eastin and E. Knill. Restrictions on transversal encoded quantum gate sets. *Phys. Rev. Lett.*, 102:110502, Mar 2009. DOI: 10.1103/PhysRevLett.102.110502. arXiv:0811.4262.

[21] Andrew Fagan and Ross Duncan. Optimising Clifford circuits with Quantomatic. In *Proceedings of the 15th International Conference on Quantum Physics and Logic (QPL)*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 85–105. Open Publishing Association, 2019. DOI: 10.4204/EPTCS.287.5.

[22] A. Fowler, A. Stephens, and P. Groszkowski. High-threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80:052312, 2009. DOI: 10.1103/PhysRevA.80.052312.

[23] Austin G Fowler and Craig Gidney. Low overhead quantum computation using lattice surgery. *Preprint, arXiv:1808.06709*, 2018.

[24] M. Freedman and D. Meyer. Projective plane and planar quantum codes. *Foundations of Computational Mathematics*, 1(3):325–332, 2001. DOI: 10.1007/s102080010013.

[25] Craig Gidney and Austin G Fowler. Efficient magic state factories with a catalyzed |CCZ⟩ to 2|T⟩ transformation. *Quantum*, 3, 2019. DOI: 10.22331/q-2019-04-30-135. arXiv:1812.01238.

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

[27] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum Hamming bound. *Physical Review A*, 54(3):1862, 1996. DOI: 10.1103/PhysRevA.54.1862. arXiv:quant-ph/9604038.

[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.

[29] D. Herr, F. Nori, and S. Devitt. Lattice surgery translation for quantum computation. *New Journal of Physics*, (19):013034, 2017. DOI: 10.1088/1367-2630/aa5709.

[30] C. Horsman, A. G Fowler, S. Devitt, and R. Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012. DOI: 10.1088/1367-2630/14/12/123011. arXiv:1111.4022.

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

[32] C. Jones, D. Kim, M. Rakher, P. Kwiat, and T. Ladd. Design and analysis of communication protocols for quantum repeater networks. *New Journal of Physics*, 18(8):083015, 2016. DOI: 10.1088/1367-2630/18/8/083015.

[33] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *International Conference on Automated Deduction*, pages 326–336. Springer, 2015. DOI: 10.1007/978-3-319-21401-6˙22.

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

[35] Aleks Kissinger and John van de Wetering. Reducing T-count with the ZX-calculus. *Preprint, arXiv:1903.10477*, 2019.

[36] Aleks Kissinger and John van de Wetering. PyZX: Large scale automated diagrammatic reasoning. *Preprint, arXiv:1904.04735*, 2019.

[37] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005. DOI: 10.1038/nature03350. arXiv:quant-ph/0410199.

[38] Andrew J Landahl and Ciáran Ryan-Anderson. Quantum computing by color-code lattice surgery. *Preprint, arXiv:1407.5103*, 2014. SAND2014-15911J.

[39] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, 2019. DOI: 10.22331/q-2019-03-05-128.

[40] N. Nickerson, Y. Li, and S. Benjamin. Topological quantum computing with a very noisy network and local error rates approaching one percent. *Nature communications*, 4:1756, 2013. DOI: 10.1038/ncomms2773.

[41] N. Nickerson, J. Fitzsimons, and S. Benjamin. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. *Physical Review X*, 4(4): 041041, 2014. DOI: 10.1103/PhysRevX.4.041041.

[42] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information.* Cambridge University Press, Cambridge UK, 2000. DOI: 10.1017/CBO9780511976667.

[43] R. Raussendorf and J. Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Physical review letters*, 98(19):190504, 2007. DOI: 10.1103/PhysRevLett.98.190504.

[44] B. Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87 (2):307, 2015. DOI: 10.1103/RevModPhys.87.307.

[45] R. Van Meter. *Quantum Networking.* John Wiley & Sons, 2014. DOI: 10.1002/9781118648919.

[46] R. Van Meter and C. Horsman. A blueprint for building a quantum computer. *Communications of the ACM*, 56(10):84–93, 2013. DOI: 10.1145/2494568.

[47] The ZX-calculus. http://zxcalculus.com/. Accessed 08/07/2019.

Appendix

In this Appendix we show how the correction operations for the CNOT procedures of Eqns. (19) and (20) equate to logical Pauli corrections on the control and/or target qubit, and also show how the choice of convention for adapting merge operations in T-state injection does not introduce any essential difficulties.

A. CNOT patterns

Eqns. (19) and (20) describe transformations of logical qubits which are realised in the "positive branch" of lattice surgery procedures involving various merges, splits, preparations, and projections, which may be inferred from the topology of the diagram involved. Below, we describe other possible evolutions which those procedures may realise, for different conventional choices of the Kraus operations $K_{1,S}$ or $K_{1,R}$ for the "negative" branch of the merge operations involved.

In each case, the transformations realised differ from the transformations realised in the positive branch only by logical Pauli operations (represented by green and/or red π-nodes). These logical Pauli operators are heralded by the outcome of the merge operations, and can therefore be tracked or corrected. This demonstrates that these protocols are not sensitive to the convention used for adapting the Pauli frame of the child memory in any given merge operation.

A.1. Smooth split control, rough merge with target.

Positive branch: (21a)

Negative branch, using
$K_{1,R} = |{+}\rangle\langle{-}{+}| + |{-}\rangle\langle{+}{-}|$:

$$\qquad\qquad\qquad\qquad\qquad = \qquad\qquad\qquad\qquad\qquad\qquad (21\text{b})$$

Negative branch, using
$K_{1,R} = |{+}\rangle\langle{+}{-}| + |{-}\rangle\langle{-}{+}|$:

$$\qquad\qquad\qquad = \qquad\qquad\qquad = \qquad\qquad\qquad (21\text{c})$$

A.2. Rough split target, smooth merge with control.

Positive branch:

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (22\text{a})$$

Negative branch, using
$K_{1,S} = |0\rangle\langle10| + |1\rangle\langle01|$:

$$\qquad\qquad\qquad = \qquad\qquad\qquad = \qquad\qquad\qquad (22\text{b})$$

Negative branch, using
$K_{1,S} = |0\rangle\langle01| + |1\rangle\langle10|$:

$$\qquad\qquad\qquad\qquad\qquad = \qquad\qquad\qquad\qquad (22\text{c})$$

A.3. Create intermediary Bell pair then merge with control and target.

By preparing a single-qubit state in the $|0\rangle$ state and then performing a smooth split, or preparing a single-qubit $|{+}\rangle$ state and then performing a rough split, we may create a Bell pair, expressed alternatively as $\frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$ or $\frac{1}{\sqrt{2}}\big(|{+}{+}\rangle + |{-}{-}\rangle\big)$. We may represent this state by a curved wire producing two outputs, or "cup" — representing the fact that the coefficients of this tensor are the same as that of the identity matrix — if we are willing to account for the fact that this representation is super-normalised by a factor of $\sqrt{2}$ (as with the representations of $|0\rangle$ and $|{+}\rangle$ described in Eqn. (2)). We may then use this to realise a CNOT operation (with some probability) by performing a smooth and a rough merge. These merge operations may independently yield the transformation for the positive branch (realising a Kraus operator $K_{0,*}$) or the negative branch (realising a Kraus operator $K_{1,*}$).

In the following, we illustrate only the analysis in which *both* merges yield the positive branch, or the negative branch. (The cases where one merge yields the positive branch and one the negative branch are easier, but otherwise similar.) Unlike for merge operations, no logical byproduct operations are produced by lattice surgery split operations: and so it does not matter from that point of view whether the "cup" is created using a red split or a green split:

Positive branch:

$$\qquad\qquad = \qquad\qquad = \quad \sqrt{2}|\Phi^+\rangle \qquad\qquad (23\text{a})$$

We therefore treat both these cases together, in terms of "cups" as in the right-hand-side above rather than explicit preparations and splits. In the following, we sometimes commute X and Z operators (represented by 'dark' red / 'light' green π-phase nodes) past one another: this induces an unimportant global phase of -1 when we do so (which we denote by $\cong$ rather than $=$ for clarity).

Negative branch, using
$K_{1,S} = |0\rangle\langle10| + |1\rangle\langle01|$,
$K_{1,R} = |{+}\rangle\langle{-}{+}| + |{-}\rangle\langle{+}{-}|$:

$$\qquad = \qquad = \qquad = \qquad (23\text{b})$$

Negative branch, using
$K_{1,S} = |0\rangle\langle10| + |1\rangle\langle01|,$
$K_{1,R} = |{+}\rangle\langle{-+}| + |{-}\rangle\langle{+-}|:$

$$(23\text{c})$$

Negative branch, using
$K_{1,S} = |0\rangle\langle01| + |1\rangle\langle10|,$
$K_{1,R} = |{+}\rangle\langle{+-}| + |{-}\rangle\langle{-+}|:$

$$(23\text{d})$$

Negative branch, using
$K_{1,S} = |0\rangle\langle01| + |1\rangle\langle10|,$
$K_{1,R} = |{+}\rangle\langle{+-}| + |{-}\rangle\langle{-+}|:$

$$(23\text{e})$$

A.4. Split control and target, and simulate an intermediary Bell measurement.

The adjoint of the above diagrams — involving preparing the state $|\Phi^+\rangle$ and then merging the two halves into the control (smooth merge) and target (rough merge) — corresponds to a procedure in which we perform a smooth split operation on the control, a rough split operation on the target, and then perform a Bell measurement involving a child memory of each of these splits to obtain the outcome $|\Phi^+\rangle$. (The "projector" $\sqrt{2}\,\langle\Phi^+|$ is represented by a "cap", or a curved wire with two inputs, again representing the fact that as a tensor it has the same coefficients as the identity matrix.) As we cannot perform this measurement deterministically, we may simulate it using one of two different merge procedures:

$$(24)$$

In each case, both the merge and the measurement yields non-deterministic results. The measurement nodes illustrated here (representing $\langle+|$ or $\langle0|$ respectively, up to scalar factors) both have a phase of 0; the alternative outcome in each case is a phase of π (representing $\langle-|$ or $\langle1|$, respectively). We may represent the possible outcomes of the measurement using a phase $b\pi$, depending on a bit $b \in \{0, 1\}$ which indicates the result of the measurement. This will allow us to reduce the possible transformations to the positive branch, regardless of the outcome of the measurement.

(a) Bell measurement using a smooth merge

Positive branch
of merge:

$$(25\text{a})$$

Negative branch
of merge, using
$K_{1,S} = |0\rangle\langle10| + |1\rangle\langle01|:$
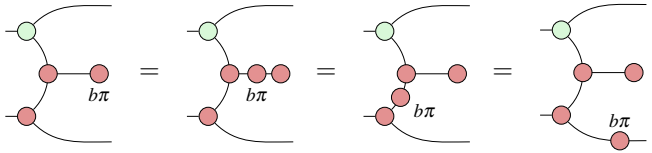

$$(25\text{b})$$

(Here, $\cong$ denotes equality up to an irrelevant phase factor of $(-1)^b$. Note that we simply absorb the red π-phase into the green degree-1 node, corresponding to the equation $\langle+| = \langle+|\,X$ of operators.)

Negative branch
of merge, using
$K_{1,S} = |0\rangle\langle01| + |1\rangle\langle10|:$

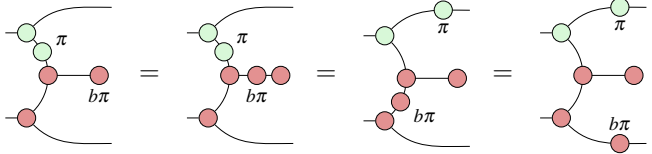
$$(25\text{c})$$

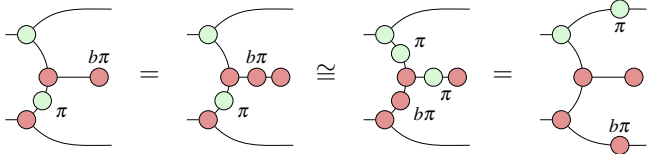**(b) Bell measurement using a rough merge**

Positive branch of merge:

(26a)

Negative branch of merge, using $K_{1,R} = |{+}\rangle\langle{-}{+}| + |{-}\rangle\langle{+}{-}|$:

(26b)

Negative branch of merge, using $K_{1,R} = |{+}\rangle\langle{+}{-}| + |{-}\rangle\langle{-}{+}|$:
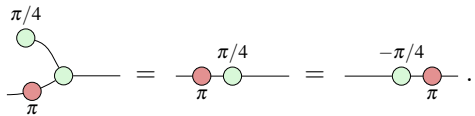

(26c)

(Here again, $\cong$ denotes equality up to an irrelevant phase factor of $(-1)^b$. Note that we simply absorb the red π-phase into the green projection node, corresponding to the equation $\langle 0| = \langle 0| Z$ of operators.)

B. Magic state merge patterns

In Section 6.1, we demonstrate how to realise a T rotation by merging a magic state with an input qubit. In this Section, we demonstrate that — similarly to the realisations of CNOT above — the choice of convention for the negative branch of the merge does not introduce any essential difficulties in the interpretation of T gate teleportation as being realised by merge operations.

The effect of the positive branch of the merge operation is the same as illustrated in Eqn. (15) regardless of the convention for the negative branch; we show that the opposite convention to (the left-hand side of) Eqn. (16) for the negative merge is equivalent. In that alternative convention, the red π-node appears on the opposite input to the 2-to-1 node than the magic state, yielding the transformation

(27)

This operator differs from the operation on the right-hand side of Eqn. (16) by a red π node, which is an X operation; the two procedures then differ only by a NOT operation controlled on the merge outcome.

Notice that the effect of the negative branch described by Eqn. (27) corresponds to the T gate teleportation procedure of [42, Fig. 10.25], but without realising this operation in terms of a CNOT gate in the form of a decomposition such as in Eqn. (5). This also demonstrates a simple example of the usefulness of the ZX calculus to simplify quantum information processing procedures, as follows. If we express the effect of the negative branch shown in Eqn. (16) in a form similar to [42, Fig. 10.25], the role of the control and target are swapped. (The target in the new circuit is still the qubit which is measured.) Interestingly however, it omits the classically-controlled NOT correction. By the interchangeability of the two conventions for adapting the Pauli frame in a merge operation, the ZX calculus thus allows us to arrive at a protocol that is simpler to realise — even for a well-established procedure in quantum information processing.