The ZX calculus is a language for surface code lattice surgery

Niel de Beaudrap

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

`niel.debeaudrap@cs.ox.ac.uk`

Dominic Horsman

Department of Physics, Durham University, South Road, Durham, DH1 1LE
Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD

`dominic.horsman@durham.ac.uk`

Quantum computing is moving rapidly to the point of deployment of technology. Functional quantum devices will require the ability to correct error in order to be scalable and effective. A leading choice of error correction, in particular for modular or distributed architectures, is the surface code with logical two-qubit operations realised via "lattice surgery". These operations consist of "merges" and "splits" acting non-unitarily on the logical states and are not easily captured by standard circuit notation. This raises the question of how best to reason about lattice surgery in order efficiently to use quantum states and operations 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 designed using category theory, 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 can 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 surgery by considering two operations (magic state use 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

As quantum computation gradually shifts from a purely theoretical concept to a mature technology, new tools will be required both to organise the components of quantum computers, and to reason effectively about the way that these systems are constructed. One of the first challenges is that of quantum error correction. In the construction of a particular device, we must determine what levels of error we are prepared to tolerate in our computation; what resources are required to protect our information to the required quality; and which computations we can do with those resources. Furthermore, to have the flexibility to adapt resources as technology improves (or as more resources become available), it is important to be able to reason effectively about how these resources are used in quantum protocols.

One of the most popular techniques for error correction is the surface code, which has a number of advantages from both theoretical and practical standpoints. The surface code has a high estimated error-correction threshold of 1%; is easily described as a stabiliser code; has a simple correction procedure which is readily adapted with heuristics; and allows a simple approach to magic state distillation. It thus provides a promising platform for universal quantum computation. Furthermore, while it allows for two-qubit logical operations to be performed transversally, it also allows for the possibility of trading physical operations against memory resources: with some additional memory elements, logical operations such as CNOT can be performed by a technique known as *lattice surgery*. This is likely to be the method

of choice for highly modular quantum computational platforms, as it would reduce the synchronisation time for quantum systems to implement the physical operations to realise a logical operation.

An intriguing feature of lattice surgery is that it necessarily involves quantum transformations of memories by CPTP maps which are not unitary transformations. Two-qubit operations such as CNOTs may be realised deterministically, using classical processing similar to that involved in standard tele-portation, but the primitive operations ("split" and "merge") do not rest easily with the standard unitary circuit model. This makes it hard efficiently to design, verify, and compile novel surgery patterns.

In this paper, in addition to a more fine-grained analysis of lattice surgery operations and corrections than appears in the existing literature, we demonstrate that lattice surgery is a model for the *ZX calculus*, an abstract graphical language for tensor networks. We show how diagrams in the calculus describe exactly the lattice surgery primitives of split and merge, including their non-deterministic byproduct operations. By transformations of these diagrams, new surgery procedures can be discovered. This work provides a ready-made syntax for computations realised using surface codes via lattice surgery, and a deeply practical spur for further development of tools for the ZX calculus.

2   The planar surface code

The planar surface code uses a 2D lattice segment, in which every edge of the lattice (between neigh-bouring vertices) is associated with a qubit [4, 15, 11, 25]. 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[1]. 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":[2] 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 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 $|1\rangle_L$.

---

[1] The "distance" of a code is the weight of the smallest non-trivial error which cannot be detected by the code.

[2] This use of the term "plaquette" differs from (and effectively generalises) the more common convention of referring to the cells of the lattice associated with the Z stabilisers as "plaquettes". We use the term "plaquette" to refer to any set of qubits, either on the boundary of a face or surrounding a vertex of the lattice.
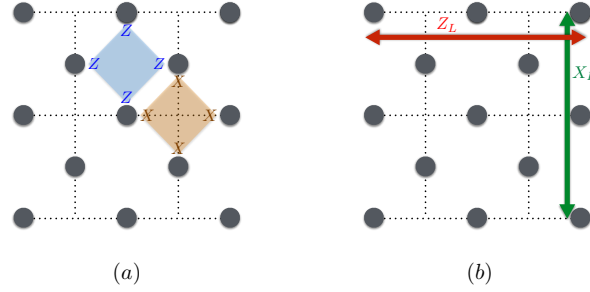
Figure 1: A distance 3 planar surface code. (a) Stabilizers on face and vertex plaquettes. (b) Support of the logical operators: a Z_L operator may be realised by a sequence of Z operators on any horizontal row of qubits connecting the two "rough" boundaries, and similarly an X_L operator my be realised by a sequence of X operators on any vertical column of qubits connecting the "smooth" boundaries.

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 stabilizers 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 classically of the accumulated errors. This information is referred to as a "Pauli frame" [20], 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[3] [17]. 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 single CPTP map from density operators on $\mathscr{H} \cong \mathbb{C}^2$ to operators on $\mathscr{H} \otimes \mathscr{H}$; and "merge" operations as taking a density operator on $\mathscr{H} \otimes \mathscr{H}$ and producing one on $\mathscr{H}$.

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,

---

[3]The other method for planar codes is transversal [11], requiring many more operations. Other procedures for surface codes involve introducing defects into the lattice, and braiding [24, 14] or twisting them [2], or by code deformation [3].
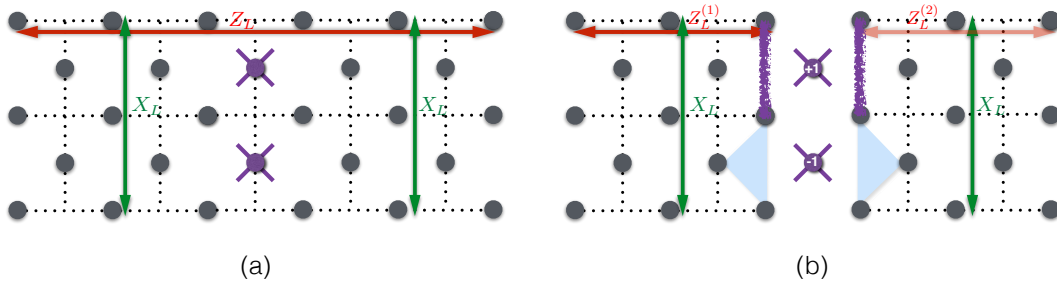
Figure 2: A rough split. (a) Purple qubits are measured out in the Z basis and determine the Pauli frame of the two new surfaces. The split copies X_L, and distributes Z_L across the two resulting surfaces. (b) Correction of a -1 measurement result on the lower qubit: a chain of X operations (purple line) joining the errored syndromes (blue shading) to a boundary.

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 $S_R(\rho) = U_R \rho U_R^\dagger$, 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 $S_S(\rho) = U_S \rho U_S^\dagger$, 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 in the $|+\rangle$ state) is added, and X-plaquette operators are measured across this join. The result is a single "child" memory, whose Pauli frame is the union of the Pauli 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
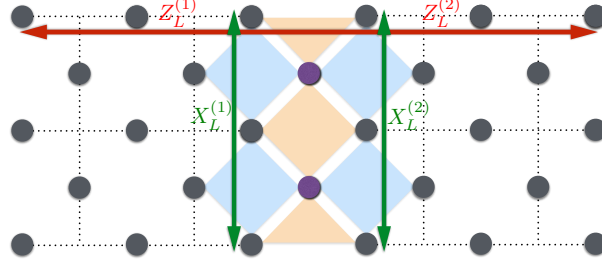
Figure 3: A rough merge. Purple qubits are initialised in $|+\rangle$. Measuring X plaquette operators (orange) across the join realises a $X_L^{(1)} \otimes X_L^{(2)}$ measurement.

child memory 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 $K_{1,R}$ depend 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 [17], 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 [16].

4   The ZX calculus

In this section, we provide an introduction to the "ZX calculus" suitable for a non-specialist reader who is familiar with the quantum circuit model. The ZX calculus is a notation together with a system of transformations, for reasoning about tensors in terms of mutually unbiased bases [8, 9]. The basic structure of this tensor notation is similar to that of the usual circuit diagrams; and 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.

The notation of the ZX calculus comprises graphs with coloured nodes (which represent operations including preparations $|\psi\rangle$ and projections $\langle\psi|$) which are connected by edges (which then represent input or output systems). The nodes come in two colours, e.g. red or green, each of which is associated to a choice of basis used to describe the effect of the node as a transformation. In all diagrams here, the time axis runs vertically up the page. For example, preparations and measurements may be denoted by

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

where each diagram fragment represents the operator written below it. (Note that setting the angle $\alpha = \pi$ 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:[4]

$$
\begin{array}{ccc}
\bullet\,\alpha & \bullet\,\alpha & \bullet\,\alpha \\
|0\rangle\langle 0| + e^{i\alpha}\,|1\rangle\langle 1| & |00\rangle\langle 0| + e^{i\alpha}\,|11\rangle\langle 1| & |0\rangle\langle 00| + e^{i\alpha}\,|1\rangle\langle 11| \\[2ex]
\bullet\,\alpha & \bullet\,\alpha & \bullet\,\alpha \\
|+\rangle\langle +| + e^{i\alpha}\,|-\rangle\langle -| & |++\rangle\langle +| + e^{i\alpha}\,|--\rangle\langle -| & |+\rangle\langle ++| + e^{i\alpha}\,|-\rangle\langle --|
\end{array}
\tag{3}
$$

---

[4]Those familiar with the ZX calculus may notice a certain non-uniformity in the scaling of the operators presented here, with the arity-1 operators of Eqn. (2). We adopt this scaling for the sake of simplicity of presentation.

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):

$$\left[\begin{array}{c} \bigcirc \alpha \end{array}\right]^{\dagger} = \bigcirc -\alpha \; ; \qquad \left[\begin{array}{c} \bigcirc \alpha \end{array}\right]^{\dagger} = \bigcirc -\alpha \; ; \qquad \left[\begin{array}{c} \bigvee \alpha \end{array}\right]^{\dagger} = \bigwedge -\alpha \qquad (4)$$

We can generalise the nodes above to ones with larger in-degree and out-degree (conventionally known as "spiders" [10]), which for any 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 horizontally 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:

$$
\begin{aligned}
&= |0\rangle\langle 0| \otimes \langle 0|+\rangle \otimes |+\rangle\langle +| \; + \; |0\rangle\langle 0| \otimes \langle 0|-\rangle \otimes |-\rangle\langle -| \\
&\quad + |1\rangle\langle 1| \otimes \langle 1|+\rangle \otimes |+\rangle\langle +| \; + \; |1\rangle\langle 1| \otimes \langle 1|-\rangle \otimes |-\rangle\langle -| \\
&= \tfrac{1}{\sqrt{2}} |0\rangle\langle 0| \otimes I \; + \; \tfrac{1}{\sqrt{2}} |1\rangle\langle 1| \otimes X \; = \; \tfrac{1}{\sqrt{2}} \text{CNOT}.
\end{aligned}
\tag{5}
$$

Using ZX as a tensor notation, we must multiply this by a scalar factor of $\sqrt{2}$ to represent CNOT exactly. (As the appropriate scalar factor may be determined by topological properties of the diagram, they are often omitted as a minor abuse of notation. However, for our results the scalar factors play an important role in understanding such diagrams as representing Kraus operators.) In this way we may represent any unitary operator with the ZX calculus, using standard results in circuit decomposition.

As a tensor notation, the ZX calculus enjoys several convenient properties which correspond to abstract representations of algebraic operations on either in the Z or the X eigenbases (such as arithmetic on bit-values in the standard basis). These properties may be expressed as transformations of diagrams which preserve equality. These properties are known as the Frobenius axioms, and 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:

$$(6)$$

These properties along with their adjoints and associativity 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 [9, Ch. 8.6.1]. 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 [9, Ch. 9.4]. The left-most equality 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, 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:

$$(7)$$

For both Eqns. (6) and (7), the reflection of the equalities about the horizontal axis also hold. 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 [19].

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. (7): 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 [9, Ch. 5.3.5], which is also the action of a rough merge. We thus appear to have an equivalence between red arity-3 nodes in the ZX calculus with rough lattice surgery operations, and between green arity-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 $|g_\alpha\rangle \propto |0\rangle + \mathrm{e}^{i\alpha}|1\rangle$ and $|r_\alpha\rangle \propto |+\rangle + \mathrm{e}^{i\alpha}|-\rangle$.

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 arity-3 nodes in Eqn. (3) for $\alpha = 0$; and similarly for the rough split/merge operations and the red arity-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:
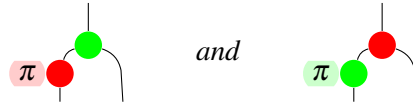
Corollary 2. *The positive branches of lattice surgery procedures (i.e. conditioned on $+1$ outcomes of all merge operations) provide a model for the equational theory of the ZX calculus without projections, in which nodes of degree 2 or 3 have an angle of 0.*

In particular: if we let ⋎ denote the unitary embedding U_S (respectively, U_R) on encoded data realised by a smooth (resp. a rough) split, ⋏ denote 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, ⟜α denote an $R_x(\alpha)$ gate (resp. a $R_z(\alpha)$ gate), and ⟜α denote 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 [9, Ch. 9.4], describing them as the effect of compositions of split and merge operations in the positive branch.
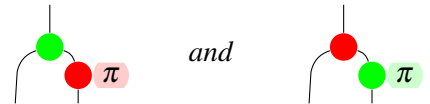
The qualification "in the positive branch" in Corollary 2 is significant, and relates to the normalisation of the diagram in Eqn. (5). Omitting the correction operations needed to realise CNOT deterministically, the standard lattice-surgical realisation of CNOT [17] is by the composition $(\mathbf{1} \otimes \mathsf{M}_R)(\mathsf{S}_S \otimes \mathbf{1})$. 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 $(I \otimes K_{0,R})(U_S \otimes I) = \frac{1}{\sqrt{2}}$ 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 both smooth and rough merges in the negative branch, consider the convention for adapting the Pauli frame in which one performs a correction on the part of the child memory which corresponds to the "first" parent memory, so that $K_{1,S} = |0\rangle\langle 10| + |1\rangle\langle 01|$ and $K_{1,R} = |{+}\rangle\langle{-+}| + |{-}\rangle\langle{+-}|$. Then these Kraus operators are equal to the operators represented by the diagrams*

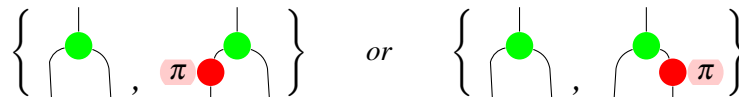$$\qquad\qquad\qquad\qquad\qquad and \qquad\qquad\qquad\qquad\qquad\qquad (8a)$$

respectively. Similarly, consider the convention for adapting the Pauli frame in which one performs a correction on the part of the child memory which corresponds to the "second" parent memory, so that $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$ and $K_{1,R} = |{+}\rangle\langle{+-}| + |{-}\rangle\langle{-+}|$. Then these Kraus operators are equal to the operators represented (respectively) by the diagrams

$$\qquad\qquad\qquad\qquad\qquad and \qquad\qquad\qquad\qquad\qquad. \qquad (8b)$$

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

Corollary 4. *A smooth merge operation may be represented by either of the two pairs of ZX diagrams,*

$$\left\{ \quad , \quad \right\} \quad or \quad \left\{ \quad , \quad \right\} \qquad (9)$$

representing the Kraus operations of the logical effect M_S of the smooth merge (where the choice of diagrams corresponds to the choice of convention, of correcting either the first parent or the second

parent respectively in the negative branch). Similarly, a rough merge operation may be represented by either of the two pairs of ZX diagrams,

$$\left\{ \raisebox{-1em}{\includegraphics{}} \ , \ \raisebox{-1em}{\includegraphics{}} \right\} \quad or \quad \left\{ \raisebox{-1em}{\includegraphics{}} \ , \ \raisebox{-1em}{\includegraphics{}} \right\} \tag{10}$$

representing the Kraus operations of the logical effect M_R of the rough merge (where again the choice of diagrams corresponds to the choice of correction convention for the negative branch).

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

The purpose of associating (the Kraus operators of) the operations of lattice surgery with simple 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. (9) (for smooth merges) and from either one of the pairs of operations of Eqn. (10) (for rough merges), and the probability of each diagram is given by its 2-norm.*

Proof. Each pair of diagrams from Eqns. (9) and (10) describe 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 arity-3 nodes (including correction operations if necessary); conversely, any given ZX diagram can be realised as Kraus operators of lattice surgery procedures. The potential for this tool as one component of a compiler should not be understated. To give a flavour of this, we here detail two examples where ZX diagrammatic equivalence gives novel lattice surgery procedures: for non-Clifford rotations, and the CNOT .

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 [5]. In the surface code, they may be distilled to high fidelity, injected into a code surface, and then used to perform teleported rotation gates [14], as these cannot be performed transversally [13]. A T gate uses a magic state and teleportation to perform
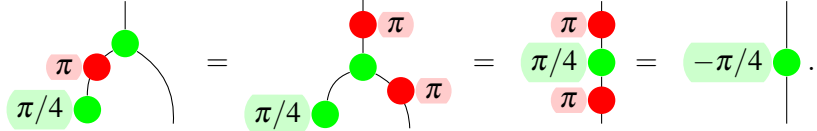
a $\pi/4$ rotation. The usual presentation of the teleported T gate involves a CNOT and measurement (see e.g. [7]). However, the ZX calculus allows us to give a much more efficient procedure for lattice surgery.

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

$$\text{(11)}$$

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
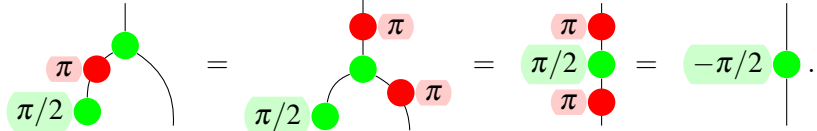
$$\text{(12)}$$

As with a standard T gate [14, §6], the rotation to correct this to $+\pi/4$ uses $|Y\rangle \propto |0\rangle + e^{i\pi/2}|1\rangle$:

$$\text{(13)}$$

The negative branch of this operation is:

$$\text{(14)}$$

This can now be corrected, if necessary, by a bit flip. The T gate has now become simple merging with magic states.[5] 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 [6].

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 [17, §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):

$$\text{(15)}$$

Visually these two presentations are topologically equivalent. By considering other topologically equivalent presentations of the coloured graph of Eqn. (5) and how they 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. (15)):

---

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

$$ (16) $$

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. (15) 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.

7 Conclusions

Almost all previous work on lattice surgery has concentrated on compiling down to a basic set of CNOT and magic state operations. This was owing to the fact that the non-unitary operations of splitting and merging (and Pauli frame updating) do not fit comfortably within the usual approach to quantum computing by unitary circuits, and no clear alternative approach was available to design lattice-surgery protocols. On the rare occasions authors have compiled down to splits and merges (for example [16]), they have had to resort to schematic-style drawings of the planar surfaces in question.

In this paper we have demonstrated a language that precisely and fundamentally describes the operations of lattice surgery. It is now possible to take a diagram in the ZX calculus, re-write it, and then interpret that diagram as a lattice surgery procedure. The equivalence shown in this paper allows us to analyse lattice surgery procedures, to determine how a ZX transformation may practically be realised.

There are a number of uses for this language; 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 [21, 22], 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. 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 [12]), for computation and also networking/communication [18, 26]. 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.

There are further uses of the ZX/lattice surgery equivalence. By producing ZX-compiled lattice surgery procedures for known algorithms and protocols, not only can (potentially significant) efficiency savings be produced, 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.

References

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

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

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

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

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

[6] E. Campbell and J. O'Gorman. An efficient magic state approach to small angle rotations. *Quantum Science and Technology*, 1(1):015007, 2016.

[7] E. Campbell, B. Terhal, and C. Vuillot. The steep road towards robust and universal quantum computation. *arXiv preprint arXiv:1612.07330*, 2016.

[8] B. Coecke and R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

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

[10] B Coecke, E. Paquette, and D. Pavlovic. Classical and quantum structures. *Semantic Techniques in Quantum Computation, eds. Gay S., Mackie I., Cambridge University Press, http://www. comlab. ox. ac. uk/files/627/RR-08-02. pdf*, 2008.

[11] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.

[12] R. Duncan and S. Perdrix. Graph states and the necessity of Euler decomposition. In *Conference on Computability in Europe*, pages 167–177. Springer, 2009.

[13] B. Eastin and E. Knill. Restrictions on transversal encoded quantum gate sets. *Phys. Rev. Lett.*, 102:110502, Mar 2009.

[14] A. Fowler, A. Stephens, and P. Groszkowski. High-threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80:052312, 2009.

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

[16] D. Herr, F. Nori, and S. Devitt. Lattice surgery translation for quantum computation. *New Journal of Physics*, (19):013034, 2017.

[17] 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.

[18] C. Jones, D. Kim, M. Rakher, P. Kwiat, and T. Ladd. Design and analysis of communication protocols for quantum repeater networks.

[19] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *International Conference on Automated Deduction*, pages 326–336. Springer, 2015.

[20] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005.

[21] 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.

[22] 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.

[23] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge UK, 2000.

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

[25] B. Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307, 2015.

[26] R. Van Meter. *Quantum Networking*. John Wiley & Sons, 2014.

Appendix

In this Appendix we show how the correction operations for the CNOT procedures of Eqns. (15) and (16) equate to logical Pauli corrections on the control and/or target qubit.
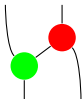
Note that in all the examples below, the transformations realised differ from the transformations realised in the positive branch only by logical Pauli operations (represented by green and/or red π-nodes). These logical Pauli operators are heralded by the outcome of the merge operations, and can therefore bed tracked or corrected. This is true regardless of which convention we adopt for the Kraus maps for the negative branch of each merge operation. This demonstrates that these protocols are not deeply sensitive to the convention used for adapting the Pauli frame of the child memory in any given merge operation.

In the following, we use the symbol $\cong$ to denote equality up to an irrelevant change of sign.
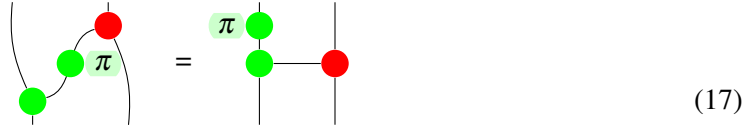
A.   CNOT patterns

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

Positive branch:

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

$$\tag{17}$$

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

$$\tag{18}$$
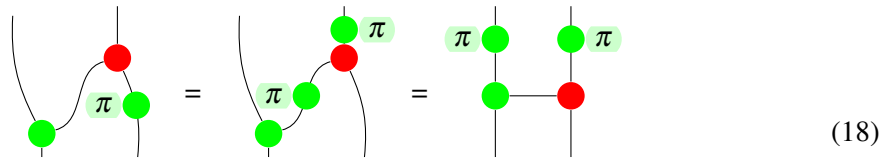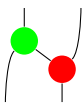
2. **Rough split target, smooth merge with control.**

Positive branch:

Negative branch, $K_{1,S} = |0\rangle\langle 10| + |1\rangle\langle 01|$:

$$ \tag{19} $$

Negative branch, $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$:

$$ \tag{20} $$

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

We may create a Bell pair expressed as a superposition of either computational ($\{|0,1\rangle\}$) or Hadamard ($\{|+,-\rangle\}$) basis states, obtaining the same result ($|00\rangle + |11\rangle = |++\rangle + |--\rangle$). Unlike for a merge, there are no correction operations from a split, and so it does not matter from that point of view whether the 'cup' is created using a red split or a green split. We therefore treat both situations identically.

Positive branch:

$$ \tag{21} $$

Negative branch $K_{1,S} = |0\rangle\langle 10| + |1\rangle\langle 01|$ and $K_{1,R} = |+\rangle\langle -+| + |-\rangle\langle +-|$:

$$ \tag{22} $$

Negative branch $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$ and $K_{1,R} = |+\rangle\langle -+| + |-\rangle\langle +-|$:

$$ \tag{23} $$

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

(24)

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

(25)

4. **Split control and target, then rough merge and measure.**

The control undergoes a smooth split, and the target a rough split. One of each pair are then rough merged (red cap node) and then immediately measured out in the computational basis (red lollipop node).

Positive branch:

(26)

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

(27)

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

(28)

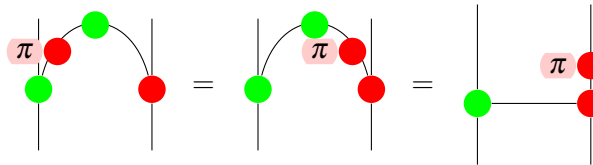5. **Split control and target, then smooth merge and measure.**

Similarly to the previous procedure, the control undergoes a smooth split, and the target a rough split. In this case, one of each pair are then smooth merged (green cap node) and then immediately measured out in the Hadamard basis (green lollipop node).
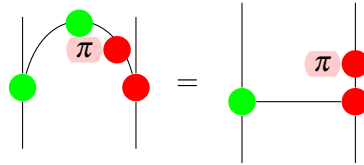
Positive branch:

$$\tag{29}$$

Negative branch, $K_{1,S} = |0\rangle\langle 10| + |1\rangle\langle 01|$:

$$\tag{30}$$

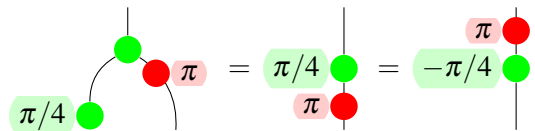Negative branch, $K_{1,S} = |0\rangle\langle 01| + |1\rangle\langle 10|$:

$$\tag{31}$$

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 (similarly to the realisations of CNOT above) that 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. (11) regardless of the convention for the negative branch; we show that the opposite convention to (the left-hand side of) Eqn. (12) 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

$$\tag{32}$$

This operator differs from the operation on the right-hand side of Eqn. (12) 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. (32) corresponds to the T gate teleportation procedure of [23, 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. (12) in a form similar to [23, 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.