UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Alejandro David Villoria González

Verification of Quantum AutoCCZ States Using the ZX-calculus

Master's Thesis (30 ECTS)

Supervisor: Dr. Dirk Oliver Theis

Tartu 2022

Verification of Quantum AutoCCZ States Using the ZX-calculus

Abstract:

The modeling, analysis, and verification of techniques used in quantum computing require formal languages designed specifically for the quantum setting. One of such techniques is the use of quantum error-correcting codes like the *surface code* to protect information from errors. The surface code protects information by defining a logical qubit inside a surface of multiple physical qubits. To execute non-Clifford gates (an essential type of computation) in the surface code, we must perform gate teleportation, a probabilistic protocol that might induce errors on the qubits. Correcting these errors can be time-consuming, but using self-correcting states like the AutoCCZ [Gidney, Fowler 2019] can significantly reduce time overheads. In the current literature, there is no formal verification of the correctness of the AutoCCZ. In this work, we verify the correctness of the AutoCCZ using the ZX-calculus, a formal language for quantum computing that consists of diagrams that can be operated on by a set of rewrite rules. We model the delayed-choice CZ (a major component of the AutoCCZ) and the AutoCCZ as diagrams in the ZX-calculus. Using the rewrite rules, we verify that the delayed-choice CZ matches its definition by being equal to the Identity or to a CCZ depending on a control bit. In the same way, we verify that the AutoCCZ corrects every possible error while reducing the time overhead compared to standard methods.

Keywords:
Formal Verification, Quantum Computing, ZX-calculus, Quantum Error Correction, Surface Code.

CERCS:
P170 Computer science, numerical analysis, systems, control.

AutoCCZ-kvantseisundite verifitseerimine ZX-arvutuse abil

Lühikokkuvõte:

Kvantarvutuses kasutatavate meetodite modelleerimiseks, analüüsiks ja verifitseerimiseks on vaja spetsiaalselt kvantarvutite jaoks loodud formaalseid keeli. Üks selliseid meetodeid on kvant-veaparanduskoodide (nagu pinnakood) kasutamine informatsiooni vigade eest kaitsmiseks. Pinnakood kaitseb informatsiooni, määrates loogilise kvantbiti mitme füüsilise kvantbiti pinna sees. Mitte-Cliffordi lüli (arvutustehte põhitüüp) täitmiseks pinnakoodis peame tegema lüli teleportatsiooni. Tegu on tõenäosusliku protokolliga, mis võib tekitada kvantbittides vigu. Nende vigade parandamine võib olla aeganõudev, kuid AutoCCZ [Gidney, Fowler 2019] sarnaste isekorrigeerivate seisundite kasutamine võib ajakulu märgatavalt vähendada. Praeguses olemasolevas kirjanduses ei leidu AutoCCZ õigsuse formaalset kinnitamist. Selles töös kinnitame AutoCCZ-i õigsuse ZX-arvutuse abil. See on kvantarvutuse formaalne keel ja koosneb diagrammidest, mida saab kasutada ümberkirjutusreeglite hulga abil. Me modelleerime ZX-arvutuses diagrammidena viitvalikuga CZ-i (AutoCCZ-i peamine komponent) ja AutoCCZ-i. Ümberkirjutusreeglite abil verifitseerime, et viitvalikuga CZ vastab oma määratlusele, olles olenevalt kontrollbitist võrdne kas identiteedi või CCZ-iga. Samamoodi verifitseerime, et AutoCCZ parandab kõik võimalikud vead, vähendades samal ajal tavapäraste meetoditega võrreldes ajakulu.

Võtmesõnad:
Formaalne verifitseerimine, kvantarvutus, ZX-arvutus, kvantveaparandus, pinnakood.

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Quantum computers are known to outperform classical devices in certain computationally expensive tasks. At their current stage, however, quantum devices have few resources and are very prone to quantum error, which can render computations useless. Because of this, Quantum Error Correction (QEC) is one of the most prominent and active research areas in quantum computing, in which many different quantum error-correcting codes like the *surface code* have been proposed to protect our systems from errors. The surface code protects computations by encoding information on a two-dimensional lattice of qubits, making computational errors less likely the more qubits we use. Although very effective against errors, this form of encoding has its limitations, such as the available set of quantum operations (also known as *gates*) we are allowed to perform in the encoded data. To circumvent this limitation, quantum operations like the CCZ gate are *teleported* rather than directly applied, a process that induces errors that can be detected and corrected later. Correcting these induced errors delay other computations and increase the chances of other harder to detect errors appearing.

For this reason, Fowler [12] introduced a new method for gate teleportation in which the corrective operations can be precomputed and applied with minimal side effects. This technique was then refined by Gidney and Fowler [16] to optimize the teleportation of the CCZ gate, which requires three corrective CZ gates. This optimization consists in embedding three precomputed CZ gates (called delayed-choice CZ gates) into the teleportation of a CCZ gate, creating a self-correcting CCZ gate named AutoCCZ.

The ZX-calculus is a formal language for representing quantum states and quantum operations in the form of diagrams. This language is also equipped with a set of rewrite rules that allows us to operate on the diagrams. Being a rigorous language, we can use these diagrams and rules to represent and prove quantum algorithms and protocols formally. In this thesis, our main objective is to use the ZX-calculus to rigorously represent and prove the correctness of the delayed-choice CZ and AutoCCZ gates. The behavior of these two constructions has not been showcased in the literature, so we take the opportunity to use the ZX-calculus to demonstrate their correctness. This way, these techniques can be trusted when using them for quantum circuit design in the future.

The thesis is structured as follows. In chapter 2 we give an overview of basic concepts in quantum computing. In chapter 3 we present the ZX-calculus and demonstrate how it can be used for reasoning about quantum computing. In chapter 4 we explain how the surface code protects quantum information and how we can make computations in it. In chapter 5 we see gate teleportation in more detail and introduce the protocols for teleporting the CZ and CCZ gates efficiently. In chapter 6 we present the results of this work, where we verify the correctness of the delayed-choice CZ and AutoCCZ. In chapter 7 we give an overview of the results and point to possible future research lines.

2 Background in Quantum Computing

In this chapter, we give a brief overview of the necessary background in quantum computing. We explain how to express a qubit in vector and matrix form and how to perform computations on it using quantum operations. We explain the more relevant quantum gates and introduce the quantum circuit notation. We refer the reader to Ref. [25, 29] for a more comprehensive introduction to the topic.

2.1 Quantum information

In quantum computing, the most basic unit of information is a quantum bit, also known as a qubit. The quantum analogue of the 0 and 1 bits are represented using Dirac bra-ket notation as $|0\rangle$ and $|1\rangle$, and their definition on vector form is

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1}$$

But unlike classical bits, whose values can only be 0 or 1, a *quantum state* can be in a *superposition* of the two. Hence, an arbitrary quantum state $|\psi\rangle$ can be expressed as a linear combination of $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,\ \alpha, \beta \in \mathbb{C}$. If we perform a *measurement* on the quantum state, we collapse the superposition into $|0\rangle$ with probability $|\alpha|^2$ or into $|1\rangle$ with probability $|\beta|^2$. We then have that $|\alpha|^2 + |\beta|^2 = 1$, defining qubits as norm-one vectors in a complex inner product space, usually referred to as *Hilbert space*. A Hilbert space $\mathcal{H} := \mathbb{C}^2$ represents a quantum system with one qubit in it. When we have multiple qubits, the quantum systems are combined via tensor (Kronecker) product, forming the combined quantum system $\mathcal{H} \otimes \mathcal{H}$.

The vectors of equation (1) form an orthonormal *basis* for $\mathcal{H}$, as every vector can be represented as a linear combination of $|0\rangle$ and $|1\rangle$. We usually refer to that basis as the *computational basis*. Another commonly used orthonormal basis for $\mathcal{H}$ is that of the $|+\rangle$ and $|-\rangle$ vectors, defined as

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2}$$

Another way to define a quantum system is via a *density matrix*. We can use density matrices to represent a quantum system more generally, as it lets us express the system as a probability distribution of arbitrary quantum states (that do not necessarily form an orthonormal basis). The density matrix that corresponds to a quantum system that is in states $|\psi_i\rangle$ with probability p_i is defined as $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. Where $\langle\psi_i|$ is the conjugate transpose of $|\psi_i\rangle$: $\langle\psi_i| = |\psi_i\rangle^\dagger$. Density matrices are positive semi-definite

($\rho \geq 0$), Hermitian ($\rho = \rho^\dagger$), and have trace 1. The set of matrices with these properties is referred to as $S(\mathcal{H})$.

2.2 Quantum computation

To perform logical computations, we manipulate a quantum state by applying *quantum operations* into it. The most general quantum operation can be defined as a map $\Phi : S(\mathcal{H}) \to S(\mathcal{H}')$ that can be written as

$$\Phi(\rho) = \sum_i K_i \rho K_i^\dagger, \tag{3}$$

where the *Kraus operators* K_i satisfy $\sum_i K_i^\dagger K_i = I$ (with I being the identity). If the set of Kraus operators consists of only one element, it follows that $K^\dagger K = I$, that is, K is a *unitary operator*. We mostly refer to quantum states using vectors and quantum operations as a single unitary transformation throughout this work. These unitary transformations are also referred to as *quantum gates* and are the quantum analogue of the logic gates in classical circuitry. An important set of quantum gates are the Pauli X, Z, and Y gates, defined as

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{4}$$

The Pauli X gate applies a bit-flip operation, mapping the basis states of equation (1) into one another. The Pauli Z acts on those same states by $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. The same happens with the second set of basis states of equation (2), but this time Z flips the states, and X adds a -1 phase to $|-\rangle$ while mapping $|+\rangle$ into itself. Notably, we can see how the computational basis states are eigenvectors of the Pauli Z, while the basis of equation (2) are eigenvectors of the Pauli X. Hence, we usually refer to these two sets of basis as the Z and X basis, respectively.

When we *measure* a state $|\psi\rangle$ on the Z basis, we project $|\psi\rangle$ into $|i\rangle$ with probability $|\langle i|\psi\rangle|^2$ and read i as the measurement outcome; with $i \in \{0, 1\}$. A similar situation occurs when measuring on the X basis, but with $i \in \{+, -\}$.

Another important group of gates is the *Clifford* gates, which are usually referred to as the Hadamard, S, CNOT, and CZ gates. To achieve universal quantum computation, that is, to have the capability to approximate any arbitrary unitary (up to a global phase), we only require to be able to apply the Clifford gates plus an additional gate, such as the T or CCZ gate. The Hadamard (H) gate maps the basis states as $|0\rangle \mapsto |+\rangle$ and $|1\rangle \mapsto |-\rangle$ and vice versa. The S and T gates are the *phase shift* gates, which map the computational basis states as $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto c|1\rangle$, where $c = e^{i\frac{\pi}{4}}$ for T and $c = e^{i\frac{\pi}{2}}$

for S. The CNOT and CZ are the two-qubit controlled-not and controlled-Z gates, which take control and target qubits and apply the Paulis X or Z to the target if the control is in the state $|1\rangle$.

Quantum computing involving these (and any other) gates is usually represented using *quantum circuits*, in which qubits are represented as straight lines that pass through quantum gates from left to right and are ultimately measured or discarded. We can find a representation of the gates we have presented in circuit and matrix form in equation (5). The control qubit of the controlled gates is identified with a (•) in its wire. In the case of the CZ and CCZ, all qubits are given the control symbol because the gate only takes action when all qubits are in $|1\rangle$. We can also represent *anti-control*, which means activation of the gate when the control qubit is $|0\rangle$; this is represented by a (∘) symbol on the control qubit.

$$
\begin{aligned}
\text{H}: \quad & \boxed{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
S: \quad & \boxed{S} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix} \\
T: \quad & \boxed{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \\
\text{CNOT}: \quad & = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\
\text{CZ}: \quad & = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\
\text{CCZ}: \quad & = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}
\end{aligned}
\tag{5}
$$

In Figure 1(a) we show how we represent measurements on the computational basis with the circuit model. The double lines represent classical information. In Figure 1(b), we

(a) (b)

Figure 1. Measurements in the quantum circuit model. (a) A Z-basis measurement. (b) An X-basis measurement.

show how we perform measurements on the X basis by applying a Hadamard gate right before a measurement on the computational basis.

Lastly, we remark two interesting quantum gates, the $R_Z(\alpha)$ and $R_X(\alpha)$, defined in equation (6). To the reader familiar with the Bloch sphere, we can think of these gates as rotations over the Z and X axis by an angle α. An interesting property of these gates is that they can make up any single-qubit unitary up to a global phase by consecutive $R_Z(\alpha) - R_X(\beta) - R_Z(\gamma)$ or $R_X(\alpha) - R_Z(\beta) - R_X(\gamma)$ rotations by some angles α, β, γ.

$$R_Z(\alpha) = \begin{pmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{pmatrix}, \quad R_X(\alpha) = \begin{pmatrix} \cos(\alpha/2) & -i\sin(\alpha/2) \\ -i\sin(\alpha/2) & \cos(\alpha/2) \end{pmatrix} \tag{6}$$

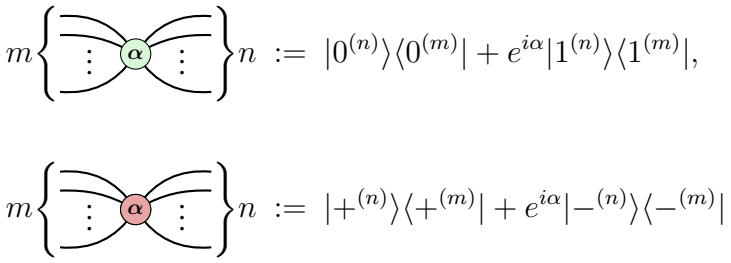
Although widely used for representing quantum algorithms and protocols, the quantum circuit model often lacks expressivity as a notation. Concepts like entanglement, gate commutations, and quantum teleportation are indeed very complicated to reason about using just gates and measurements. For this, we will now present the ZX-calculus, which is another language that incorporates significantly more tools for reasoning about quantum computing.

3 The ZX-calculus

This chapter introduces the ZX-calculus, a formal language based on diagrams used to reason about linear maps between qubits. This language is the main tool used throughout this work. We will show the generators that characterize the ZX-calculus and the set of rules used to operate on these generators. We also show the important properties of the language and discuss the motivation for using this tool instead of the quantum circuit model.

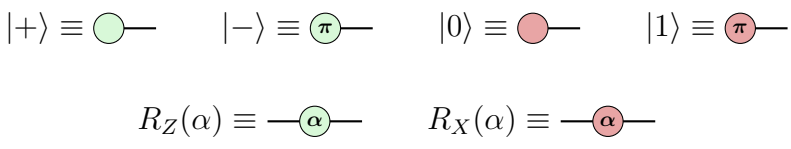
3.1 ZX-diagrams

The ZX-calculus was introduced by Coecke and Duncan [7, 8] with the intention of creating a rigorous yet intuitive language for diagrammatic reasoning of quantum computation and information (QIC). The language itself consists of two main parts: ZX-diagrams, made of nodes connected by edges, and the graphical rewrite rules one can use on them. As we will see now, the Pauli Z and X matrices and their corresponding eigenvectors are a central part of the language. This relation becomes clear when we look at the definition of the two most essential generators, the Z-spiders and X-spiders. A Z-spider (respectively X-spider) is defined as a green (red) node with any number of edges coming in and out of it, representing inputs and outputs. These spiders (and all the other generators) represent linear maps between qubits and are defined as follows:



(7)

Where $|0^{(n)}\rangle$ represents the tensor product of $|0\rangle$ n times, and $\langle 0^{(n)}|$ represents the tensor product of $\langle 0|$ n times. As we see, the Z-spider and X-spider are defined using the eigenvectors of the Pauli Z and X matrices, respectively. Moreover, by direct calculation from equation (7), we learn how to represent Z and X rotations and state preparations with ZX-diagrams[1]:



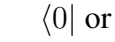
(8)

[1]Diagrams like the ones in equations (7, 8) have been generated using the program *TikZiT* (https://github.com/tikzit/tikzit) with custom styles taken from the source code of [30].

When writing equivalences between diagrams and other notations, we use (:=) when presenting a new generator, (=) for exact equalities, and (≡) for equalities up to a non-zero scalar factor. In equation (8) we remove the phase inside the spiders when $\alpha = 0$. The 1-to-1 rotation spiders correspond to the Pauli Z and X quantum gates when $\alpha = \pi$, and to the identity when $\alpha = 0$.

We can also represent measurements using annotated ZX-diagrams [9], in which we introduce boolean variables to indicate the possibility of different measurement outcomes occurring. In equation (9), the value we give later in the calculation to a will determine if we measured $|0\rangle$ (for $a = 0$) or $|1\rangle$ (for $a = 1$). Analogously, we would have measured $|+\rangle$ when $b = 0$ and $|-\rangle$ when $b = 1$.

$$\langle 0| \text{ or } \langle 1| \equiv \quad \langle +| \text{ or } \langle -| \equiv \tag{9}$$

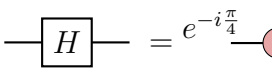


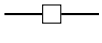
The next generator we are going to see is the *H-box*. Originally [8, 7], the H-box in the ZX-calculus was defined with just one input and one output (equivalent to a Hadamard gate, thus the name), but the *ZH-calculus* [2] introduced the version with arbitrary inputs and outputs. Although the ZH-calculus was intended to be an alternative to the ZX-calculus, we can use this new generator to extend the language [30]. The definition of the H-box is

$$m \quad a \quad n \;:=\; \sum a^{i_1 \ldots i_m j_1 \ldots j_n} |j_1 \ldots j_n\rangle\langle i_1 \ldots i_n|, \tag{10}$$

where the sum is over all $i_1, ..., i_m, j_1, ..., j_n \in \{0, 1\}$, and $a \in \mathbb{C}$. By direct calculation, we can see that an H-box with one input and output and with $a = -1$ is equal to the Hadamard gate. We will use this generator to represent the Hadamard, the CZ, and the CCZ gates. In equation (11) we can see the representation of the Hadamard gate in quantum circuit and ZX-calculus notations:

$$H \quad = e^{-i\frac{\pi}{4}} \quad \equiv \tag{11}$$

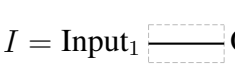




We omit the label inside the H-boxes when $a = -1$. In equation (11) we have two ZX-calculus representations for the Hadamard gate; the RHS diagram is the standard one and the one we will mainly use, while the LHS showcases how we could decompose the operation into rotation gates (with an added global phase).

The remaining generators take the form of edges (also referred to as "wires") and will help us illustrate how ZX-diagrams work while we define them. We can think of ZX-diagrams as being enclosed in a box with inputs coming from the left and outputs going

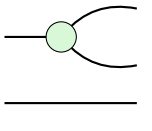
to the right. Each input-output pair denotes the state space of a qubit $\mathcal{H}$, which means that stacking vertically n inputs and outputs represents a multi-qubit quantum system $\mathcal{H}^{\otimes n}$. Knowing this, we can represent the Identity I operator as a wire connecting an input with an output:

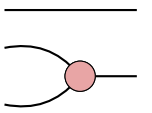
$I =$ 

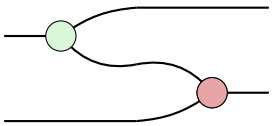
$\qquad I \otimes I =$ 

(12)

As we can see in equation (12), composing diagrams vertically means applying the tensor product of the operators each diagram represents. Similarly, horizontal composition (by connecting diagrams with a wire) entails multiplication. We can illustrate better how both vertical and horizontal composition work by defining the CNOT gate (up to a scalar) in the ZX-calculus. We will continue omitting the "box" enclosing and input and output labels:


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

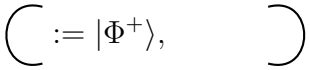

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$\rightarrow$ 
$$= \left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \right)$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \text{CNOT}$$

(13)

We have generators for representing the (unnormalized) maximally entangled two-qubit Bell state $|\Phi^+\rangle = |00\rangle + |11\rangle$ and the bell effect $\langle\Phi^+| = \langle 00| + \langle 11|$. These generators are referred to as *cap* and *cup*, respectively:


$:= |\Phi^+\rangle, \qquad := \langle\Phi^+|$
(14)

Lastly, the SWAP operation is represented as two bent wires:

$$:= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \text{SWAP} \tag{15}$$

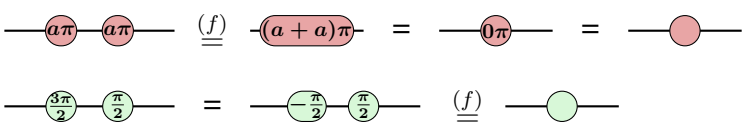
As the reader may have noticed, the ZX-calculus representations for the CNOT, cap, cup, and basis states' are up to some scalar compared to their standard matrix or vector representations. It is customary to work with these unscaled representations for convenience. If scalar values or normalization are relevant to the computation, the ZX-calculus provides a way to multiply diagrams by any complex number using 0-input, 0-output diagrams [30, Section 3.4]. For this same reason, whenever we get a sub-diagram with no inputs or outputs (a scalar diagram), we eliminate it from the computation.

3.2 The graphical rules

We will now see the rules we can use to manipulate ZX-diagrams. Our intention here is to introduce the core set of rules and some derived rules, meaning that the ruleset shown is not minimal. For a minimal set of rules (including proofs), we refer the reader to [30].

Figure 2 shows the rules of the ZX-calculus. We have also included rules from the ZH-calculus, which are the ones involving H-boxes of arbitrary arity. All rules (unless we specify otherwise) still hold if we switch the colors of the spiders, negate the phases of the spiders, or interchange the inputs and outputs. We are now going to mention how they operate briefly.

- The (f)usion rule shows how we can fuse and unfuse spiders of the same color by summing their phases and combining inputs and outputs into the same node. When summing the angles, we always do it modulo 2π. This means that $\pi \equiv -\pi$ and $2\pi \equiv 0$. When the phase is conditioned by a boolean variable similar to what we had in equation (9), the addition of the boolean variables is modulo two. An example of these cases can be seen in equation (16).



(16)

- The (π)-copy rule states that we can commute 1-to-1 spiders with phase $\alpha = \pi$ through a spider of the opposite color by negating the phase of the latter and

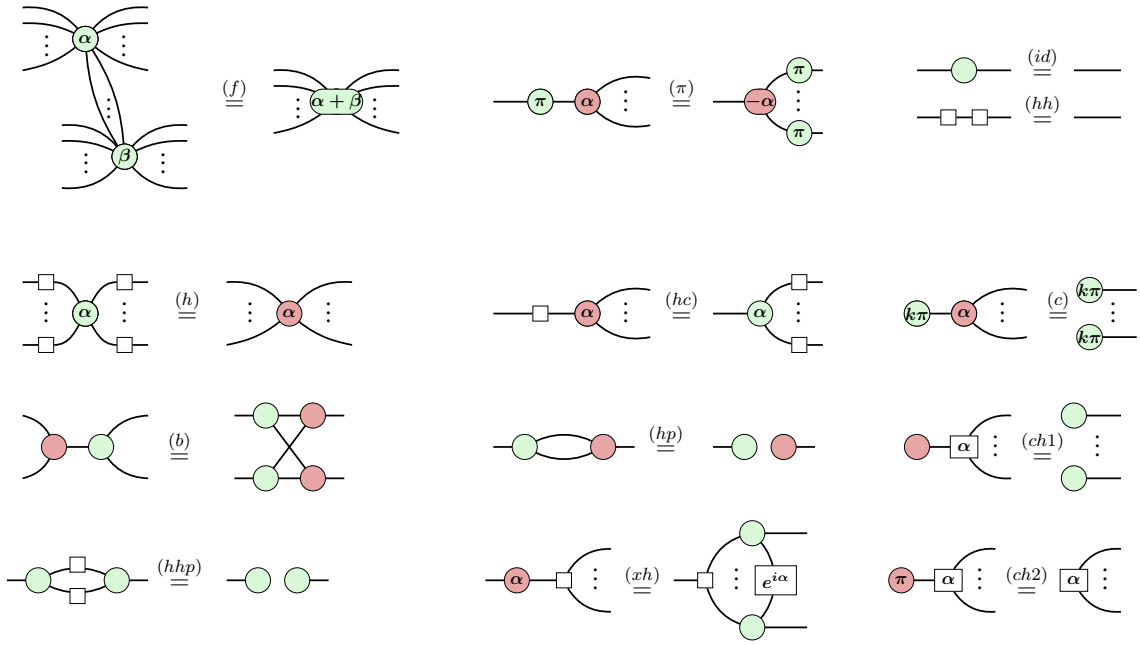
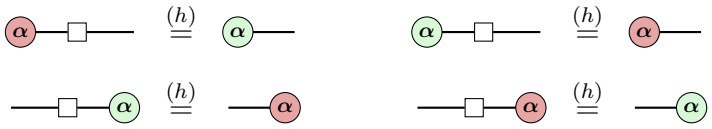


Figure 2. Rules of the ZX-calculus. We have also included some rules that arise from the generalization of H-boxes introduced in the ZH-calculus.

copying the former into all the other wires. As we will learn later, this rule expresses how well Pauli matrices commute through most quantum gates.

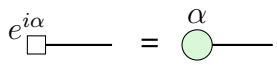
- The (id)entity rule demonstrates how we can add and remove 1-to-1 spiders with phase $\alpha = 0$ without changing the value of the overall diagram. And the (hh)-deletion rule shows the self-inverse properties of the Hadamard gate.
- The (h)adamard rule expresses that a spider with a Hadamard gate in every wire absorbs these gates by changing colors. This also applies to states and measurements, as we see in equation (17).



(17)

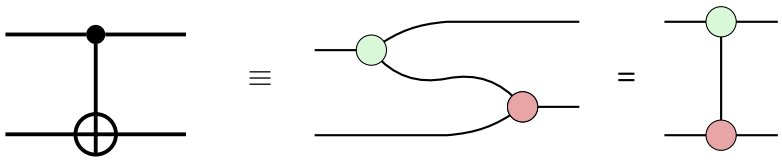
- We can derive (hc) from (h): a Hadamard gate commutes through a spider by changing its color and being copied into the other wires.
- Similar to (π), the (c)opy rule shows that we can copy basis states through a spider of the opposite color. We remind that basis states are arity-1 spiders with phases $\alpha = 0$ or $\alpha = \pi$, so $k \in \{0, 1\}$.

- The (b)ialgebra and Hopf (hp) rule are the other two ZX-calculus rules in which we can alter the connectivity of spiders.
- The (xh), $(ch1)$, and $(ch2)$ rules address how X spiders interact with H-boxes and thus do not work with inverted colors. $(ch1)$ shows that an H-box connected to $|0\rangle$ copies this state as $|+\rangle$ in the other outputs, while $(ch2)$ shows that $|1\rangle$ is absorbed. (xh) shows that an X spider creates a multi-controlled Z phase gate on the other side of an adjacent H-box.
- Lastly, the (hhp) rule lets us disconnect spiders of the same color that are attached by an even number of wires with Hadamards in them. The following equivalence also holds for H-boxes, where we take the opportunity to show that sometimes we will write the phase of an H-box or a spider *outside* the gadget itself:



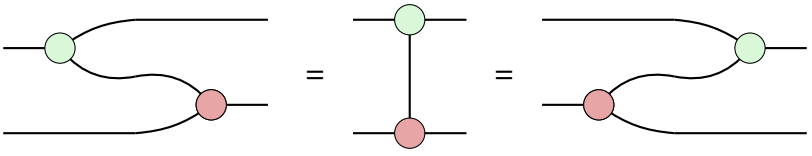
(18)

Apart from Figure 2, we have a more general rule stating that, as long as the connectivity between nodes is left unchanged, we can alter our diagrams without changing their meaning. This means that we can enlarge, contract, and bend wires at will and can move nodes around the diagram without changing the underlying matrix, given that we preserve all connections between nodes (including connectivity between nodes and inputs/outputs). With this rule in mind, we can make the representation for the CNOT operation of equation (13) more compact:



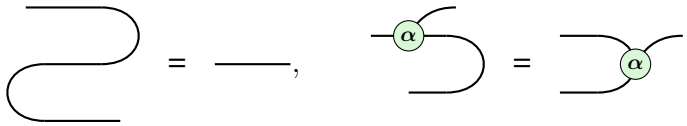
(19)

And, by the same rule, we can bend the wire to the other side:



(20)

As we can see, we can evaluate a vertical wire as being an input to one spider and output to the other (and vice versa) without modifying the matrix it represents. We can identify the next equalities as bending wires and sliding spiders through wires:

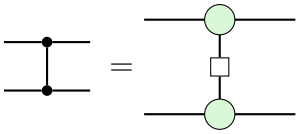


(21)

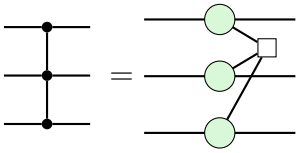
Another convenient feature of ZX-diagrams is that they are not braided. Crossed wires do not distinguish which is at the top or the bottom [30]:

(22)

Now that we are familiar with vertical wires, we can also see how to represent the CZ and CCZ gates in the ZX-calculus:



(23)



(24)

3.3 Universality, soundness, and completeness

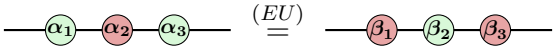
With the generators and rules introduced, it is essential to know how powerful these representations of quantum operations are. We are interested in knowing if ZX-diagrams can represent any map between qubits (*universality*), if the ruleset preserves the equality of the matrix representations for any diagram (*soundness*), and if we can transform any two equal diagrams into one another with our set of rules (*completeness*).

As we saw earlier, any single-qubit unitary can be represented as a succession of the Z and X rotation gates of equation (6). With these gates and an extra two-qubit gate like the CNOT, we have a universal set of quantum gates. Since we can represent both the rotation and CNOT gates, as seen in diagrams (8, 13), we can conclude that ZX-diagrams can represent any unitary transformation on qubits. Furthermore, we are not restricted to unitary transformations; we can represent *any* linear map between qubits using ZX-diagrams, making our diagrams universal [8].

Soundness in the ZX-calculus would ensure that the rules from Figure 2 preserve the underlying matrices when we apply them inside larger diagrams [30]. Denoting by D_1, D_2 any two diagrams and by $[\![D_1]\!], [\![D_2]\!]$ their underlying matrices, our rewrite rules from Figure 2 asserting $D_1 = D_2$ are sound (i.e., $[\![D_1]\!] = [\![D_2]\!]$) by direct calcultation. To

check whether this still holds when we apply them in larger diagrams, we do the following. Having a diagram being part of a larger diagram by either horizontal composition $D_1 \circ E_1$ or vertical composition $D_1 \otimes E_2$, for some other diagrams E_1, E_2, we need to check if $D_1 = D_2$ holds then $D_1 \circ E_1 = D_2 \circ E_1$ and $D_1 \otimes E_2 = D_2 \otimes E_2$ also hold. Since we compose diagrams by composing their matrices, we have that $[\![D_1 \circ E_1]\!] = [\![D_1]\!] \circ [\![E_1]\!] = [\![D_2]\!] \circ [\![E_1]\!] = [\![D_2 \circ E_1]\!]$. Similarly, $[\![D_1 \otimes E_2]\!] = [\![D_1]\!] \otimes [\![E_2]\!] = [\![D_2]\!] \otimes [\![E_2]\!] = [\![D_2 \otimes E_2]\!]$, which confirms the soundness of our rewrite system.

The last property is completeness, which ensures that any two diagrams with the same matrix representation can be transformed into one another using a specific set of rules. Different rule sets have been proposed to give completeness to the ZX-calculus, some of which shared most of the ZX-rules of Figure 2 but had to introduce complicated rules to ensure completeness [30]. The simplest rule that combined with the ones of Figure 2 give completeness is the (EU)ler rule [32]:



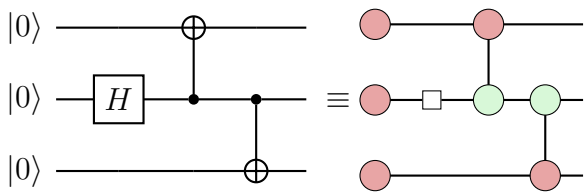
(25)

Equation (25) shows how we can find two sets of rotations on the Bloch sphere that give the same unitary, although the relation between $\alpha_1, \alpha_2, \alpha_3$ and $\beta_1, \beta_2, \beta_3$ is difficult and involves complicated computations.

3.4 Applications of the ZX-calculus

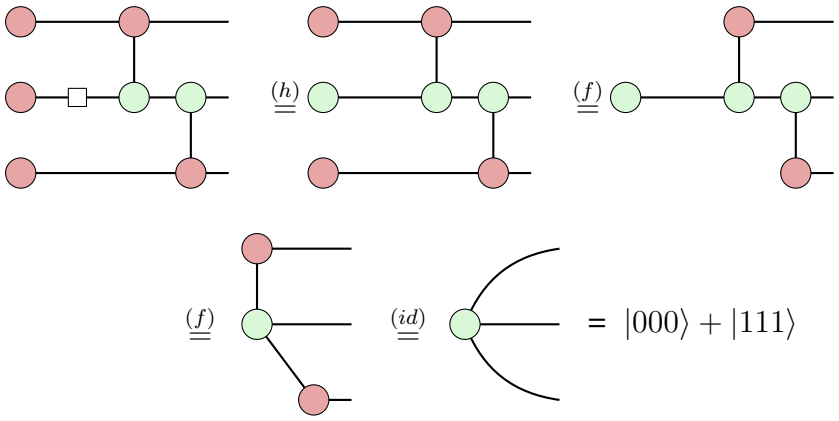
Having a universal set of generators and a complete set of rules, we can use the ZX-calculus for any reasoning about quantum computing. There is great potential in all the structure behind ZX-diagrams and their rewrite rules. Diagram simplification can lead to optimization (i.e. gate count reduction) of quantum circuits [10, 3]; completeness ensures that we can verify that a sequence of quantum operations (in ZX-diagram representation) lead to a specific quantum state [21], and being able to manipulate quantum operations without recurring to matrix representation or Dirac bra-ket notation can overall help in reasoning about QIC [9, 19].

We mainly use the ZX-calculus in this thesis to verify circuit equivalences, namely, given a large quantum circuit (in our case depending on multiple measurement outcomes), ensure that the desired computation is always executed. We can illustrate a very simple example of how the ZX-calculus can be used for this task: given the quantum circuit for the preparation of the state $|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, we convert the circuit to ZX-calculus:



(26)

and then use the rules of Figure 2 to verify the operation:



(27)

We can observe that the last expression of equation (27) is equal (up to normalization factor) to $|GHZ\rangle$. As we can see, the ZX-calculus gives a very intuitive and efficient framework for verifying quantum operations. Simplifying ZX-diagrams usually consists of iteratively looking for local parts of a larger diagram where a rewrite rule can reduce the number of vertices or edges until we can apply no more rules or we have reached the desired end diagram. We will use the ZX-calculus for verification of circuits operating in the context of the Surface code, which we are going to introduce now.

4 Quantum computing with the Surface code

In this chapter, we introduce the topic of fault-tolerant quantum computing with the surface code. We explain the necessary background on the stabilizer formalism, how qubits are encoded in a planar surface code lattice, how errors are detected and corrected, and how universal quantum computation can be achieved with lattice surgery and gate teleportation.

Similar to its classical counterpart, quantum computation is not absent of noise. What separates classical from quantum error-correction is the many restrictions the latter has due to its quantum nature, namely the impossibility of duplicating arbitrary quantum data due to the *no-cloning theorem*, and the destruction of information from measurement [25]. We are interested in a concrete form of quantum error correction: the surface code, an error-correcting code introduced by Kitaev [22]. The surface code works by encoding a *logical* qubit in a 2D lattice of physical qubits. The extra physical qubit redundancy protects the encoded qubit from local errors and enables error detection without destroying information. The surface code relies heavily on concepts related to *stabilizers*, which we are going to present now.

4.1 The stabilizer formalism

This section is a summary of the essential stabilizer concepts needed for our purposes. For a more thorough analysis, we refer the reader to Refs. [6] and [25, Section 10.5].

To understand the stabilizer formalism, we will review the properties of the Pauli matrices and certain group theory concepts. We will start by defining the Pauli group on a single qubit as the set of matrices

$$G_1 := \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z\rangle. \tag{28}$$

The group of equation (28) is closed under multiplication, and its elements $\{X, Z, Y, I\}$ refer to the set of Pauli matrices plus the identity matrix. The notation $\langle g_1, \ldots, g_k\rangle$ is another way of defining a group by an independent set of *generators* that, via the group operation, can generate the whole group. The Pauli group on n qubits G_n can be defined as the set of matrices of equation (28) acting on each qubit via n-fold tensor product.

We are interested in particular subgroups of G_n that are *commutative* and do not have the element $-I$ in them for the following property: Let S be an abelian (commutative) subgroup of G_n such that $-I \notin S$, there exists a non-trivial subspace of $\mathcal{H}^{\otimes n}$ that is invariant under S. A subspace V_S is invariant under a group S if for any quantum state $|\psi\rangle \in V_S$ and for any $g_i \in S$ we have $g_i|\psi\rangle = |\psi\rangle$; we say that S *stabilizes* V_S.

The condition $-I \notin S$ is required since the only vector stabilized by the operator $-I$ (meaning that $-I|\psi\rangle = |\psi\rangle$) is the zero vector.

The set of generators g_i that can form such subgroup S have the property $g_i^2 = I, \forall i$. This implies that the eigenvalues of such operators are ± 1. When we measure one of these stabilizers g_i on any quantum state $|\phi\rangle \in \mathcal{H}^{\otimes n}$ using projectors $(I \pm g_i)/2$, we project $|\phi\rangle$ onto $|\phi^{\pm}\rangle$ which is now a ± 1 eigenstate of g_i, depending on the measurement outcome. The probability of measuring ± 1 on $|\phi\rangle$ (and thus projecting the state onto the ± 1 eigenspace) is defined as

$$\begin{aligned} p(+1) &= \text{tr}\left(\frac{I + g_i}{2}|\phi\rangle\langle\phi|\right) \\ p(-1) &= \text{tr}\left(\frac{I - g_i}{2}|\phi\rangle\langle\phi|\right) \end{aligned} \tag{29}$$

Moreover, the properties of our subgroup S give us the ability to define a projector P_S onto the joint eigenstate of all $g_i \in S$; that is, a projector onto V_S:

$$P_S = \frac{1}{|S|}\sum_{g \in S} g \tag{30}$$

where $|S|$ is the order (number of elements) of S.

In the surface code, our *codespace* (the space containing our logical qubit) is the joint eigenstate of certain stabilizers, achieved by applying P_S to multiple qubits. Apart from these stabilizers, we will have logical operations and errors in our system. Logical operators preserve the code, meaning that the joint eigenstate will preserve its eigenvalues with respect to each stabilizer after the operation is applied. The main condition for code preservation is commutativity: if an operation O commutes with *all* the stabilizers g_i of a group S we have $\forall|\psi\rangle \in V_S$ that $g_i O|\psi\rangle = O g_i|\psi\rangle = O|\psi\rangle$. Hence, we can operate *logically* on the encoded qubit without perturbing the codespace. On the other hand, errors will take the form of Pauli operators that *anti-commute* with some of the stabilizers in S. If an error E that anti-commutes with some stabilizer g_i takes place, the faulty state $E|\psi\rangle$ will still be an eigenvector of g_i, but the eigenvalue will be flipped: $g_i E|\psi\rangle = (-1)E g_i|\psi\rangle = (-1)E|\psi\rangle$. Thus, we can see how measuring stabilizers can be used as a form of error detection: if we measure a stabilizer and the outcome is different from a previous measurement, then our system has been subject to an error equivalent to the Pauli operation that anti-commutes with that stabilizer.

Encoding qubits in the joint eigenstate (also called "quiescent state") of stabilizers and detecting errors via stabilizer measurements are at the core of the surface code. We will

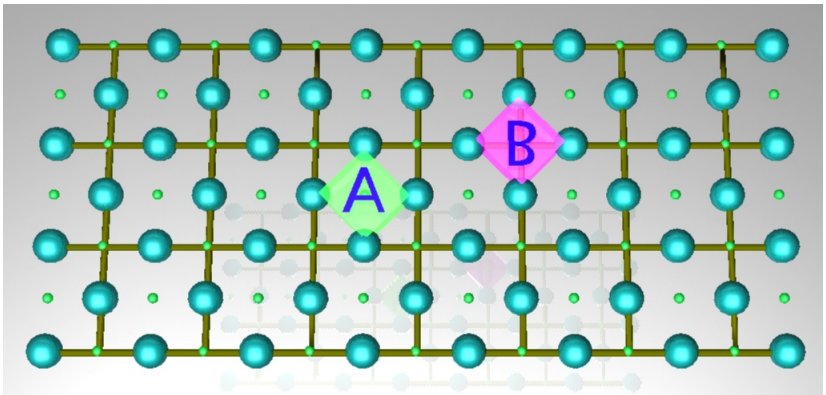


Figure 3. A 2D surface code lattice [20]. Physical qubits are represented as large and blue (data qubits) and small and green (measurement qubits) spheres. Lines connecting qubits are for visual clarity and depict vertex plaquettes where they intersect and face plaquettes where they are absent. An example face plaquette is labeled 'A', and an example vertex plaquette is labeled 'B'.

now see in more detail how the surface code uses the tools provided by the stabilizer formalism to create a state-of-the-art error correction and detection protocol.

As we know, the surface code uses a 2D lattice of physical qubits to encode a logical qubit in the quiescent state. There are different ways to define this logical qubit inside the quiescent state: the *planar* version defines *rough* and *smooth* boundaries in the lattice [5, 20], the *defect* version removes a stabilizer from the system to introduce a degree of freedom [12], and another method introduces *twists* in the lattice [4]. In this work, we are interested in the planar version of the surface code.

4.2 Surface code stabilizers

In a surface code lattice that employs n physical qubits, we require $n-1$ stabilizers to create a quiescent state with 2 degrees of freedom (the necessary to encode a logical qubit). These stabilizers take the form of Pauli X and Z operators in tensor product form:

$$\bigotimes_{i\in F} Z_i, \text{ or } \bigotimes_{j\in V} X_j, \tag{31}$$

where F and V are the set of indexes for the qubits located around the faces and vertices of the lattice, respectively. In Figure 3 we can see an illustration of the two types of stabilizers we have: the face stabilizer with label A has the four nearest qubits in a simultaneous eigenstate of $Z \otimes Z \otimes Z \otimes Z$, while the vertex stabilizer with label B has

the four qubits around the vertex in an eigenstate of $X \otimes X \otimes X \otimes X$. Note that each stabilizer acts on three or four qubits, depending on if it is located on the boundaries of the lattice or not. These stabilizers can be seen as local relationships (also called "plaquettes") between qubits, as they all act in a nearest-neighbor manner.

The stabilizers of equation (31) satisfy the properties to form a stabilizer subgroup of the Pauli group G_n. The property $g^2 = I$ is clear from the very properties of the X and Z Pauli matrices, which satisfy $X^2 = Z^2 = I$. This means that all the stabilizers of the surface code have eigenvalues ± 1. We can also see the commutativity property satisfied [12]: Stabilizers that don't share any qubits trivially commute; the same happens with stabilizers of the same type. Stabilizers of different types that also share qubits commute since they always share two qubits (see Figure 3): let any face stabilizer $Z_{abcd} = Z_a \otimes Z_b \otimes Z_c \otimes Z_d$ and vertex stabilizer $X_{abef} = X_a \otimes X_b \otimes X_e \otimes X_f, \forall a, b, c, d \in F$ and $\forall a, b, e, f \in V$ we can verify their commutativity on the two qubits they intersect:

$$\begin{aligned}\Big[Z_{ab}, X_{ab}\Big] &= (Z_a \otimes Z_b)(X_a \otimes X_b) - (X_a \otimes X_b)(Z_a \otimes Z_b) \\ &= Z_a X_a \otimes Z_b X_b - X_a Z_a \otimes X_b Z_b \\ &= (-1) X_a Z_a \otimes Z_b X_b - X_a Z_a \otimes X_b Z_b \\ &= (-1)(-1) X_a Z_a \otimes X_b Z_b - X_a Z_a \otimes X_b Z_b \\ &= 0\end{aligned} \tag{32}$$

4.3 Logical Pauli operations

Having the stabilizers defined, it is essential to see how we can describe logical operations on the surface code. First, we are interested in finding the representation of the Pauli X and Z operations. These two operators are of particular importance because, apart from their use in computing bit and phase flips, they define the behavior of the logical qubit. Indeed, we can define the logical $|0\rangle$ as the joint eigenstate of all stabilizers and the $+1$ eigenvector of the logical Z operator. In the same way, our logical $|1\rangle$ is the -1 eigenvector of Z. From this, we can see that, since X anti-commutes with Z, X maps $|0\rangle$ to $|1\rangle$ and vice versa. We summarize the relationships our operators must satisfy to define a logical qubit in equation (33).

$$\begin{aligned} Y &= ZX, \\ X^2 &= -Y^2 = Z^2 = I, \\ XZ &= -XZ, \\ \Big[X, Y\Big] &= XY - YX = -2Z \end{aligned} \tag{33}$$

We know from our stabilizer formalism background that logical operations must commute with all the stabilizers. Apart from that, they must also satisfy the relations of equation (33). We will start by defining the logical X operation on the surface code, which we will refer to as X_L. This operation is achieved by applying single-qubit X gates on the physical qubits, such that the overall procedure satisfies the required properties.

Addressing the property of commutativity with the stabilizers, we know that X operations on individual qubits commute with adjacent X stabilizers and anti-commute with adjacent Z stabilizers (they trivially commute with non-adjacent stabilizers). To make a succession of individual X gates commute with the Z stabilizers, we need to apply X to two qubits in the same Z plaquette. The reason for applying two anti-commuting operations to a stabilizer is analogous to why the face and vertex stabilizers commute: we switch eigenspaces twice, returning to the original stabilized subspace.

To find a way to create a chain of single-qubit X operations that operate twice with nearby face stabilizers, we need to take a closer look at the arrangement of the stabilizers in the lattice. More specifically, we want to focus on the four boundaries our lattice has. Looking at Figure 3, we can find similarities between boundaries that are on opposing sides of the lattice. Indeed, the lattice terminates with vertex stabilizers at the top and bottom and with face stabilizers on the left and right. We refer to the top and bottom boundaries as *smooth* boundaries, while the left and right are called *rough* boundaries [12]. We can see how applying an X gate to all the qubits alongside a rough boundary commutes with the nearby Z stabilizers: since we start and end the chain on the lattice's vertical limits, we can ensure that we always operate on two qubits of the same plaquette, making the X chain commute with the Z stabilizers. It is also true that there is no need for the X chain to be applied on the boundary qubits: as long as the chain traverses the lattice vertically (or, in other words, connects the smooth boundaries), the chain will commute with the Z stabilizers. Then, we have many different options to implement the commuting X chain. These options perform the same operation on the encoded qubit, as they differ from each other by some vertex stabilizers that don't affect the logical qubit.

In the same way, a chain of single-qubit Z operations connecting the rough boundaries (by traversing the lattice horizontally) commutes with all the X stabilizers. In Figure 4 we can visualize the two chains on a square lattice as the logical operators X_L and Z_L. To ensure that these chains indeed perform logical X and Z operations, we need to check the relations of equation (33). We can easily verify that $X_L^2 = Z_L^2 = I$ since the single-qubit gates we are applying satisfy this property. We can verify $X_L Z_L = -Z_L X_L$ by looking at Figure 4: the chains of single-qubit X and Z operations work in different qubits (thus commuting) except for one qubit where they anti-commute, making the overall operations anti-commute. Finally, with $Y_L = Z_L X_L$, we can see that $-Y_L^2 = I$. Then, we can conclude that we have a set of logical operations X_L, Z_L, Y_L that satisfy

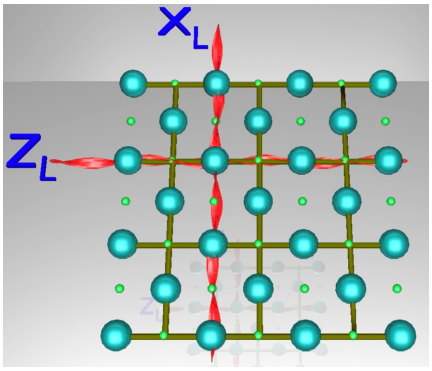


Figure 4. Logical X_L and Z_L operators as single-qubit X and Z chains [20].

the necessary relations to encode a logical qubit in the lattice.

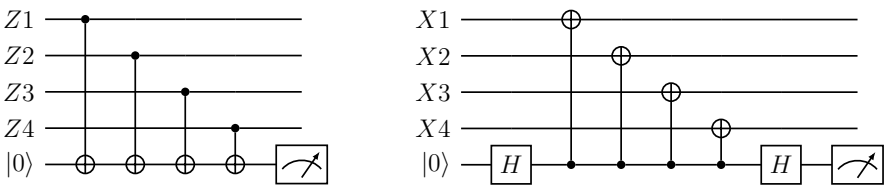
Measuring the logical qubit in the Z or X basis is similar to applying X_L and Z_L. Indeed, if we measure a chain of qubits connecting the rough boundaries on the computational basis, then the parity of the results is the outcome of the measurement for the logical qubit. Similarly, measuring a chain of qubits connecting smooth boundaries on the X basis will yield the result of an X measurement on the logical qubit.

4.4 Error correction and detection

We will see now how errors on individual physical qubits can affect our code. Errors take the form of single-qubit bit-flip Pauli X errors, phase-flip Z errors, or a combination of both, making Pauli Y errors. We can see that a single-qubit Pauli X or Z error on a physical qubit i anti-commutes with any face or vertex (respectively) stabilizer that acts on i. Y errors anti-commute with both vertex and face stabilizers. Consequently, we can detect a single-qubit error if we see a change in the outcome of a stabilizer measurement. The process of performing stabilizer measurements on the surface code in order to detect errors is known as *syndrome measurements*.

We measure a stabilizer on the surface code by entangling data qubits with auxiliary measurement qubits; we will then measure these auxiliary qubits to read out the stabilizer measurement. In Figure 3 we can see how we position measurement qubits inside the plaquette of the nearest data qubits. In Figure 5 we showcase how the measurement of Z and X stabilizers is done in circuit notation. The measurement outcome tells us whether the plaquette data qubits are in the ± 1 eigenspace of the stabilizer, so detecting a change in this value entails that an error has occurred in one of the qubits.

If multiple errors of the same type appear in adjacent qubits, then an "error chain" is formed. Syndrome measurements only report errors in plaquettes that have qubits at the end of a chain of errors, as having an even number of errored qubits does not change



(a) (b)

Figure 5. Circuits for syndrome measurements. (a) Z-stabilizer measurement. Involved data qubits are represented as Z_i. (b) X-stabilizer measurement. Involved data qubits are represented as X_i.

the eigenvalue of the stabilizer. Finding the chain of errors that has led to a specific syndrome measurement is known as decoding, and great efforts have been made to find optimal solutions to this problem. Several algorithms have been proposed to solve this problem, like minimum weight perfect matching decoding [13], cellular automaton decoding [18], and machine learning-based decoding [14, 31, 28]. Despite the efforts, it is an open problem to realize an algorithm with high accuracy that can operate reasonably fast within the currently achievable logical qubit coherence times [31]. Corrections for detected errors are not applied to the physical qubits; instead, we track them in classical control software to correctly interpret future results. Since Pauli errors will only flip the measurement result of nearby stabilizers, the control software can "correct" the error classically by simply flipping the measurement results of the affected stabilizers again.

4.5 Lattice surgery

Having studied how a single surface code lattice works, it remains to understand how we can make multiple lattices interact to perform two-qubit gates. To this end, we will now introduce lattice surgery [20] procedures, which consist of "splitting" and "merging" lattices by manipulating their boundaries. The split and merge are logical operations on lattices that increase and reduce (respectively) the number of surfaces present in our system. Depending on the lattice boundaries they act upon, the split and merge can be either "smooth" or "rough."

4.5.1 Splitting

We can define the lattice surgery split as a map from states on $\mathcal{H}$ to $\mathcal{H} \otimes \mathcal{H}$. This operation divides a planar lattice by performing measurements on data qubits positioned in a way that will create new rough or smooth boundaries, thus dividing the lattice in two and creating another logical qubit.

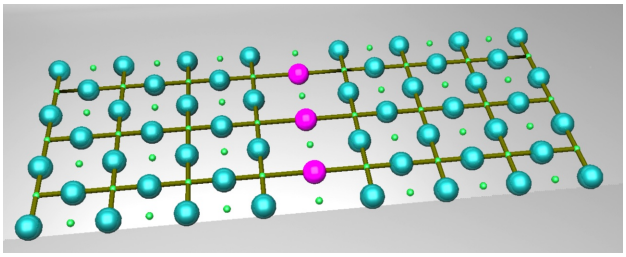


Figure 6. A smooth split [20]. Purple qubits are the data qubits that we measure on the X basis to separate the large lattice in two. Vertex stabilizers connected to the purple qubits might flip their eigenvalue.

The smooth split (illustrated in Figure 6) takes a planar surface code lattice and divides it into two lattices by performing X basis measurements on a chain of physical qubits. The measured qubits will now be between the two new lattices, thus creating new smooth boundaries. Performing these measurements can lead to some side effects on the stabilizers adjacent to the new boundaries, depending on the outcomes. Indeed, if we measure -1 on any qubit, the plaquettes adjacent to this qubit will go to their -1 eigenspace. This error is corrected by a chain of single-qubit Z operations connecting the faulty plaquettes to the boundaries in a way that will flip again the stabilizers with eigenvalue -1 and don't disturb the rest. The operators X_L and Z_L are also affected by the lattice splitting. When smooth splitting, Z_L can still be achieved in both child lattices by chaining Z gates connecting rough boundaries. On the other hand, the original X_L can't be achieved as the original rough boundaries (that we must traverse with a chain of X gates) have been separated; we now implement X_L on the new lattices by traversing their respective (and smaller) rough boundaries with a chain of X gates. By how the original X_L and Z_L are affected, we can reason about the smooth split as an operation that preserves the eigenvalue of the Z_L before the split while splitting it for X_L across the two new surfaces. This operation may be represented as the unitary

$$U_S := |00\rangle\langle 0| + |11\rangle\langle 1|. \tag{34}$$

The rough split works very similarly to the smooth version. In fact, if we interchange X and Z, and smooth and rough boundaries in the explanation above, we obtain the rough splitting procedure. Since the rough split introduces discontinuities in the smooth boundaries, X_L is the operator that will be preserved after the operation, while Z_L is split across the two new surfaces. The unitary operation that represents this procedure is

$$U_R := |++\rangle\langle +| \; + \; |--\rangle\langle -|. \tag{35}$$

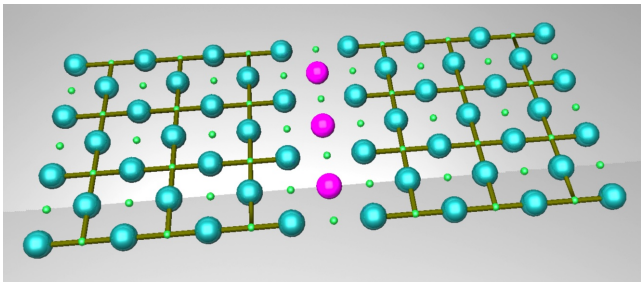


Figure 7. A rough merge [20]. Auxiliary qubits (in purple) are placed between two surfaces. Then, a new row of vertex stabilizers are measured, reducing the degrees of freedom and unifying the two lattices into a larger one.

4.5.2 Merging

Merging planar lattices can be considered the "inverse" of a split, where a quantum system in $\mathcal{H} \otimes \mathcal{H}$ is sent to $\mathcal{H}$. We merge two independent surface code lattices (each encoding a different logical qubit) by preparing a row of auxiliary qubits between them and then measuring new stabilizers on the union, resulting in a unified lattice encoding a single logical qubit. A rough merge unifies two lattices along their rough boundaries (pictured in Figure 7), while the smooth merge joins the lattices by their smooth boundaries.

We produce a rough merge by preparing the auxiliary qubits in $|0\rangle$ and introducing new vertex stabilizers in the system by measuring new X-plaquettes on the union. This process unifies the two lattices, which will now encode a logical qubit. As with splitting, we must see the effect of this operation on the operators Z_L and X_L. Let us denote the logical Z_L operation on the original surfaces by $Z_L^{(1)}$,$Z_L^{(2)}$, while denoting the one on the child lattice by $Z_L^{(3)}$ (we do the same with X_L). We can see how rough merging affects the Z_L operation: $Z_L^{(3)}$ now spans where $Z_L^{(1)}$ and $Z_L^{(2)}$ operated, resulting in $Z_L^{(3)} = Z_L^{(1)} \otimes Z_L^{(2)}$. On the other hand, $X_L^{(3)}$ is affected by the measurement results of the new X-plaquettes. Since the product of the new X stabilizers is equivalent to $X_L^{(1)} X_L^{(2)}$, then measuring the new stabilizers is equivalent to measuring $X_L^{(1)} X_L^{(2)}$. If the product of the measurement outcomes gives $+1$, we have that $X_L^{(3)} \equiv X_L^{(1)} \equiv X_L^{(2)}$, and no corrections are needed. If the outcome is -1, we have that $X_L^{(1)}$ is not equivalent to $X_L^{(2)}$: more specifically, we have $X_L^{(3)} \equiv X_L^{(1)} \equiv -X_L^{(2)}$ or $X_L^{(3)} \equiv -X_L^{(1)} \equiv X_L^{(2)}$. The correction for this discordance is to apply (in control software) a chain of Z gates on one half of the resulting lattice as if we were executing a $Z_L^{(1)}$ or $Z_L^{(2)}$. We can express the rough merge as a quantum operation M_R with Kraus operators

$$
\begin{aligned}
M_R(\rho) &= K_{0,R}\rho K_{0,R}^\dagger + K_{1,R}\rho K_{1,R}^\dagger, \text{ where} \\
K_{0,R} &= |+\rangle\langle++| \ + \ |-\rangle\langle--|, \text{ and} \\
K_{1,R} &= |+\rangle\langle-+| \ + \ |-\rangle\langle+-| \text{ or} \\
K_{1,R} &= |+\rangle\langle+-| \ + \ |-\rangle\langle-+|.
\end{aligned}
\tag{36}
$$

Where $K_{b,R}$ refers to the operation we do when we have $(-1)^b$ as the measurement outcome. We can choose between the two definitions of $K_{1,R}$ depending on which half of the resulting lattice we want to apply the corrective Z gates.

The smooth merge works similarly as the rough version. We initialize the auxiliary qubits in $|+\rangle$ and merge the lattices along their smooth boundaries, measuring Z stabilizers equivalent to a $Z_L^{(1)}Z_L^{(2)}$ measurement. This time we have that $X_L^{(3)} = X_L^{(1)} \otimes X_L^{(2)}$ and that $Z_L^{(3)} \equiv Z_L^{(1)} \equiv Z_L^{(2)}$ when the measurement is $+1$. When we measure -1, we will have a disagreement on the original Z_L operators, corrected by applying a chain of X gates to the physical qubits as if we were executing a $X_L^{(1)}$ or $X_L^{(2)}$. In terms of operators, and using the same notation as in equation (36), we define the smooth merge as

$$
\begin{aligned}
M_S(\rho) &= K_{0,S}\rho K_{0,S}^\dagger + K_{1,S}\rho K_{1,S}^\dagger, \text{ where} \\
K_{0,S} &= |0\rangle\langle 00| + |1\rangle\langle 11|, \text{ and} \\
K_{1,S} &= |0\rangle\langle 10| + |1\rangle\langle 01| \text{ or} \\
K_{1,S} &= |0\rangle\langle 01| + |1\rangle\langle 10|.
\end{aligned}
\tag{37}
$$

4.5.3 Lattice surgery CNOT

It is not immediately clear how we can perform logical operations between encoded qubits using lattice surgery. In fact, since the merge operation can also introduce irregularities in the logical operators (corrected by half a X_L or Z_L on the resulting lattice), we must study carefully how these operations can be used to perform two-qubit gates. Luckily, we can reason about lattice surgery operations using the ZX-calculus [9].

We will start by finding the equivalence in ZX-diagrams of the rough and smooth split. Looking at the definitions of the rough and smooth split in equations (35,34), we can see how each unitary is in a form very similar to the red and green spiders (respectively) of equation (7). The direct translation to ZX-diagrams of the split operations is found by creating 1-to-2 spiders:

$$U_S = \quad = |00\rangle\langle 0| + |11\rangle\langle 1|,$$

$$U_R = \quad = |++\rangle\langle +| \ + \ |--\rangle\langle -|. \tag{38}$$

In the same way, there is a straightforward translation of $K_{0,R}$ and $K_{0,S}$ from equations (36, 37) into ZX-diagrams. We recall that these Kraus operators represent the rough and smooth merge in the case when the measurements have parity $+1$ and no corrections are required. These operators are written in ZX-diagrams as 2-to-1 spiders:

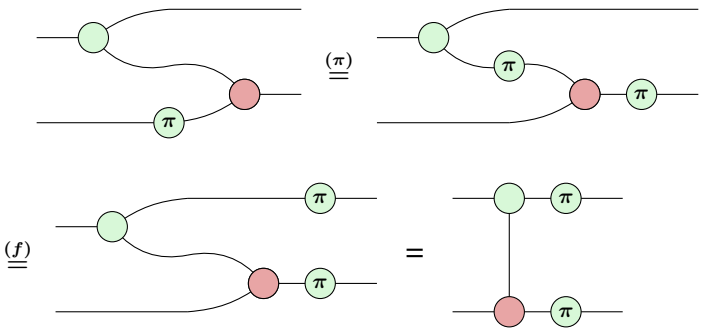
$$K_{0,S} = \quad = |0\rangle\langle 00| + |1\rangle\langle 11|,$$

$$K_{0,R} = \quad = |+\rangle\langle ++| \ + \ |-\rangle\langle --|. \tag{39}$$

As we can see, ZX-diagrams can represent lattice surgery operations naturally. Intuitively, not only do 1-to-2 spiders represent splits and 2-to-1 spiders represent merges, but we can also associate smooth operations with green spiders and rough operations with red spiders. Looking back at the definition of the CNOT in the ZX-calculus from equation (13) we can see that to compute a CNOT, we perform a smooth split on the control lattice followed by a rough merge of the target lattice with one of the control lattices from the split.

A merge with outcome -1, represented by $K_{1,R}$ and $K_{1,S}$ in equations (36, 37), can also be represented in the ZX-calculus very easily. In fact, we only need to take the representations for the $+1$ outcome of equation (39) and add a 1-to-1 π spider of the opposite color in one of the two input wires. We can think of this extra spider as the half X_L or Z_L correction that we do to make the logical operators coincide after the merge. The only difference is that, if we think of time going from left to right on the diagrams, we are then applying the correction *before* the merge, which wouldn't be possible since we will only know the outcome of the merge afterward. If we commute the π spider forward, we can then think of the extra Pauli spider as an error that causes the difference in logical operators between the two lattices; this Pauli error would, as usual, be fixed by applying another identical Pauli operation that cancels the previous (which coincides with the fact that we correct merges with a chain of Paulis that would make a logical operator on the original lattices). Being able to choose either input wire for the Pauli error is tied to the decision as to which half of the output lattice (previously an independent lattice encoding a logical qubit) we want to subject to the correction. Equation (40) provides an example lattice surgery CNOT in which we measure -1 in the

merge and decide to correct the bottom wire. Interestingly, we can effortlessly see with the ZX-calculus how this "error" propagates after the CNOT, even affecting the control lattice.



(40)

We remark that in equation (40) the wires do not represent a single physical qubit. In fact, they are an abstraction that represents the logical qubits that each lattice encodes. We will have a similar situation with the quantum circuit notation from here on.

4.6 The Hadamard, S, and T gates on the surface code

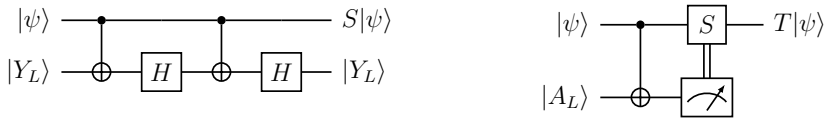
Apart from Pauli and CNOT operations, we require other quantum gates to achieve universal quantum computation. Each gate is implemented uniquely on the surface code and comes with certain limitations.

The Hadamard gate

We perform a logical Hadamard in the surface code *transversally*. A logical gate is done transversally by applying that same operation *physically* to every data qubit in the lattice. This process, apart from computing a logical Hadamard gate to the encoded qubit, changes the orientation of the lattice by 90 degrees: Z stabilizers will now be X stabilizers, and vice-versa. This means that now rough and smooth boundaries have also switched places. The surface can be rotated back to its original orientation (without operating on the logical qubit) by physically rotating the qubits or by lattice expansion and contraction [20].

The S gate

The S gate is computed by interacting our main lattice that encodes some state $|\psi\rangle$ (where we want to apply S) with an auxiliary state $|Y_L\rangle = S|+\rangle$. To create $|Y_L\rangle$, we go through a process named "state injection". First, a single physical qubit is prepared in the desired state. We then entangle this state with other physical qubits, measuring stabilizers along the process, to create a planar lattice of suitable size that encodes $|Y_L\rangle$ [20]. Once our



(a) (b)

Figure 8. Applying phase gates in the surface code. (a) An S gate is applied using an auxiliary state and four Clifford gates. (b) A T gate is teleported through a measurement that might require a corrective S gate depending on the outcome.

auxiliary qubit is ready, we can apply the S gate to our main logical qubit by performing the quantum circuit of Figure 8(a). It is important to mention that we can only prepare noisy auxiliary states that encode imperfect versions of gates. To create higher fidelity encodings, we create multiple noisy states (instead of just one) and run a process called "state distillation", which returns a $|Y_L\rangle$ of much higher fidelity [12]. Once we have our high-fidelity $|Y_L\rangle$, we can execute the circuit of Figure 8(a).

The T gate

The process for executing the T gate in the surface code resembles that of the S gate. In Figure 8(b), we see the circuit we need to perform between our primary state $|\psi\rangle$ and the auxiliary qubit $|A\rangle = T|+\rangle$. The auxiliary state, just like the one for the S gate, needs to be "injected" and "distilled" before we can use it reliably. Remarkably, the circuit for the T gate is *probabilistic*: if we measured 0 on the auxiliary qubit, then the T gate was applied correctly. If we measure 1, then we have applied $T^\dagger$, which is a $-\frac{\pi}{4}$ rotation instead of the correct $\frac{\pi}{4}$. To fix this, we need to apply an S gate. We refer to circuits with the form of Figure 8(b) as *gate teleportation* circuits.

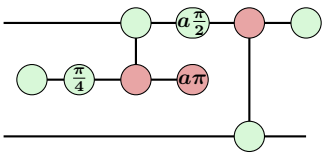
The circuit for teleporting the T gate of Figure 8(b) reveals how expensive a T gate is timewise: after performing the CNOT, we have to measure and apply an S gate on-demand depending on the outcome. This means that our computation can't continue until we have read the measurement outcome and adequately reacted to it by either doing nothing or applying the S gate, which has a considerable overhead of two CNOTS and two Hadamards. In Refs. [16, 11], a time-optimal method for teleporting non-Clifford gates that require Clifford fixups is proposed, which we are going to discuss in the next chapter.

5 Efficient gate teleportation in the surface code

In this chapter, we introduce and explain optimizations in the standard gate teleportation constructions on the surface code. We show how a delayed-choice quantum gate works, giving examples for the S and CZ gates. We then explain what a self-correcting gate is and provide the constructions for the self-correcting T and CCZ gates.

5.1 Motivation and the Pauli Frame

Part of the process of performing gate teleportation is applying corrective gates (also known as "fixup" gates) depending on the measurement result of the auxiliary qubit. These corrective gates, which need to be applied on-demand depending on a classical bit, can slow computations if they do not commute with upcoming gates. Take, for example, the following simple circuit (in ZX-calculus form) of a T gate teleportation followed by a CNOT and a measurement:



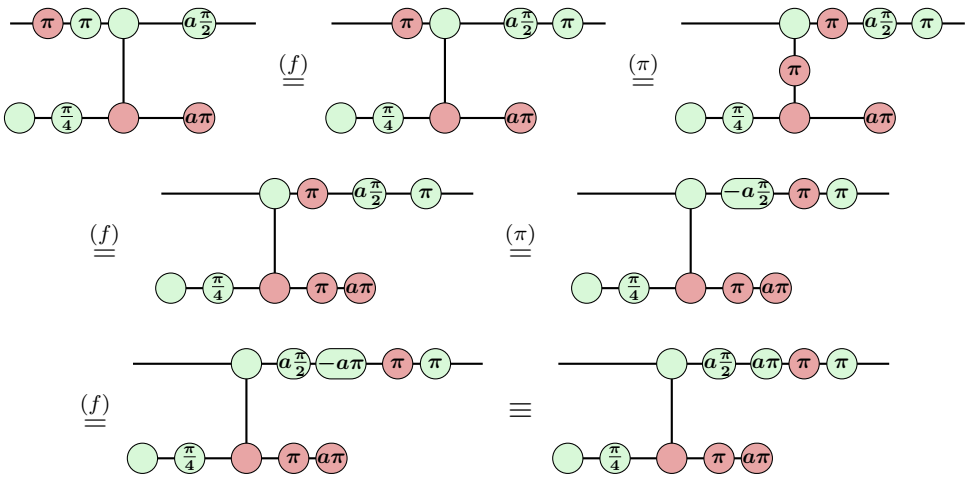
(41)

In the ZX-calculus, we represent gates controlled by classical bits using annotations with boolean variables. Being a a boolean variable, when we set $a = 0$ in equation (41), we convey that we have measured $|0\rangle$ while also making the controlled-S spider have phase 0 (and thus equal to the identity). When $a = 1$, we have measured $|1\rangle$, and the controlled-S spider will have the phase equivalent to the rotation needed to perform an S gate. We can see in (41) that the computation cannot continue until we have applied the S gate because this gate does not commute with the target of a CNOT. Additionally, since the S gate is not a π rotation, it will change the probability distribution of the X basis measurement in ways that we cannot correct later on a classical computer. Thus, the execution of the CNOT and the measurement are postponed until we measure the auxiliary qubit, read the result, and complete the S gate. In contrast, there are some gates that commute very well with most gates and change measurements in a way that can be fixed classically. We are referring to the Pauli gates, whose commutativity properties are exploited by using the Pauli Frame [27].

The Pauli Frame is a mechanism proposed by Knill [23] in which Pauli gates are (almost always) tracked in a classical computer instead of being applied in the quantum computer. Since we can't perform quantum operations with 100% fidelity, being able to skip a quantum computation by delegating the work to a classical computer will ensure a lower error rate. We can do this by keeping a *Pauli record* on every qubit, which, together with

the rest of the qubits, form a Pauli Frame [27]. A Pauli record tracks the Pauli operations that we should have applied (but didn't) to a qubit at the current point in time. We can interpret the state of a qubit $|\psi\rangle$ with Pauli record P_r as being in the effective state $P_r|\psi\rangle$. All possible values of P_r are in the set $\{I, X, Z, XZ\}$. Thus, any upcoming operation interacts with the qubit in the following way:

- If the next operation is a Pauli gate P_g, it is directly stored in the Pauli record instead of physically applied. Since the set of Pauli gates forms a group, the current entry in the Pauli record will be mapped to another Pauli gate: $P_g P_r = P'_r \in \{I, X, Z, XZ\}$ and thus $P_g P_r|\psi\rangle = P'_r|\psi\rangle$.
- If the next gate is a Clifford gate (e.g., H, S, CNOT, CZ), it can be applied without any disturbances. This is because the Clifford gates are members of the Clifford group, defined as the normalizer of the Pauli group [17]. The property of Cliffords normalizing Paulis implies that any Clifford gate can be commuted with any member of the Pauli group by mapping said element into a different member of the Pauli group. Thus, the Pauli record will be modified, while the Clifford operation C will be applied to $|\psi\rangle$ without compensations. We have $CP_r|\psi\rangle = P'_r C|\psi\rangle$, for $P'_r \in \{I, X, Z, XZ\}$.
- If the operation is a measurement, it is applied without extra quantum actions. Instead, the classical software might interpret the result differently, depending on the contents of the Pauli record. Suppose the Pauli record contains a Pauli X (respectively a Pauli Z), and the measurement was performed on the Z (resp. X) basis. In that case, the software must flip the result extracted from the measurement to get the correct reading.
- If the operation is a non-Clifford gate (e.g., T, CZ), we must apply (physically) the content of the Pauli record before performing the non-Clifford gate. This is because Pauli gates do not commute well with non-Clifford gates (for instance, X commutes with T by $TX = XT^\dagger$). In practice, we can avoid applying the content of the Pauli record if we implement the non-Clifford gate via gate teleportation. Indeed, we can see in equation (42) how commuting a Pauli X and Z through a teleported T gate only requires us to classically post-process the measurement result of the auxiliary qubit, which is flipped by the Pauli X. In the last two steps we compensate the conjugation of the S gate by adding a corrective Pauli Z (which would be added to the Pauli Frame) and use the fact that $-\pi \equiv \pi$.

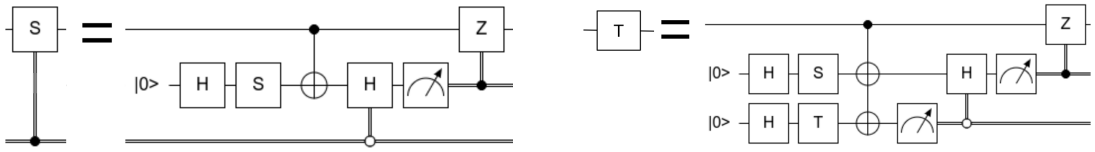


(42)

5.2 Gate teleportation with Pauli fixups

We have seen the convenience of the Pauli gates and how well they interact with all the computations necessary to have universal quantum computing. It is then natural to ask if it is possible to reduce the overhead that comes from gate teleportation in favor of more Pauli gates. Such an idea was first introduced by Fowler [11] in the form of selective destination and selective source teleportation in which, via measurements and CNOTs, it is possible to "pre-compute" Clifford fixup gates on auxiliary qubits to then transfer the gate to the main qubit in the time it takes to perform on-demand X and Z basis measurements (which introduce byproduct Paulis if the output is 1). This technique was then improved by Gidney, Fowler and Litinski [16, 24, 15], reducing the number of auxiliary qubits needed and giving specific constructions for optimizing the teleportation of the non-Clifford T and CCZ gates. In this work, we are mostly interested in the work by Gidney and Fowler [16] concerning the CCZ gate.

The gate teleportation protocol of Figure 8(b) has some interesting properties. First, the preparation of the auxiliary state (commonly referred to as the "magic state") is an offline process: it can be prepared at any point in time, even before our main computation starts, so it does not delay any other tasks. Secondly, if the measurement of the magic state gives 0, then the main qubit $|\psi\rangle$ was only required to perform a CNOT gate to get the T gate teleported into it. But, if the measurement is 1, the computation is slowed down by the need to apply the S gate onto the main qubit. We can see how a measurement outcome over which we do not have any control can mean a significant delay in the computation. The idea of a delayed-choice gate [16] is to pre-compute the fixup operation into another magic state, so we can then teleport the correction into the main qubit with just another CNOT interaction. The description of a delayed-choice S gate can be found in Figure 9(a). We are still required to perform a classically-controlled quantum gate



(a) (b)

Figure 9. Efficient gate teleportation [15]. (a) Delayed-choice S gate. (b) Self-correcting T gate. The T gate teleportation protocol, when paired with a delayed-choice S gate, leads to Pauli fixups (exclusively) on the top wire.

that, based on the measurement outcome of the T magic state, decides if the S gate is going to be teleported or not. But now, the on-demand quantum gate is a (significantly cheaper) Hadamard gate that is applied **on the auxiliary qubit** that encodes the S gate. It is crucial to understand that this Hadamard gate before a measurement changes the basis of measurement from the Z to the X basis and that this change of measurement basis determines if the fixup gate is teleported or not. In equation (43), we provide a proof using the ZX-calculus of how this change of measurement basis decides if the S gate is teleported or not.

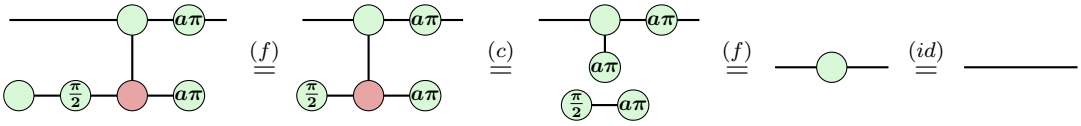
We can then combine the delayed-choice S gate with the T gate teleportation protocol to create a self-correcting T gate, illustrated in Figure 9(b). The most important difference between the self-correcting T gate and the standard T gate teleportation is how we have lifted the expensive S gate from the main qubit to an auxiliary qubit that is prepared offline. Apart from that, we now have a Pauli correction instead of an S correction on the main wire, which is never applied physically and can be commuted through any upcoming gates. The implications of this construction are the following:

- The effective time cost of applying a T gate on the main qubit is that of the multi-target CNOT. This is the consequence of having a Pauli operation as the fixup gate, which is never applied as a quantum gate.
- We need to maintain the qubit that encodes the S gate until we know the measurement result for the T magic state. Once we have retrieved the result, the classical control system will apply the Hadamard gate (or not), which will decide if the corrective S gate is teleported.
- As we saw in equation (42), if the Pauli record of the main qubit contains a Pauli X before gate teleportation, the measurement outcome of the magic state will be flipped. If we have such Pauli Frame when teleporting a self-correcting state, the Pauli will affect the decision of whether or not to apply the classically controlled Hadamard on the delayed-choice gate. This implies that we cannot decide the measurement basis of the delayed-choice gate until we are certain what is the

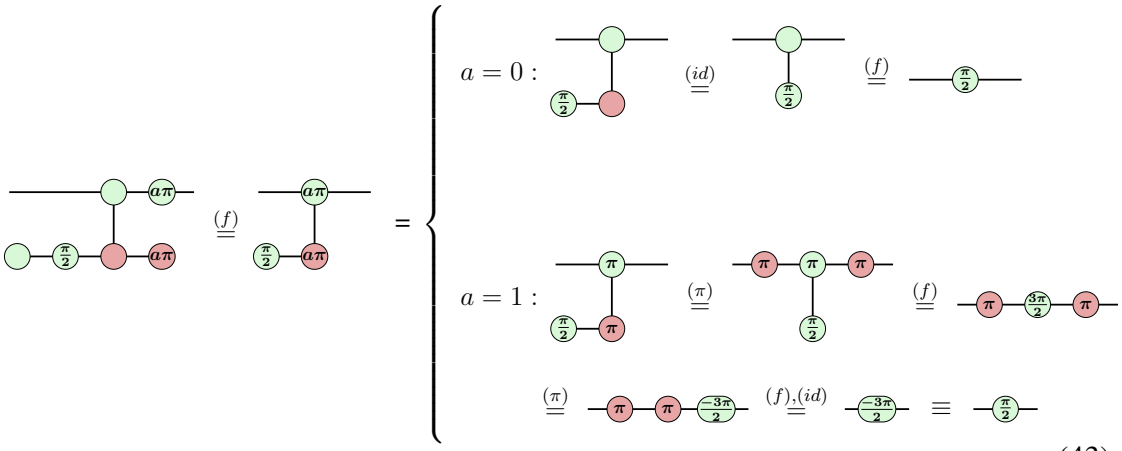
Pauli record on the main qubit. Since delayed-choice gates have Pauli fixups, then a self-correcting state might alter the Pauli Frame in a way that can flip the measurement outcome of the next gate teleportation. This implies that we cannot fully resolve a self-correcting state until the previous ones have been dealt with, with the twist that the dependencies are not in the main qubit but on the auxiliary ones. These dependencies are also resolved mainly in the control system instead of the quantum computer: once the control system determines the Pauli frame before a gate teleportation, it can decide on which basis it will measure the delayed-choice qubit, and once the measurement result for that qubit is retrieved, it can compute the Pauli frame for the next gate teleportation.

Delayed-choice S gate outcomes determined on classical bit.

Classical bit = 0. Measurement on the X basis:



Classical bit = 1. Measurement on the Z basis:



(43)

We have used single-qubit gates (T, S) in order to introduce the concepts of delayed-choice and self-correcting operations more simply. Although the T gate is very important as it can be used to achieve universal quantum computation, we are interested in another non-Clifford operation, the CCZ gate.

The CCZ gate can also be used to generate a universal gate set. In fact, the CCZ and H gates are sufficient to do so [1]. Being a non-Clifford gate, it cannot be implemented in the surface code without gate teleportation. When teleported, the required fixups consist of three CZ and three Pauli Z gates. Then, we have a similar situation as we

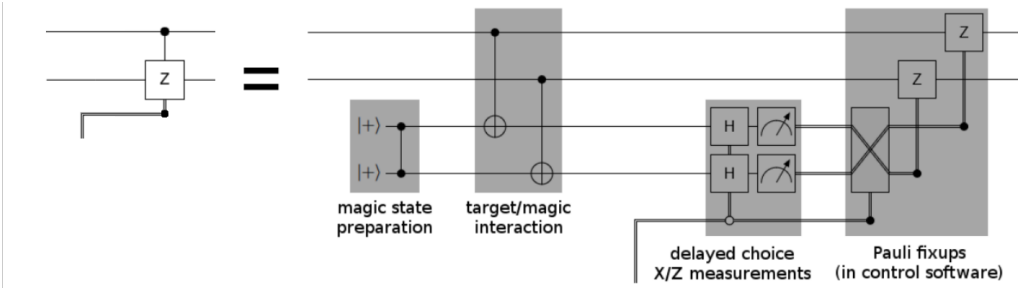


Figure 10. The delayed-choice CZ gate [16]. We encode the CZ into two auxiliary ("magic") states, and then later decide the measurement basis in order to teleport or not the CZ.

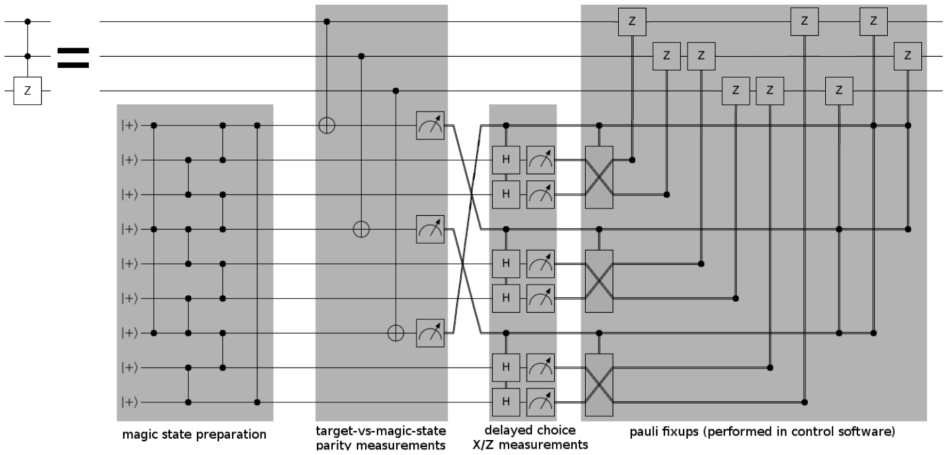


Figure 11. The self-correcting ("AutoCCZ") CCZ [16]. Three delayed-choice CZs are incorporated into a CCZ teleportation to compensate the necessary CZ corrections.

had with the T gate, where the corrections are Clifford gates present on the main qubits that significantly slow computations. We can circumvent this overhead by creating delayed-choice CZ gates and then constructing a self-correcting CCZ gate. We will now present these two constructions.

5.2.1 Delayed choice CZ gate

We can see the definition of the delayed-choice CZ in Figure 10. It works identically as the delayed-choice S, where we move the heavier computation into auxiliary qubits (this time two qubits, as we are teleporting a two-qubit gate) and let the choice of measurement basis determine whether the CZ will be teleported or not. Again, we manage to only need to perform CNOT gates on the main qubits, and the fixups are all Pauli gates.

5.2.2 Self correcting CCZ

Injecting a CCZ gate into three qubits via gate teleportation can induce up to three CZ errors and three (less important) Pauli Z errors[2] on the main qubits. To alleviate the potential overhead, we can incorporate three delayed-choice CZ gates into the gate teleportation protocol, which results in the self-correcting CCZ gate, shown in Figure 11. The self-correcting CCZ is a significantly more complex magic state whose measurement outcomes only induce byproduct Pauli Z gates on the main qubits. Similar to the self-correcting T, we have two layers of measurements. The first one is for the qubits encoding the CCZ, which can be measured at any time; this measurement teleports the CCZ into the main qubits, plus the byproduct CZ and Z gates. The second measurement layer can only be executed after we have determined the current Pauli record of the main qubits before the CNOTs, so we can decide how to interpret the first layer of measurement results. Until we know that, we need to keep the delayed-choice CZ qubits alive, but we also can continue doing computations on the main qubits.

The constructions of Figures (10, 11) were first introduced in Ref. [16], where the main focus was on how to use them for optimizing quantum ripple-carry additions and quantum table lookups instead of on their correctness. Alongside the quantum circuit representations we have shown, the original source also gives ZX-calculus and topological spacetime diagram representations for the delayed-choice CZ and self-correcting CCZ. The proof of correctness of these constructions is missing in the original source and never pointed out in related works. The main result of this thesis is to prove that these constructions are correct, which we do in the next chapter.

[2]We provide proof of this fact using the ZX-calculus in the next chapter.

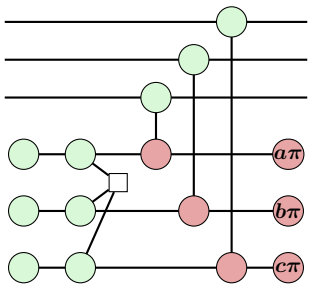
6 Verification of efficient gate teleportation protocols

In this chapter, we will present the main output of this work. We begin by demonstrating with the ZX-calculus that teleporting a CCZ magic state can introduce three byproduct CZ and Z gates into the main qubits. We then prove the correctness of the delayed-choice CZ gate for the two values of the classical control bit and all possible measurement outcomes. Finally, we prove the correctness of the self-correcting CCZ for all possible outcomes of the two measurement layers.

6.1 Byproduct operations of a CCZ teleportation

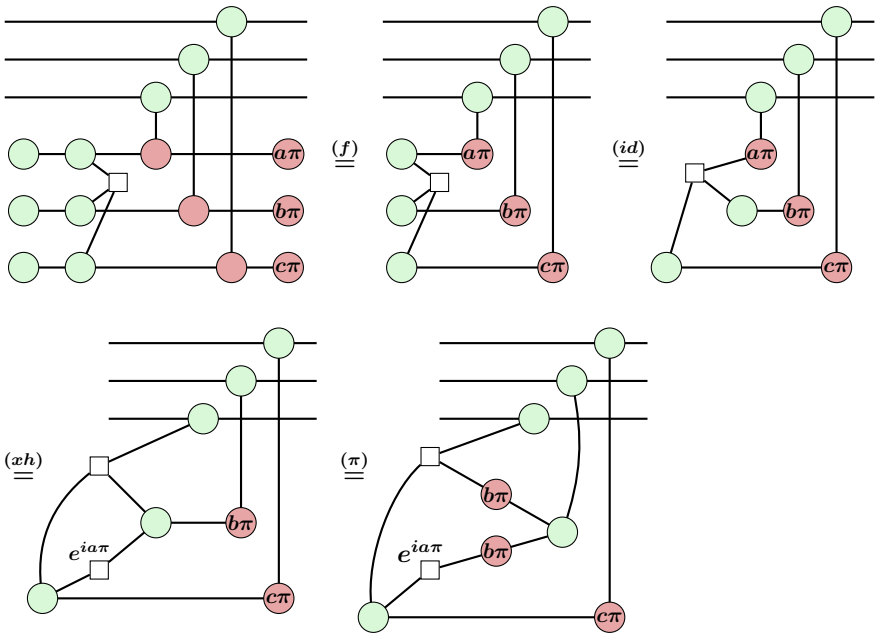
Before demonstrating the correctness of the delayed-choice CZ and AutoCCZ states, it is interesting to develop the intuition of how the injection of a CCZ can generate the byproduct operations claimed by Ref. [16]. Since there is no published formal demonstration (to the best of our knowledge) that the CCZ teleportation behaves in that way, we are now going to verify it with the ZX-calculus.

To inject a CCZ into the desired (main) qubits, we first create the auxiliary state $|CCZ\rangle = CCZ|{+}{+}{+}\rangle$. We then interact the main qubits with the auxiliary qubits via CNOTs, and then we measure all the auxiliary qubits on the Z basis. We represent this process with the ZX-diagram of equation (44).



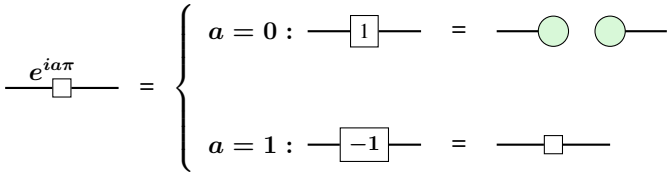
(44)

We are now, using the rules of Figure 2, going to transform the diagram to find the byproduct operators. We do so by applying ZX-rules until we arrive at a diagram that we can understand as only operations on the main qubits. We remark that, instead of evaluating the diagram for all eight possible values of a, b, c, we showcase a way of transforming the diagram into the desired one while keeping the conditional variables, allowing for additional expressiveness in the end result. By applying a series of rules, we arrive at:



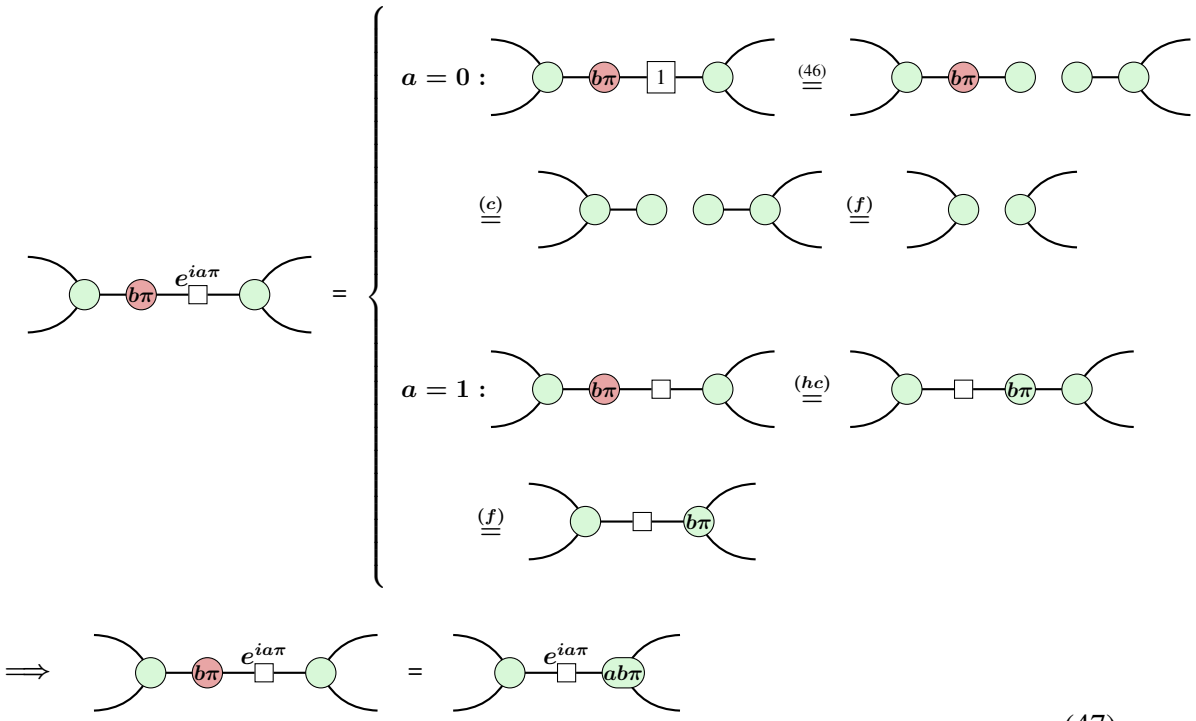
(45)

Where we encounter a problem when needing to commute the conditional X-spider with phase $\alpha = b\pi$ into the conditional H-box with phase $\beta = e^{ia\pi}$. To deal with this, we first inspect what does the H-box represent for each value of a:



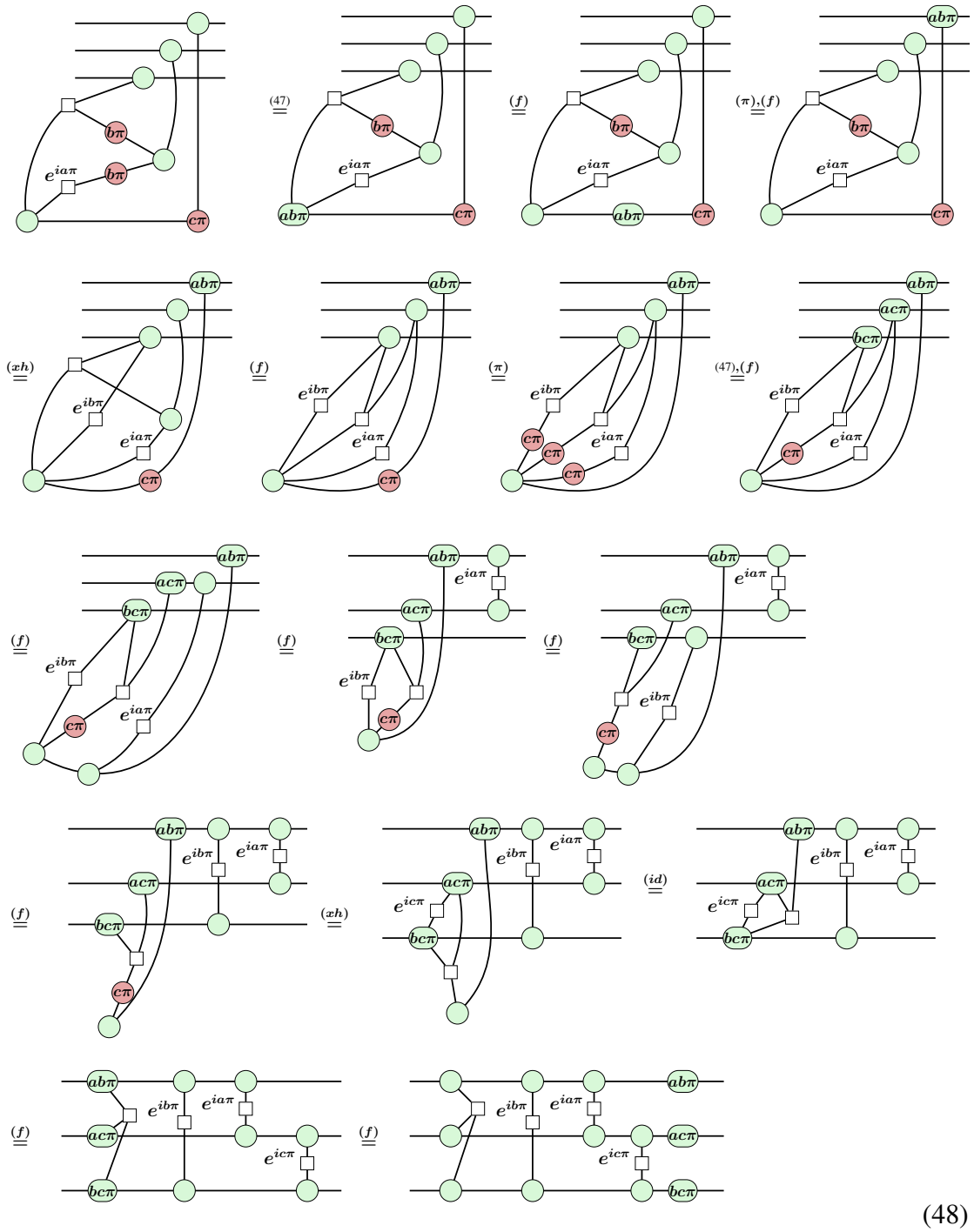
(46)

For a proof of the last equality for the case $a = 0$, we refer the reader to [30, Section 8.2]. For $a = 1$, the result is the Hadamard gate. With equation (46) in mind, we can create a specific rule for our situation, in which we manage to define how the X spider can be commuted through the conditional H-box:



(47)

We can then continue our simplification with our newfound rule. The final derivation can be found in equation (48). The resulting diagram can indeed be read as a quantum circuit with a CCZ gate, plus byproduct CZ and Z gates depending on the measurement outcomes. We remark that the conditional CZ gates on the last diagram are reduced to the identity when the boolean variable is 0 and to a CZ when the boolean variable is 1; this follows from equation (46). The Z gates will only appear if the two measurement outcomes in their phase give 1. Naturally, both CZ and Z errors are corrected by applying that same error again since the CZ and Z gates are self-inverse. It then makes sense to embed three delayed-choice CZ into the CCZ teleportation, plus three extra Pauli Z tracked in control software. Hence, in principle, the idea of the AutoCCZ is based on the correct foundation that the possible byproduct operators are three CZ and three Z gates. We can then proceed to verify if the *implementation* of both the delayed-choice CZ and the AutoCCZ are correct.

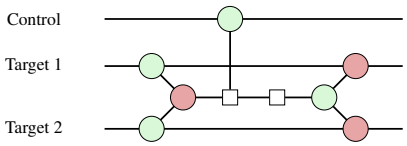


(48)

6.2 Correctness of the delayed choice CZ

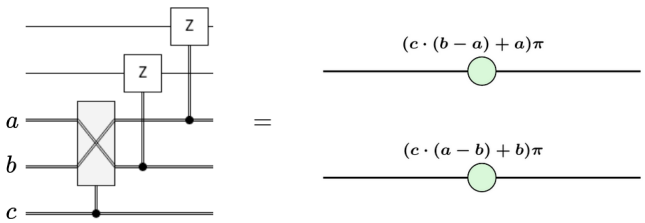
We are now going to verify if the delayed-choice CZ implementation of Figure 10 implements a CZ when the control is 1 and applies the identity when the control is 0. Apart from that, we must ensure that we exclusively generate byproduct Pauli gates if we measure 1 on the auxiliary qubits.

To verify the delayed-choice CZ, we must first understand how to write a classically controlled Hadamard and a classically controlled SWAP in the ZX-calculus. We begin by explaining the SWAP, which controlled on a quantum bit is represented as follows [30, Section 8.3]:



(49)

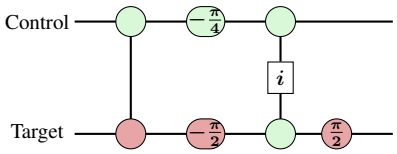
The construction of a controlled SWAP is, as we can see, complicated and not visually intuitive. Conveniently for us, since our gates are controlled by classical wires and we intend to swap classical bits, we can easily represent the controlled SWAP by just using annotations, without the need to represent any classical wire[3]. Let c be a classical bit that controls a SWAP operation that interchanges two classical bits a, b that in turn serve as control bits for Pauli Z gates on two different wires; this operation is represented in ZX-calculus form as:



(50)

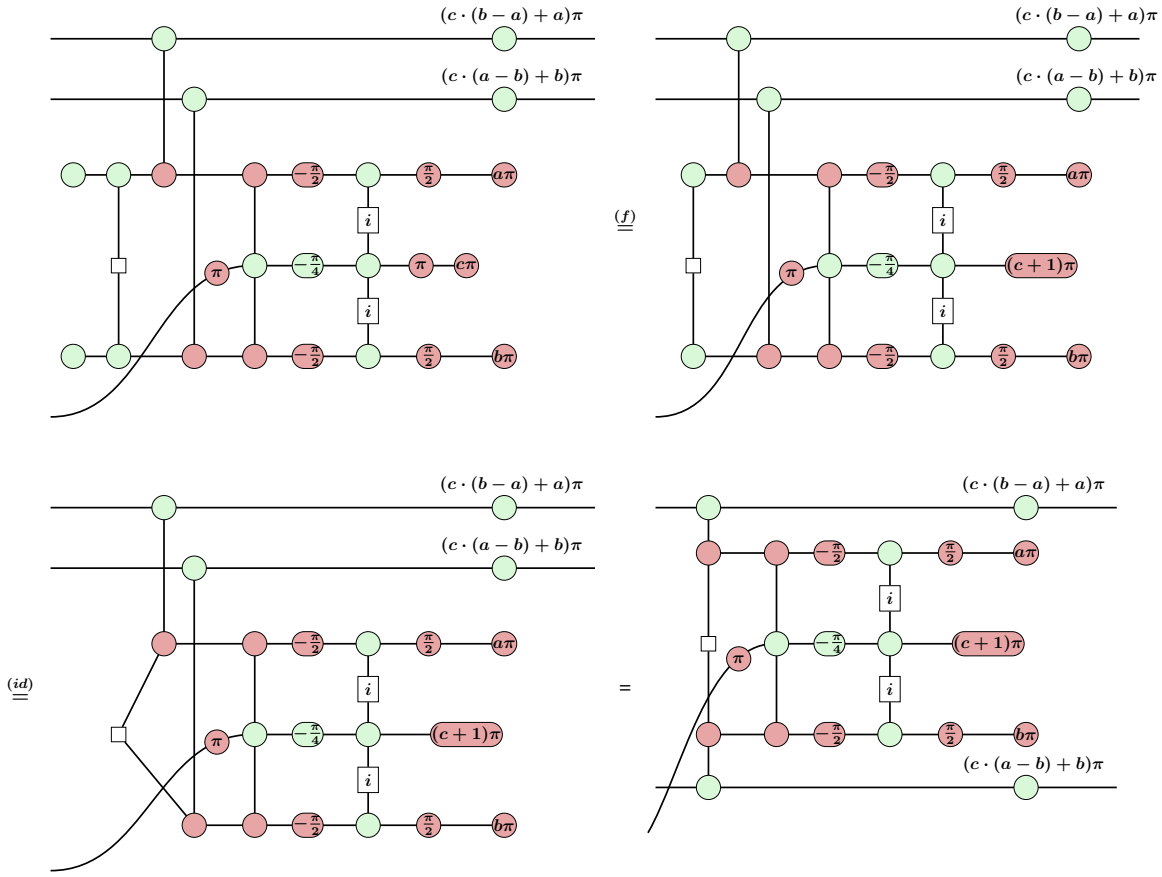
In equation (50) we can easily see how the phase of the Z spiders change between $a\pi$ and $b\pi$ depending on the value of the control bit c. A controlled Hadamard on a quantum bit is represented in the ZX-calculus as [30, Section 8.3]:

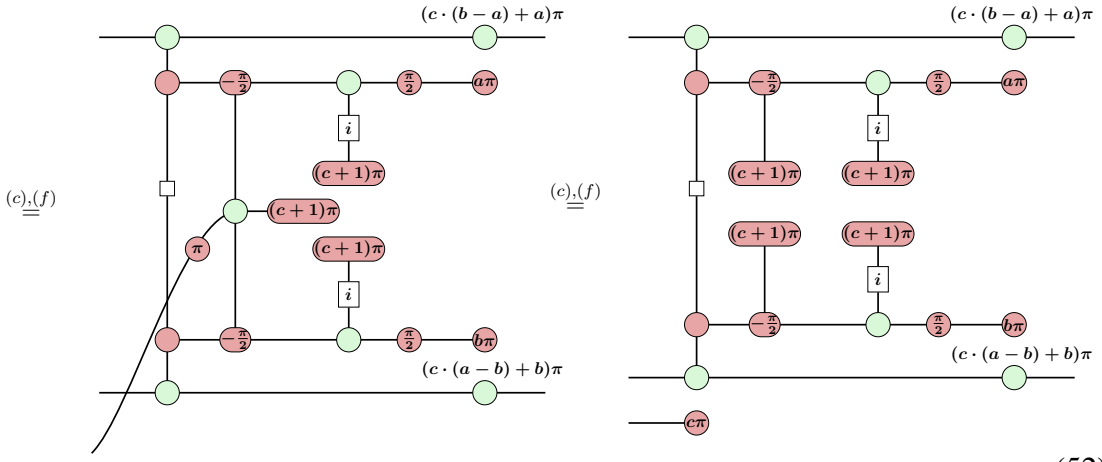
[3]There are ways to represent classical information in the ZX-calculus using thin/thick wires to denote classical/quantum information [30, Section 10.2]. Although this extension of the language can be helpful for visual intuitiveness, we have decided in favor of a more compact representation that relies on annotations.



(51)

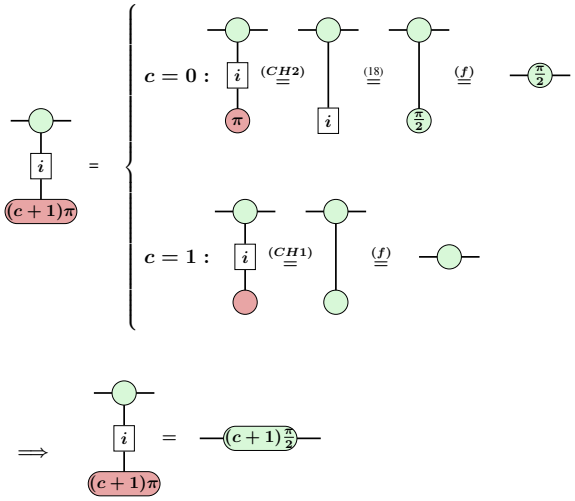
The Hadamard gates from Figure 10 are anti-controlled on a classical bit. We can derive how to represent this by doing the following. First, we say that the Hadamards are anti-controlled by the value of a classical bit c. Then, we control the two Hadamards on a quantum wire, which we will set as being measured on the computational basis with outcome c. To represent an anti-control on c, we flip the control wire before and after using it for the controlled Hadamards using X gates. In equation (52), we define and simplify the delayed-choice CZ of Figure 10 with the ZX-calculus using the described representations for the controlled SWAP and Hadamards.





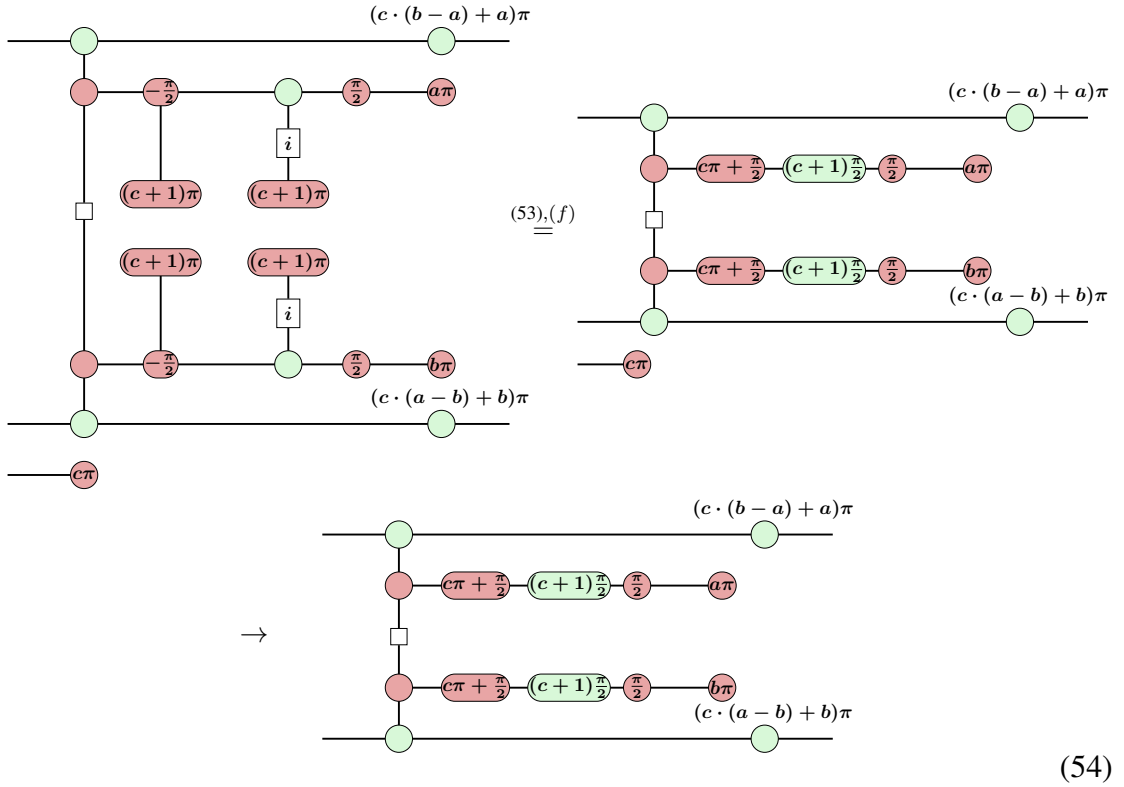
(52)

To further simplify the diagram of equation (52), we introduce the following custom rule that describes how the H-box of phase i interacts with the arity-1 X-spider of phases 0 or π:



(53)

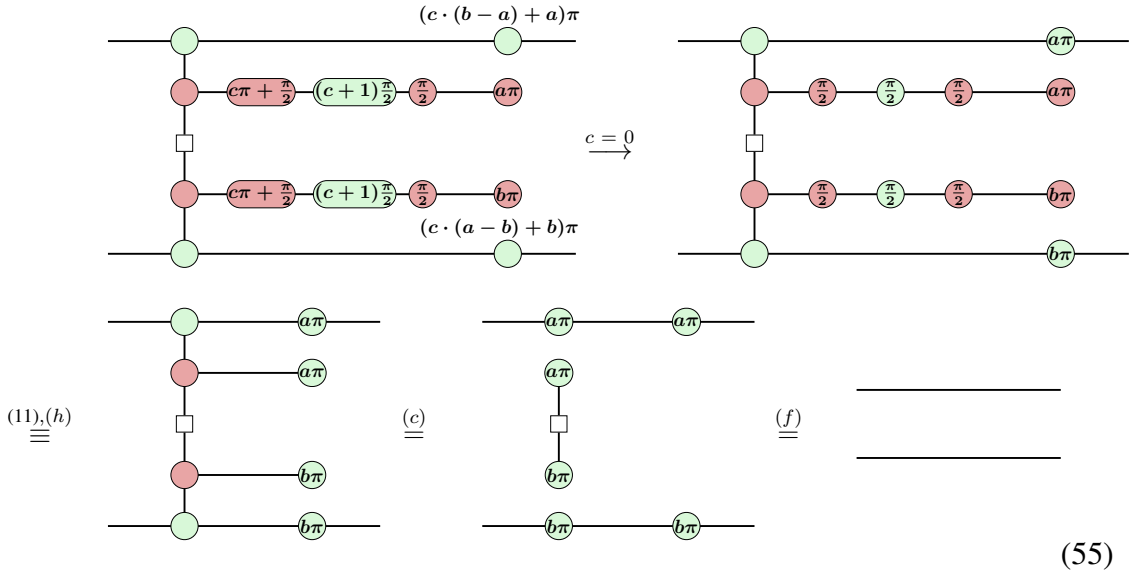
We can then finish the simplification of the delayed-choice CZ, shown in equation (54). In the last step, we remove the measurement of the qubit that is used as a control. We do this to have a more general delayed-choice CZ, in which the source of the control bit c is abstracted away. This lets us take the diagram of equation (54) and use it in any other larger diagram that requires it, in which the source of the c bit can be anything as long as it represents a classical bit. We remark that the $X - Z - X$ rotation spiders placed before the measurements represent a Hadamard gate using the decomposed version of equation (11), but with an added control variable c that will reduce it to the identity when $c = 1$ and leave the correct phases when $c = 0$; that is exactly a *classically anti-controlled* Hadamard gate.



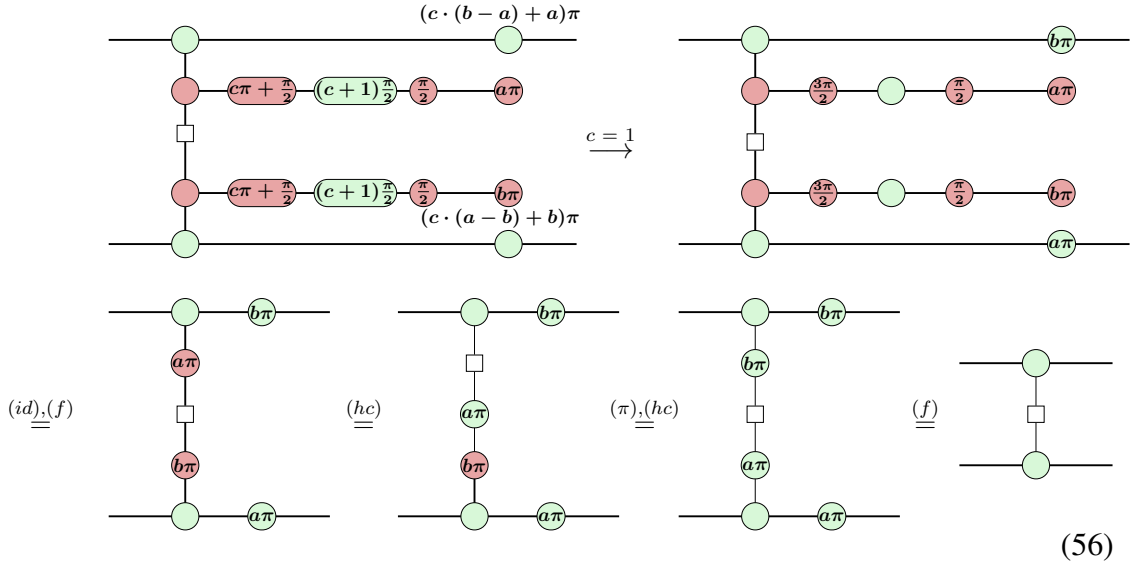
(54)

Reference [16] also provides a representation in ZX-calculus form of the delayed-choice CZ, shown in Figure 12(a). We can see how the original representation differs from our result. First, in notation, the original diagram uses white and black nodes to represent X and Z spiders. Secondly, custom red nodes represent a measurement on either the Z or X basis, as opposed to our representation that uses Hadamards controlled on a classical bit. We have chosen our representation over the original because later, we will have three delayed-choice CZs in the same diagram. This will generate six different conditional measurements that we would benefit from annotating so that we know which classical bit controls each delayed-choice CZ. Lastly, we can also see that the original diagram does not represent the Pauli Z fixups for reasons we discuss later.

We shall now verify if the implementation of the delayed-choice CZ of Ref. [16] works as intended. The main difficulty is to prove it for all possible outcomes of the two measurements that we have and for the two values of the control c. We will see now that, instead of taking every possible combination of a, b, and c, we only need to give values to c while the a and b annotations can be removed using ZX-rules. In equation (55) we can see that, when we set $c = 0$, the CZ gate is not teleported into the main qubits. In the same manner, we verify in equation (56) that the CZ gate is teleported when $c = 1$.



(55)



(56)

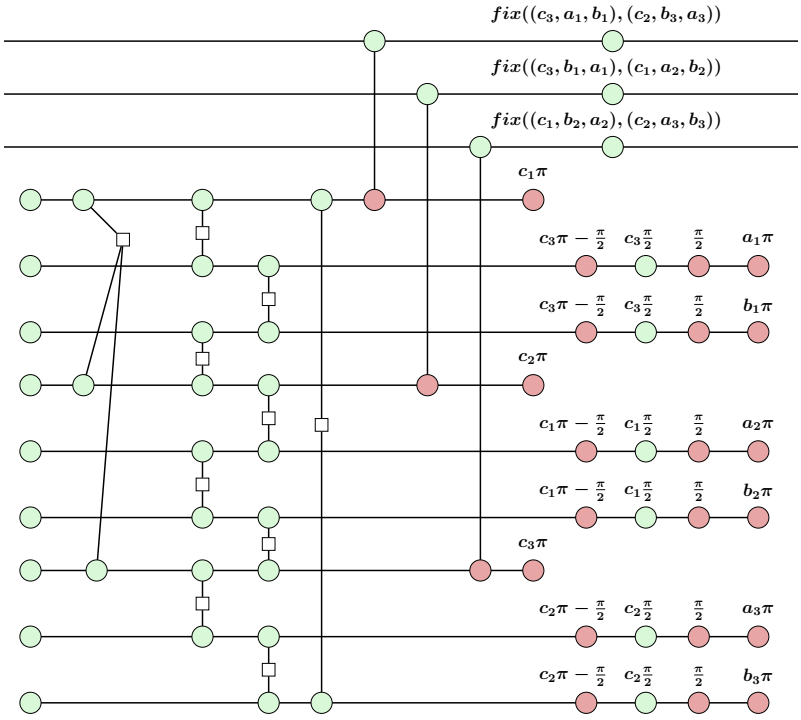
Hence, we have formally proved the correctness of the delayed-choice CZ implementation from Ref. [16] (that is, we have checked that a CZ is teleported with Pauli fixups when $c = 1$). This construction is a crucial part of the AutoCCZ state. Now that we have explained all the concepts necessary to understand the AutoCCZ and proven one of its main components, we are prepared to verify its correctness formally.

6.3 Correctness of the AutoCCZ

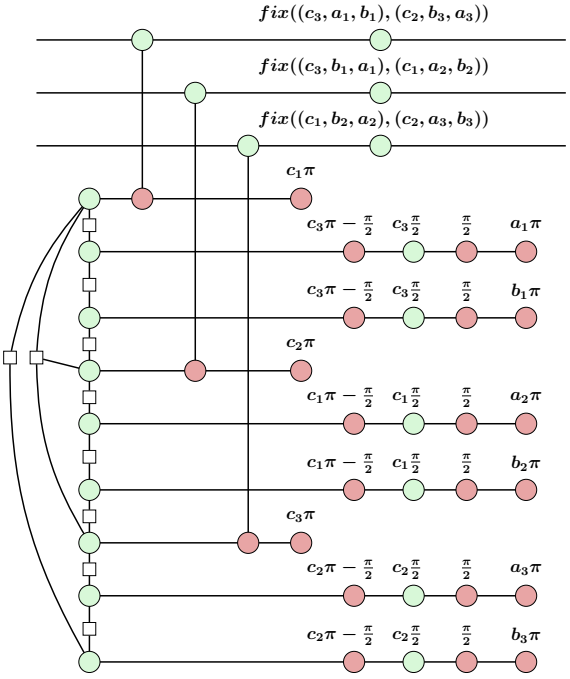
We will now show if the construction of Figure 11 successfully encodes a CCZ gate on the main qubits for every possible measurement outcome. Inspecting the circuit, we can find three delayed-choice CZ gates embedded in the $|CCZ\rangle$ state. Each one has its choice of measurement basis controlled by the measurement outcome of one of the $|CCZ\rangle$ qubits. Apart from that, we can see that the last three Pauli fixups are not part of the Pauli fixups required by the delayed-choice CZ gates, but are the extra necessary to negate the byproduct Z gates that we derived in equation (48).

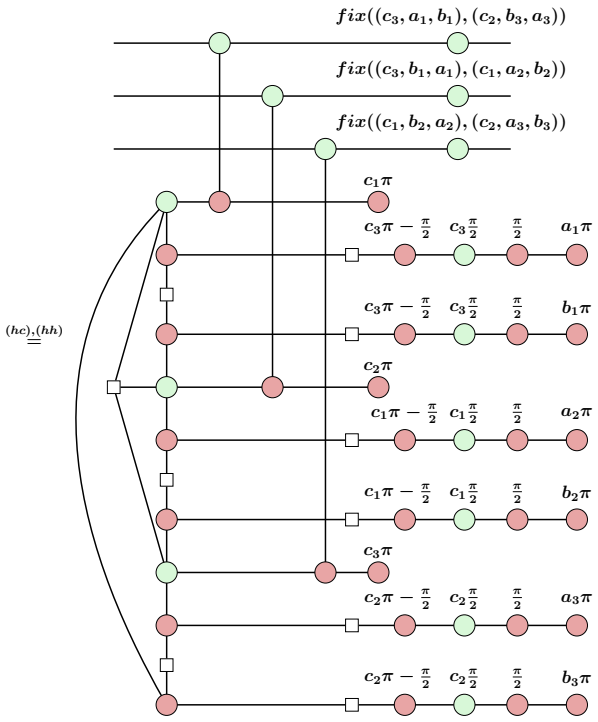
Before going into the proof, it is convenient first to take the circuit of Figure 11, transform it into ZX-calculus form, and make it as compact as possible without giving values to the annotations. This process can be found in equation (57). To account for the single-qubit Z errors that come from the delayed-choice CZ and $|CCZ\rangle$ teleportation, we have annotated a Z spider in each main wire with the function $fix((c_i, a, b), (c_j, e, f)) = (c_i \cdot (b - a) + a + c_j \cdot (f - e) + e + c_i \cdot c_j)\pi$. Each input three-tuple of the function accounts for the possible Z errors that come from a delayed-choice CZ. The function, when added to a Z-spider on each main wire, aggregates the possible Pauli errors of the two CZs that qubit might receive, plus the extra Pauli that is conditioned on $c_i \cdot c_j$ that might come from the $|CCZ\rangle$ teleportation.

In the second to last step of the simplification, we use a "conditional measurement" gadget that we define in equation (58). This gadget represents a measurement in a choice of basis that is conditional on one of the annotations (c_i), with measurement result a_i or b_i. We can think of it as a (classically) anti-controlled Hadamard before a measurement.

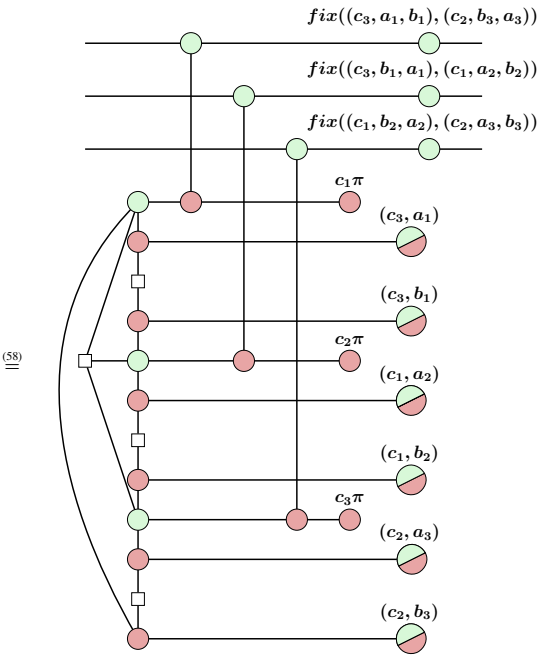


$\underline{\underline{(f)}}$

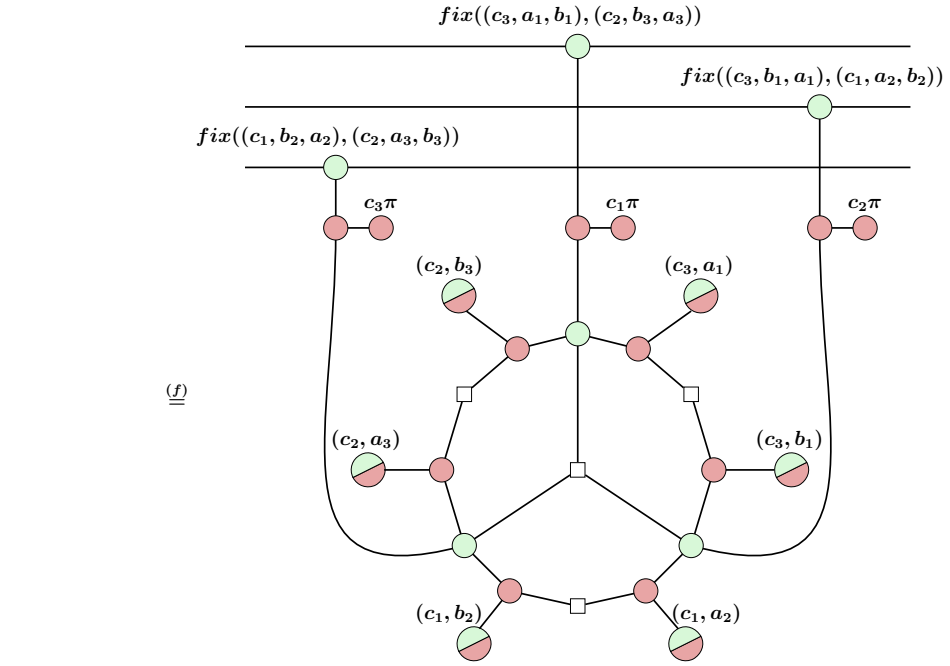




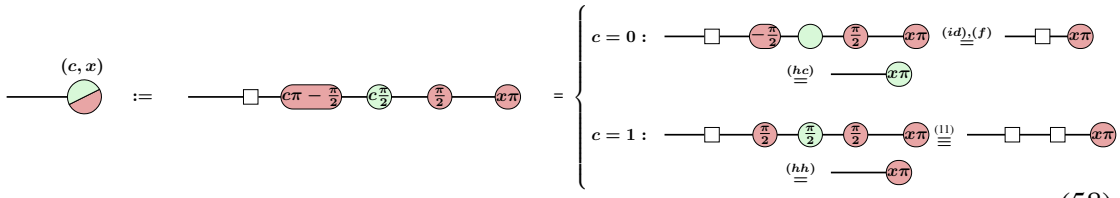
$\overset{(hc),(hh)}{=}$



$\overset{(58)}{=}$



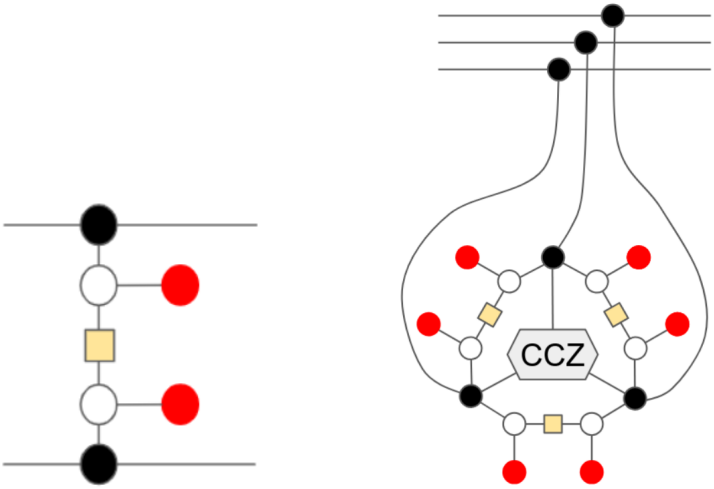
(57)



(58)

In Ref. [16, Figure 4], a ZX-calculus representation of the AutoCCZ is also given, which we show in Figure 12(b). We will now indicate the differences between this construction and our result of equation (57):

- The most notable difference is how the original figure does not use annotated ZX-diagrams; we cannot express the different possible measurement outcomes we can get and how they affect the overall computation. We can interpret this as the figure providing a representation of the case where we measure 0 on every qubit. Interestingly, this results in the removal of the spiders that represent the first measurement layer using the ZX identity rule. In our diagram, we can find these measurements in the longer vertical wires that connect the main qubits with the AutoCCZ, annotated with c_1, c_2, and c_3.
- Similar to the delayed-choice CZ, the original diagram does not include the Pauli fixups on the main qubits. This is, to our understanding, because these operations are never applied to the qubits but tracked in software, so we can think of the



(a) (b)

Figure 12. Original ZX-calculus constructions of the delayed-choice CZ in Figure (a) and AutoCCZ in Figure (b).

original diagram as a representation of the operations present on the quantum computer itself. We have chosen to add these Pauli operations to the diagram as annotated Z spiders, which we aggregate together in each wire to simplify the diagram. We choose to add the Pauli corrections so that when we prove the correctness of the AutoCCZ, we can demonstrate how these Pauli fixups indeed remove the byproduct operations. An alternative to this approach would be not to write the fixups and verify that the end diagram is a CCZ plus Pauli errors that should coincide with the fixup part of Figure 11.

- The other notable difference appears in the conditional measurement gadgets. Our result creates gadgets that depend on the measurement outcomes of the $|CCZ\rangle$ qubits, whereas the ones from the original diagram could be seen more as indications that three delayed-choice CZ states need to be handled if we want the corrections to be teleported. In our representation, we can also identify which conditional measurement pair create a delayed-choice CZ by recognizing which gadgets share the control annotation c_i.

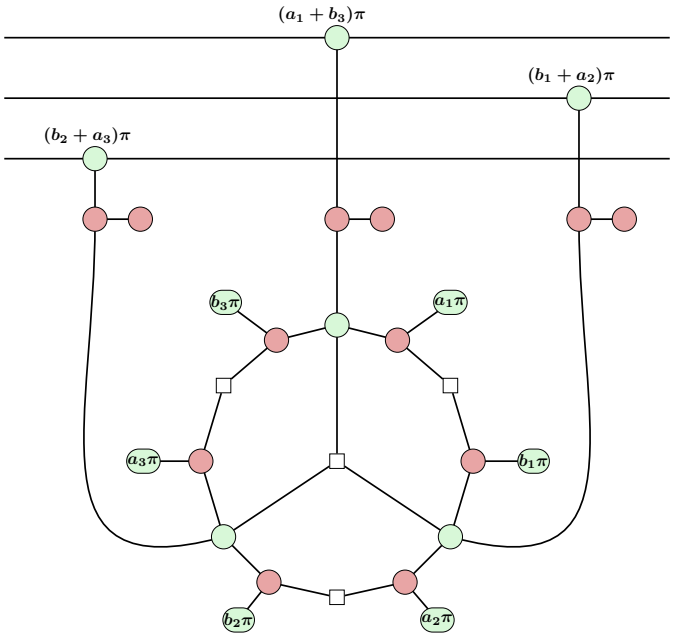
We will now verify the correctness of the AutoCCZ construction. This requires us to verify every possible combination of both the control measurements (c_1, c_2, c_3) and the delayed measurements $(a_i, b_i, i \in \{1, 2, 3\})$. Conveniently for us, the latter measurements will never need to be given a specific value, as we will absorb the conditional phases using ZX rules. On the contrary, the control measurements require us to take specific values. We do not need to evaluate all eight possible cases; instead, it suffices to prove

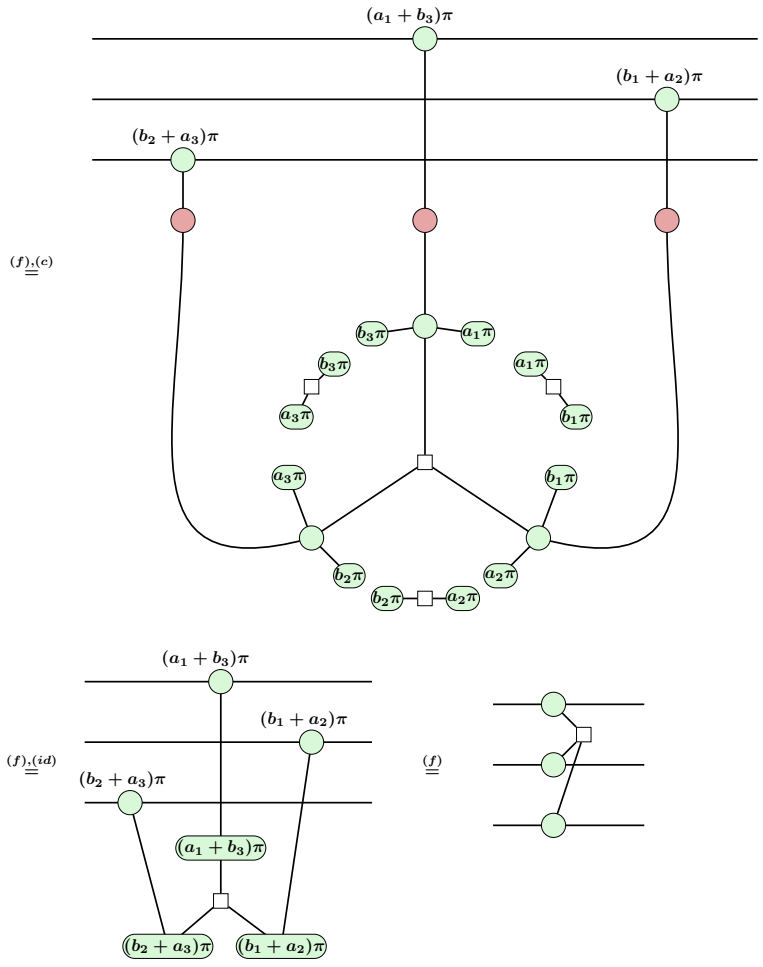
correctness for some cases and show equivalence between the proven and the remaining cases.

Lemma 1. *The self-correcting CCZ is equivalent to a CCZ without byproduct operators for the measurement outcomes* $c_1 = c_2 = c_3 = 0$, $c_1 = 1, c_2 = c_3 = 0$, $c_1 = c_2 = 1, c_3 = 0$, *and* $c_1 = c_2 = c_3 = 1$.

Proof. We begin by evaluating the diagram of equation (57) for $c_1 = c_2 = c_3 = 0$. We do so by writing 0 where the c_i annotations appear. This process also evaluates the conditional measurement gadget, giving it a specific spider type. We find the result in equation (59), where we can verify that the final diagram is indeed equivalent to a CCZ gate.

Case 1. $c_1 = c_2 = c_3 = 0$

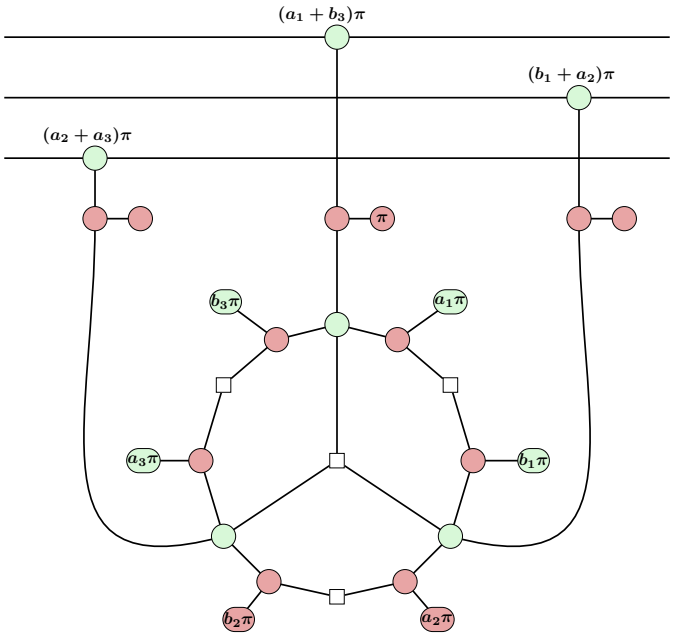


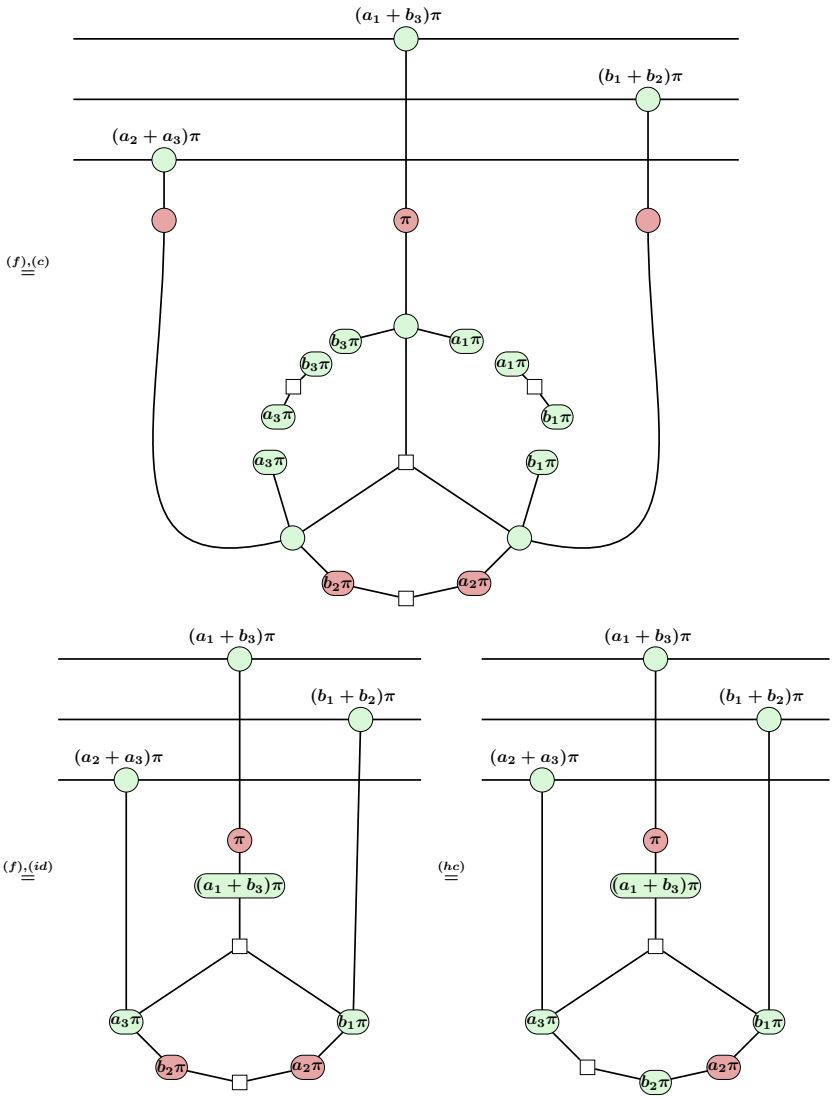

$(a_1 + b_3)\pi$
$(b_1 + a_2)\pi$
$(b_2 + a_3)\pi$
$(f),(c)$
$b_3\pi$
$a_1\pi$
$b_3\pi$
$a_1\pi$
$a_3\pi$
$b_1\pi$
$a_3\pi$
$b_1\pi$
$b_2\pi$
$a_2\pi$
$b_2\pi$
$a_2\pi$
$(a_1 + b_3)\pi$
$(b_1 + a_2)\pi$
$(f),(id)$
$(b_2 + a_3)\pi$
$(a_1 + b_3)\pi$
$(b_2 + a_3)\pi$
$(b_1 + a_2)\pi$
(f)

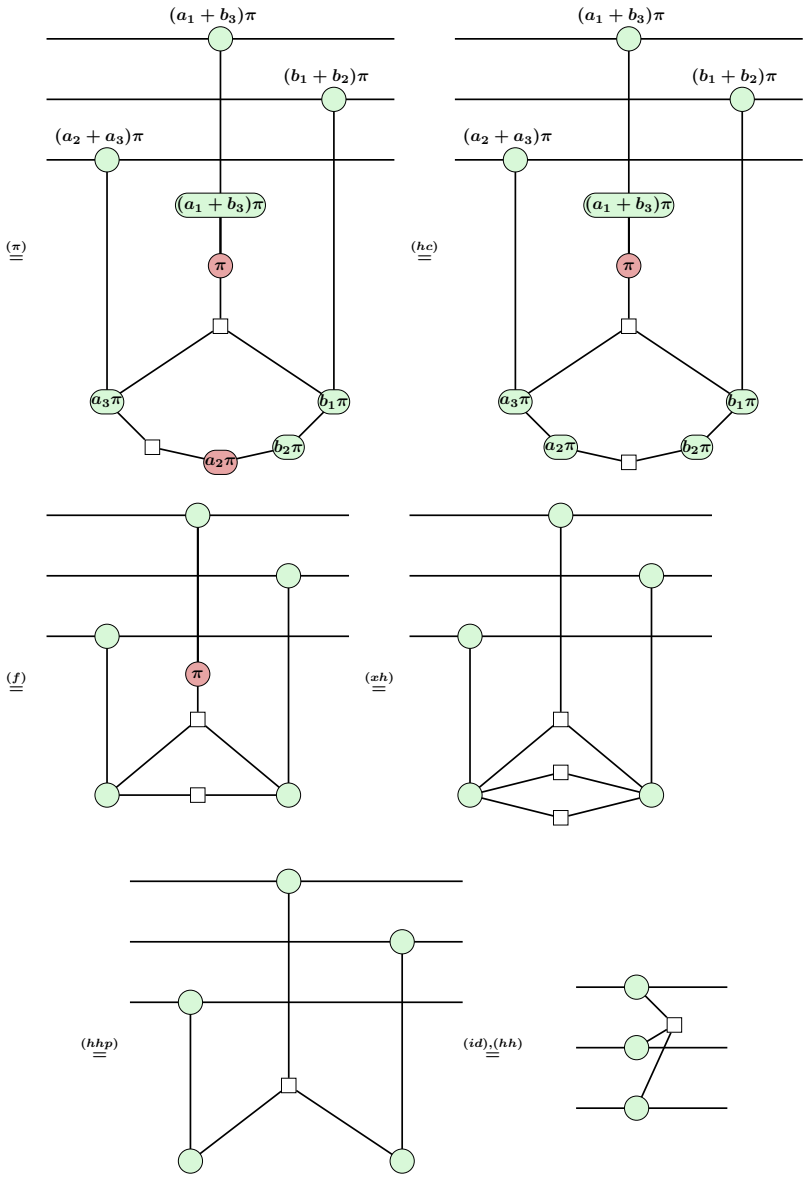
(59)

The verification for the case $c_1 = 1, c_2 = c_3 = 0$, $c_1 = c_2 = 1, c_3 = 0$, and $c_1 = c_2 = c_3 = 1$ can be found in equations (60), (61), (62). We can see how, even though we saw in equation (48) that we get byproduct CZ gates when measuring 1 on any of the $|CCZ\rangle$ qubits, the AutoCCZ successfully compensates for this by using the delayed-choice CZ states. We can also see that, without the need to test for specific values of a_i and b_i, the byproduct Z gates are also removed.

Case 2. $c_1 = 1, c_2 = c_3 = 0$

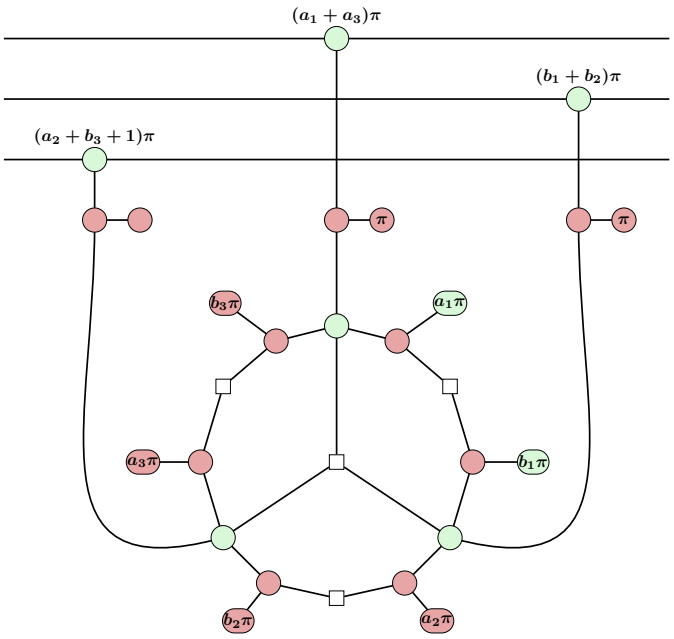



$(a_1 + b_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + a_3)\pi$
π
$(f),(c)$
$b_3\pi$
$a_1\pi$
$b_3\pi$
$a_1\pi$
$a_3\pi$
$b_1\pi$
$a_3\pi$
$b_1\pi$
$b_2\pi$
$a_2\pi$
$(a_1 + b_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + a_3)\pi$
π
$(f),(id)$
$(a_1 + b_3)\pi$
$a_3\pi$
$b_1\pi$
$b_2\pi$
$a_2\pi$
$(a_1 + b_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + a_3)\pi$
π
(hc)
$(a_1 + b_3)\pi$
$a_3\pi$
$b_1\pi$
$a_2\pi$
$b_2\pi$

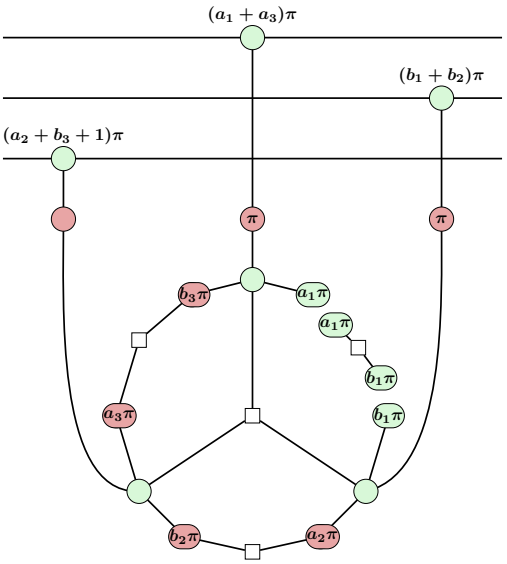

$(a_1 + b_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + a_3)\pi$
$(a_1 + b_3)\pi$
π
$a_3\pi$
$b_1\pi$
$a_2\pi$
$b_2\pi$
(π)
(hc)
(f)
(xh)
(hhp)
$(id)_{\cdot}(hh)$

(60)

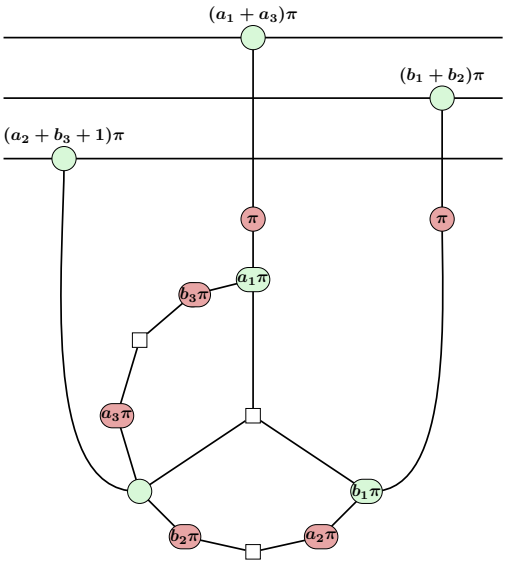
Case 3. $c_1 = c_2 = 1, c_3 = 0$

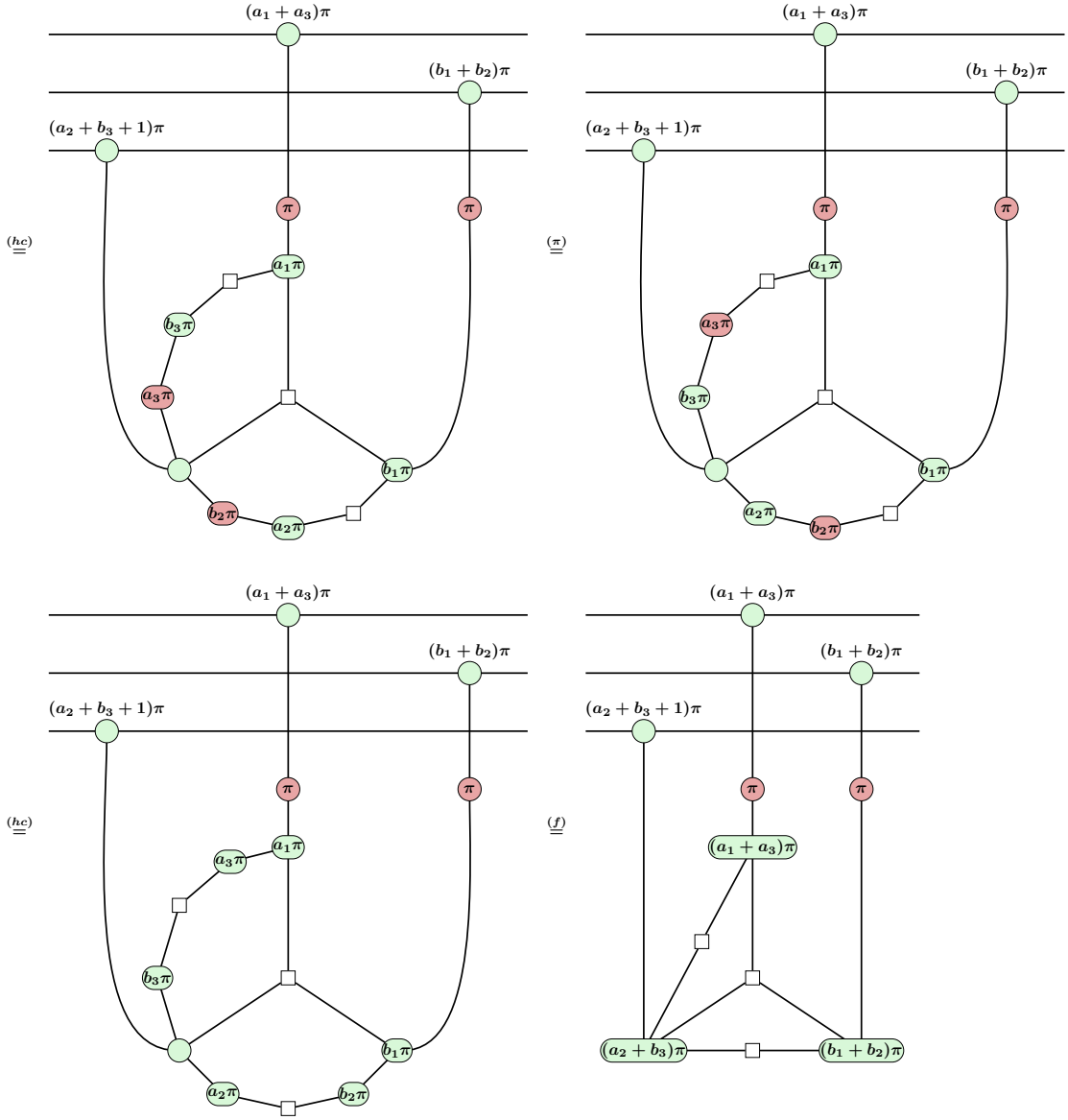


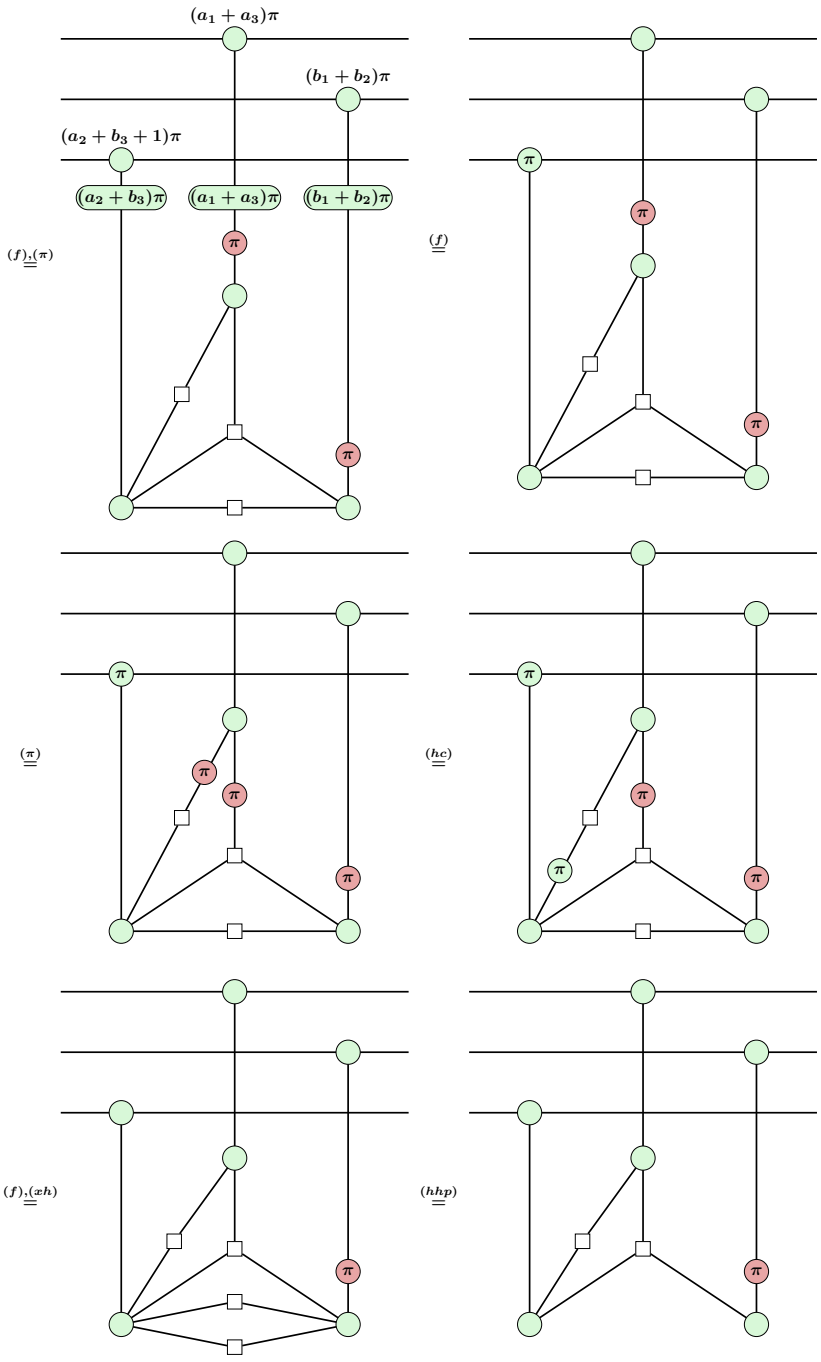
$\overset{(f),(c)}{=}$

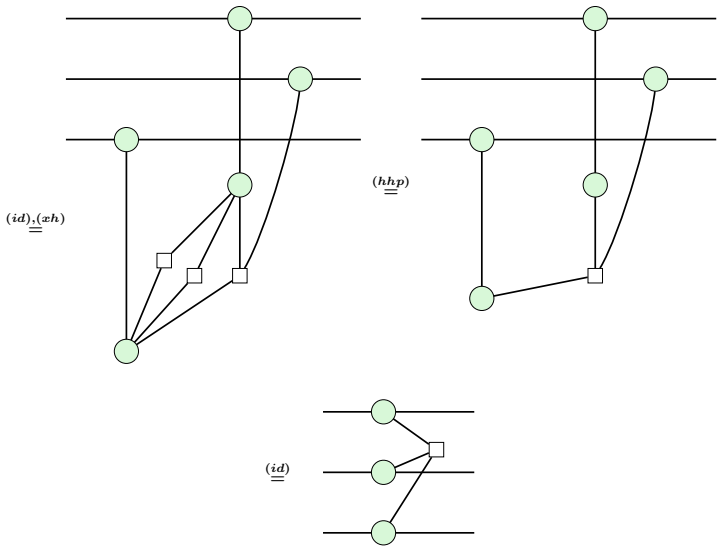


$\overset{(f),(id)}{=}$



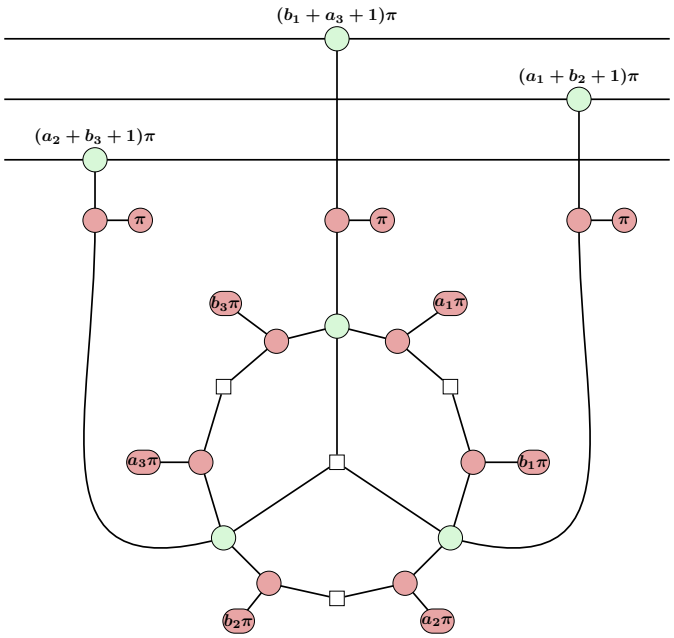

$(a_1 + a_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + b_3 + 1)\pi$
π
π
$a_1\pi$
$b_3\pi$
$a_3\pi$
$b_1\pi$
$b_2\pi$
$a_2\pi$
(hc)
(π)
(hc)
(f)
$(a_1 + a_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + b_3)\pi$

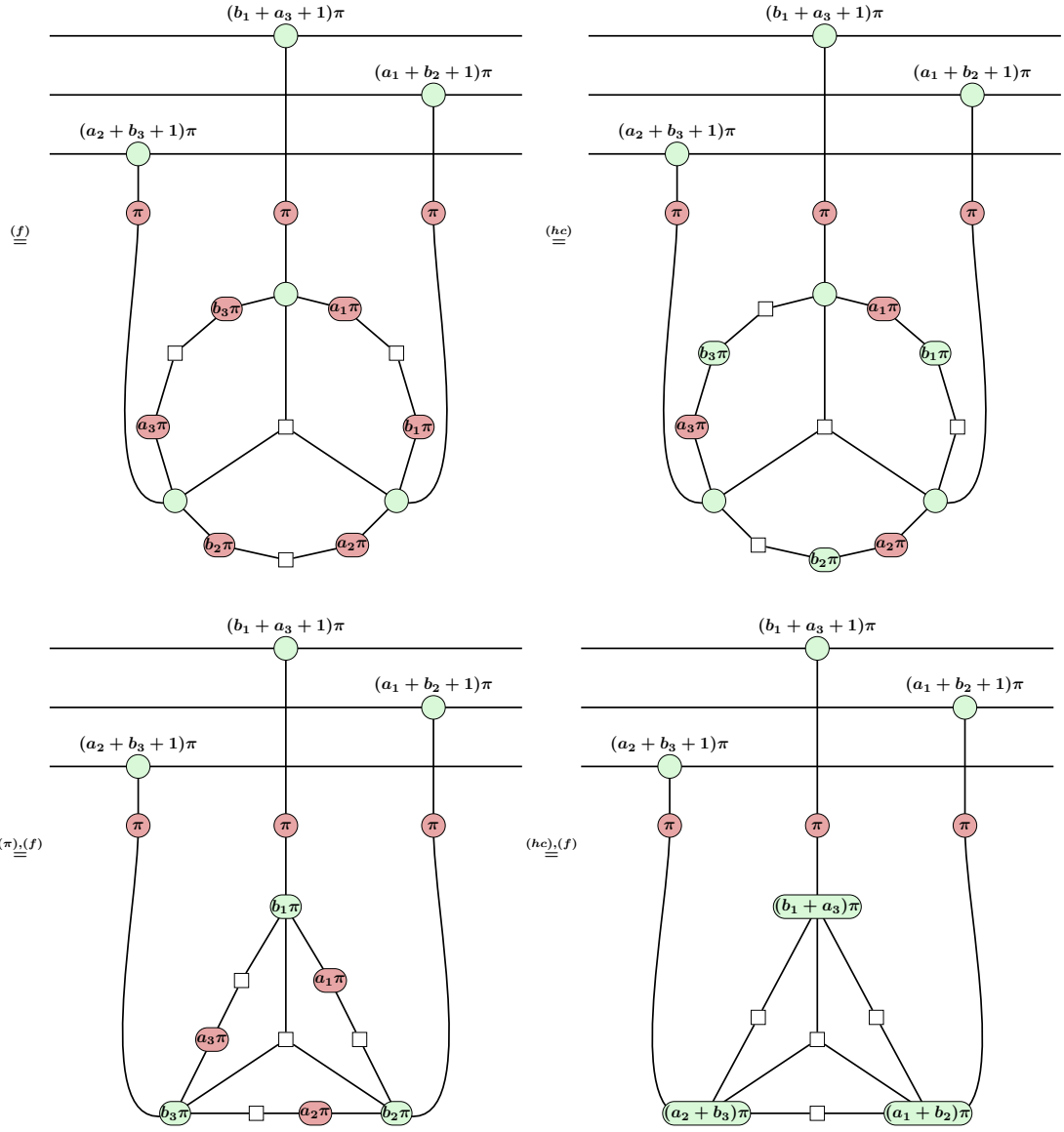

$(a_1 + a_3)\pi$
$(b_1 + b_2)\pi$
$(a_2 + b_3 + 1)\pi$
$(a_2 + b_3)\pi$
$(a_1 + a_3)\pi$
$(b_1 + b_2)\pi$
π
$(f),(\pi)$
(f)
(π)
(hc)
$(f),(xh)$
(hhp)

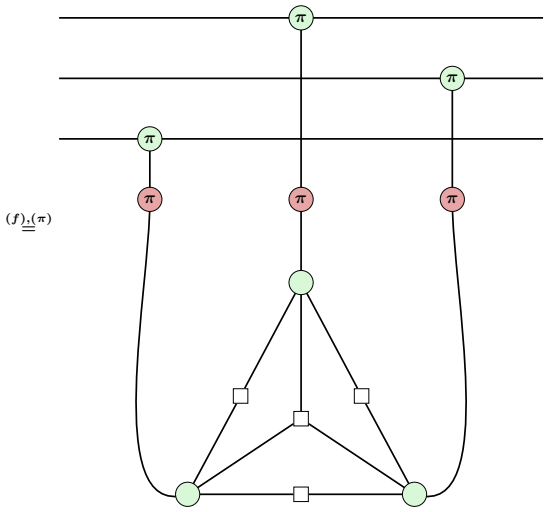


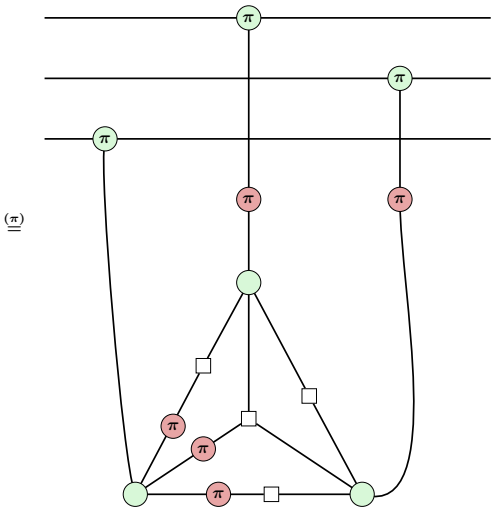
(61)

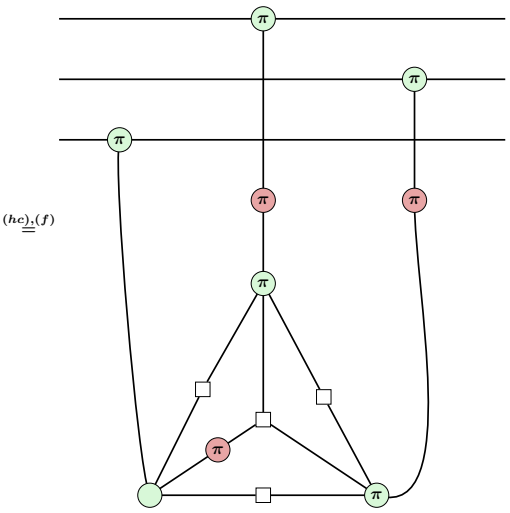
Case 4. $c_1 = c_2 = 1 = c_3 = 1$

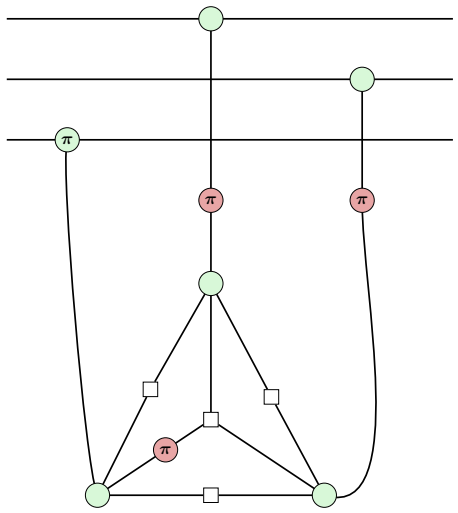


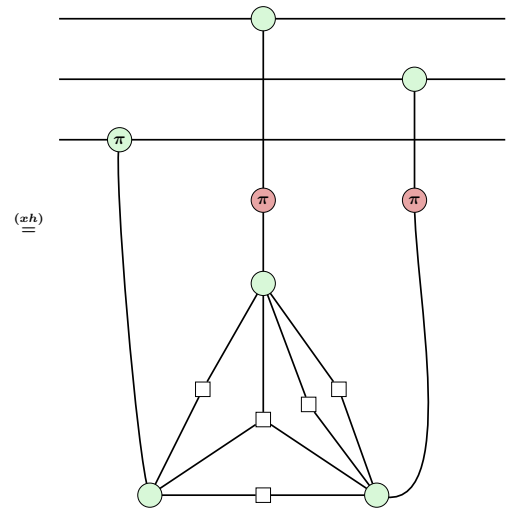

$(b_1+a_3+1)\pi$
$(a_1+b_2+1)\pi$
$(a_2+b_3+1)\pi$
π
$b_3\pi$
$a_1\pi$
$a_3\pi$
$b_1\pi$
$b_2\pi$
$a_2\pi$
(f)
(hc)
$(\pi),(f)$
$(hc),(f)$
$(b_1+a_3)\pi$
$(a_2+b_3)\pi$
$(a_1+b_2)\pi$

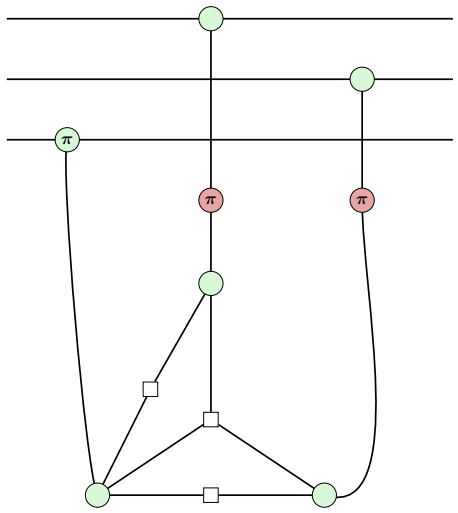

(f),(π)

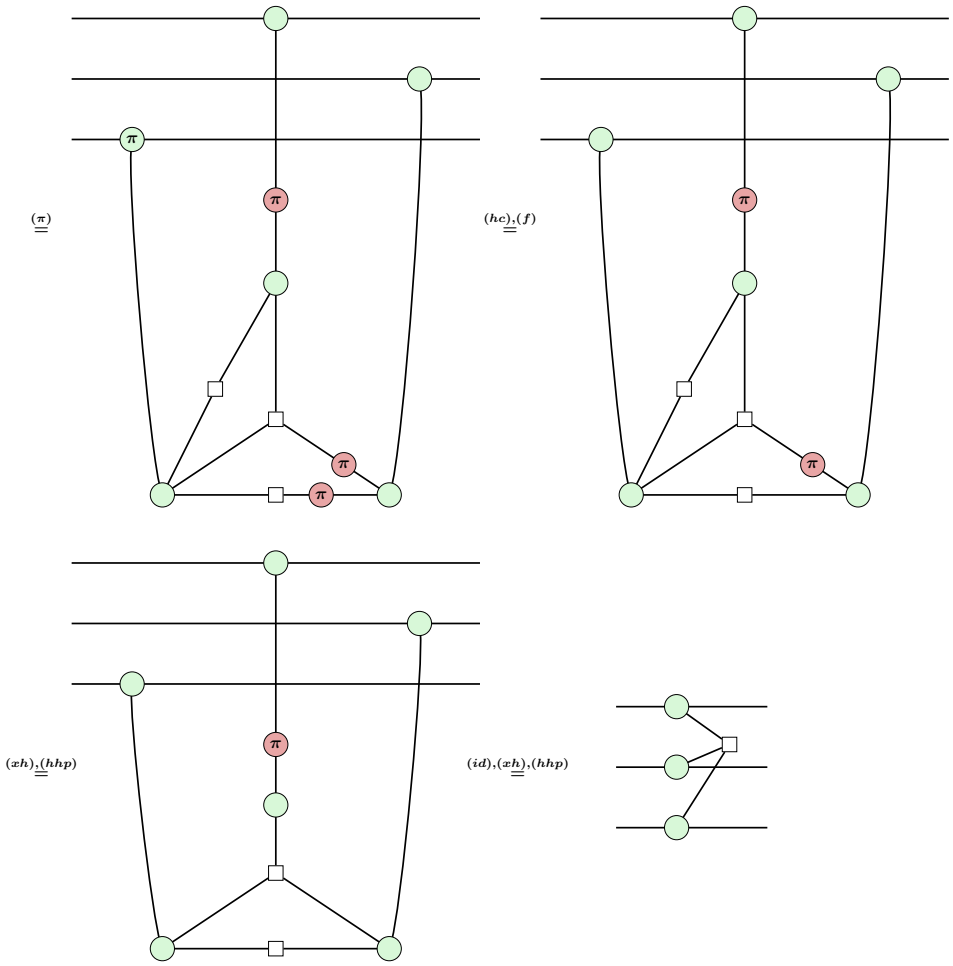

(π)


(hc),(f)


(π),(f)


(xh)


(hhp)



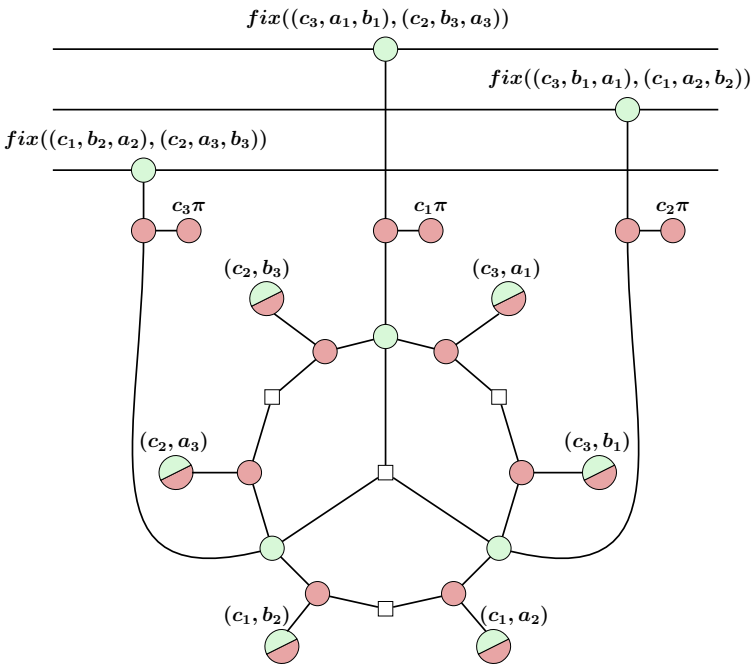
(62)

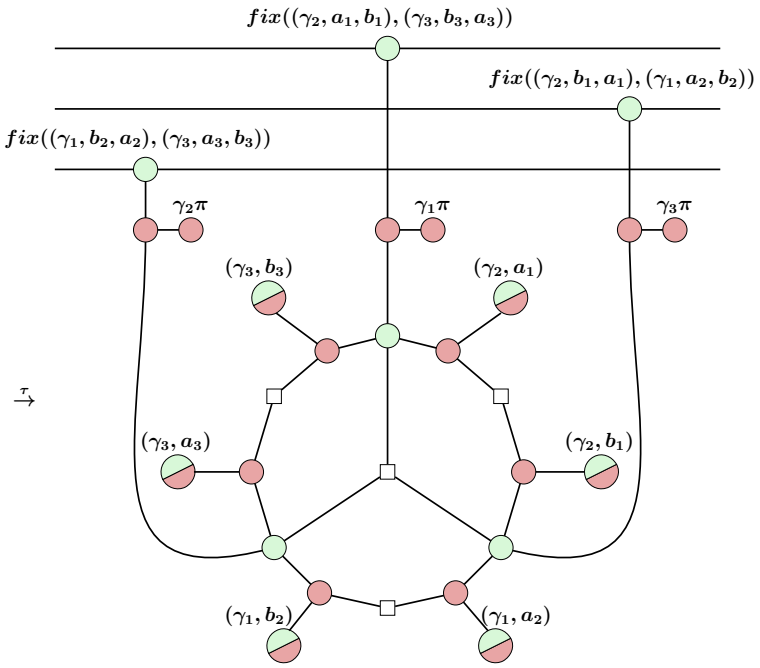
□

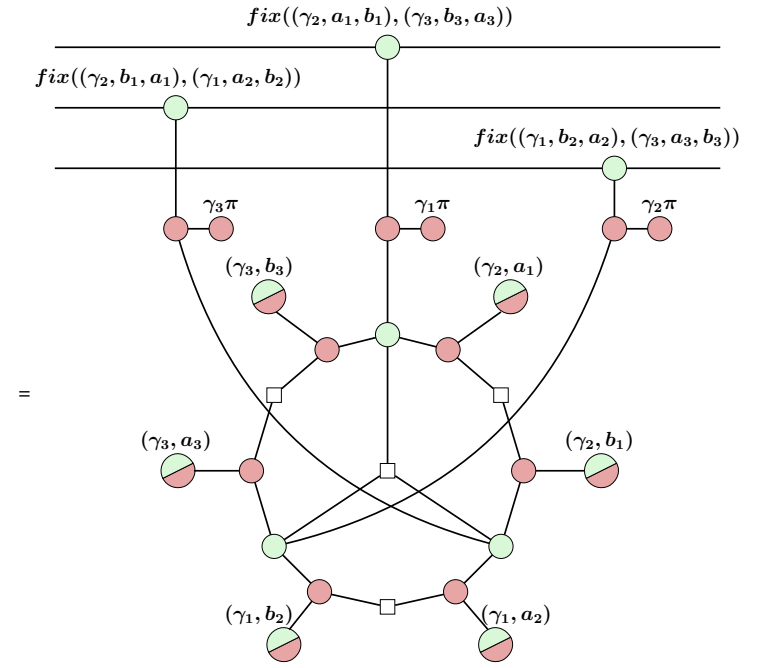
We can generate the set of all possible measurement outcomes $S = \{0, 1\}^3$, where the ith bit corresponds to the value of outcome c_i, by transposing (i.e., swapping) certain bits of the set of already proven combinations $T = \{000, 100, 110, 111\}$. Indeed, taking the transpositions $\sigma = (c_1\ c_2)$ and $\tau = (c_2\ c_3)$, we can generate the remaining elements of S. For instance, the missing elements that represent one measurement outcome being 1 and the rest being 0 can be generated by $\sigma(100) = 010$, and $\tau \circ \sigma(100) = 001$. The remaining bitstrings are generated similarly. Hence, we need to show that performing the transpositions σ, τ on the ZX-diagram of equation (57) preserves the underlying matrix. This would imply that the remaining cases generated by the transpositions are equivalent to the already proven cases.

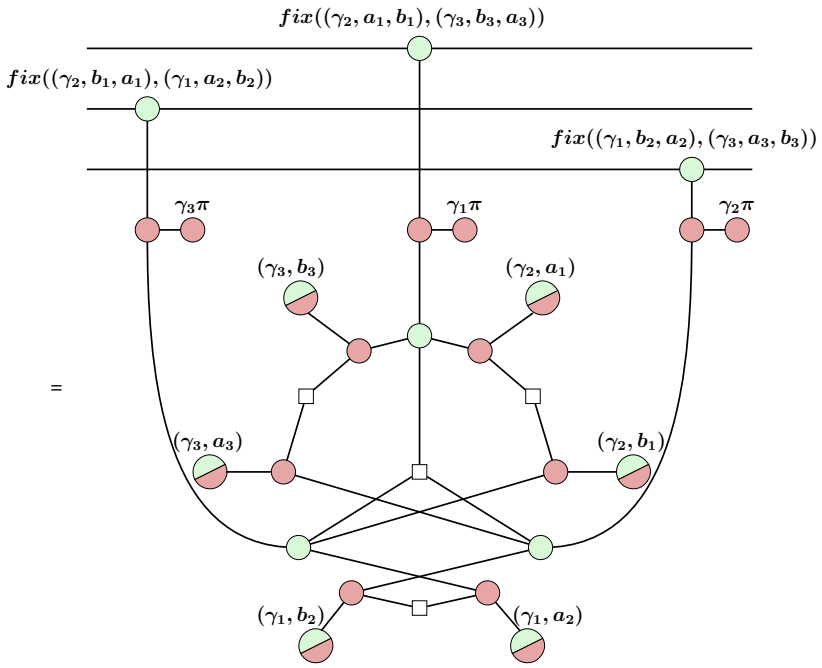
Lemma 2. *Let any arbitrary measurement result $c_1 = \gamma_1, c_2 = \gamma_2, c_3 = \gamma_3$, for $\gamma_1, \gamma_2, \gamma_3 \in \{0, 1\}$, the AutoCCZ diagrams before and after transposing the measurement results by $\tau = (c_2\ c_3)$ are equivalent.*

Proof. We apply τ to the AutoCCZ diagram by writing $c_1 = \gamma_1, c_2 = \gamma_3, c_3 = \gamma_2$. Then, we rearrange the diagram by moving nodes and wires (which does not modify the underlying matrix):

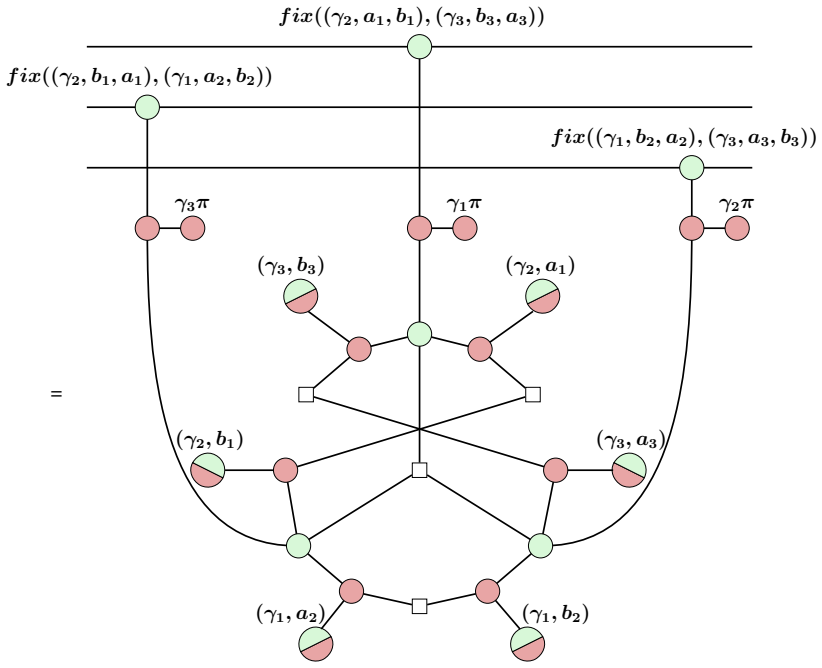



$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$\gamma_2\pi$
$\gamma_1\pi$
$\gamma_3\pi$
(γ_3, b_3)
(γ_2, a_1)
$\xrightarrow{\tau}$
(γ_3, a_3)
(γ_2, b_1)
(γ_1, b_2)
(γ_1, a_2)

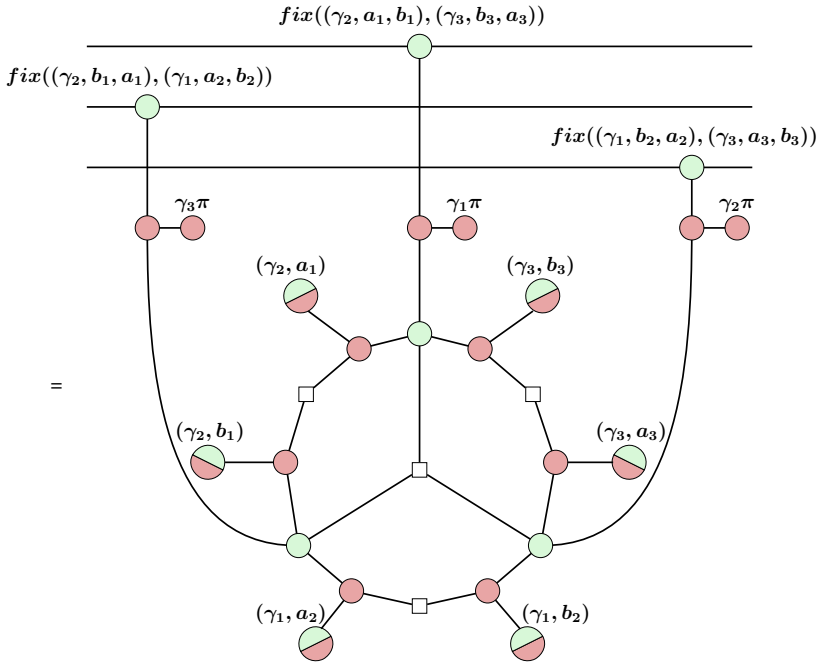

$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_3, b_3)
(γ_2, a_1)
=
(γ_3, a_3)
(γ_2, b_1)
(γ_1, b_2)
(γ_1, a_2)

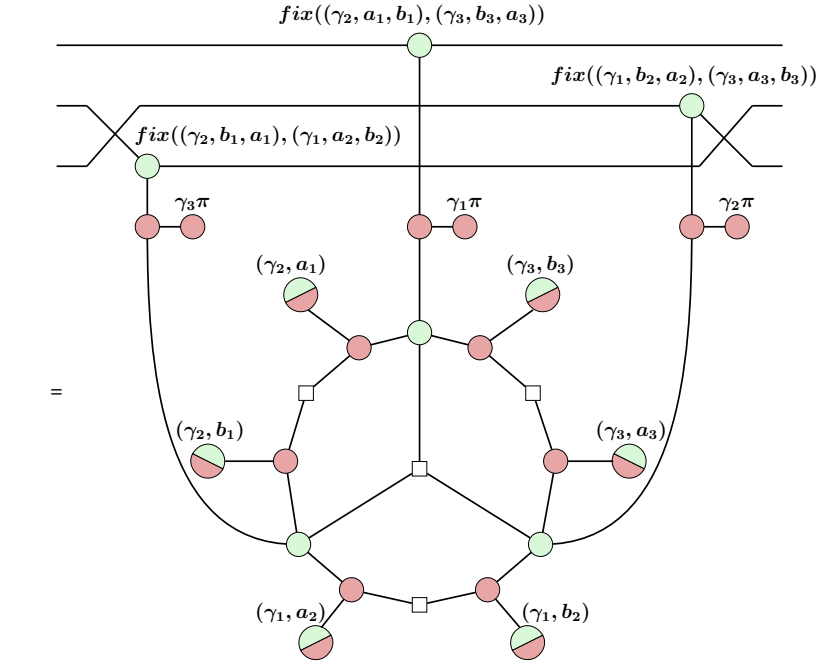

$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_3, b_3)
(γ_2, a_1)
(γ_3, a_3)
(γ_2, b_1)
(γ_1, b_2)
(γ_1, a_2)

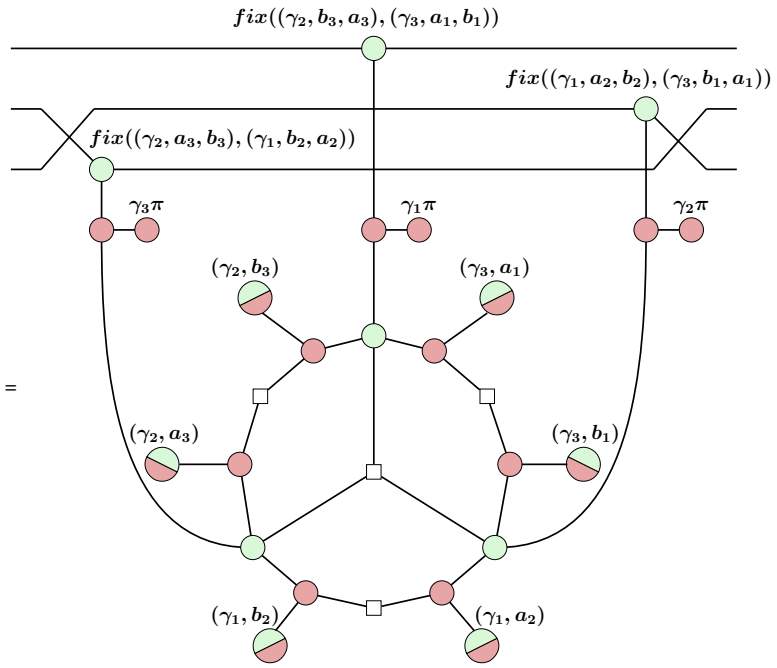
=


$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_3, b_3)
(γ_2, a_1)
(γ_2, b_1)
(γ_3, a_3)
(γ_1, a_2)
(γ_1, b_2)

=


$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, a_1)
(γ_3, b_3)
=
(γ_2, b_1)
(γ_3, a_3)
(γ_1, a_2)
(γ_1, b_2)

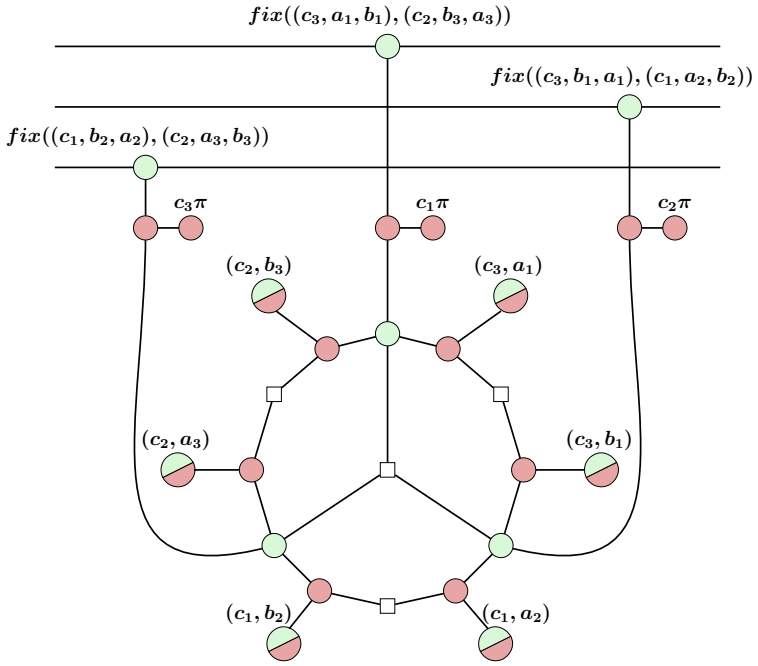

$fix((\gamma_2, a_1, b_1), (\gamma_3, b_3, a_3))$
$fix((\gamma_1, b_2, a_2), (\gamma_3, a_3, b_3))$
$fix((\gamma_2, b_1, a_1), (\gamma_1, a_2, b_2))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, a_1)
(γ_3, b_3)
=
(γ_2, b_1)
(γ_3, a_3)
(γ_1, a_2)
(γ_1, b_2)



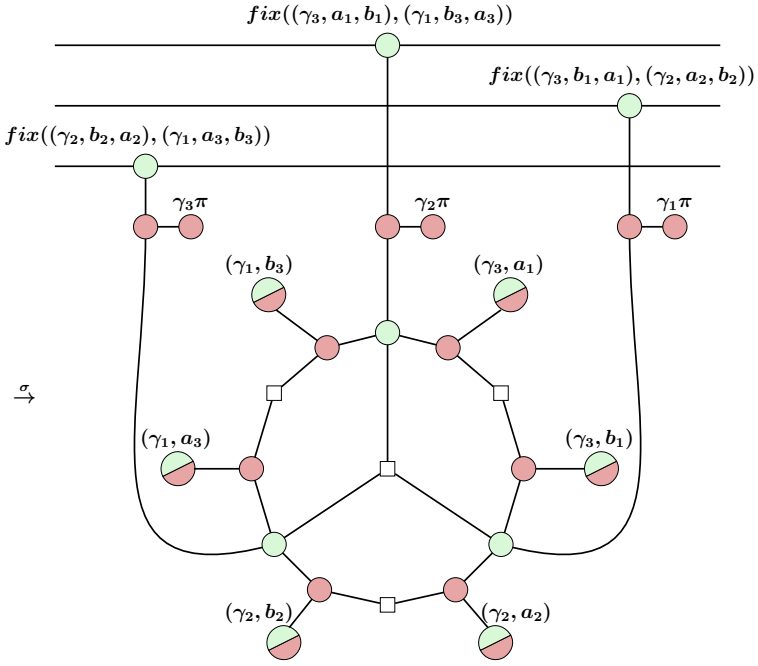
Remark. This process aims to recover the original diagram from the transposed one. We do so by moving every spider with either a γ_2 or γ_3 annotation to their respective places before the transposition. Intuitively, this operation is a reflection across the y axis in the middle of the diagram. We also needed to bend the wires of the main qubits at the beginning in order to get the top spiders in the correct place, and then bend back to the original place at the end. The last step, where we replace the labels by $a_2 \leftrightarrow b_2, a_3 \leftrightarrow b_1, a_1 \leftrightarrow b_3$, does not modify the diagram. Indeed, since we are not assuming any values on these variables, the last step is merely a variable renaming, and results in the same diagram as before the transposition τ. □

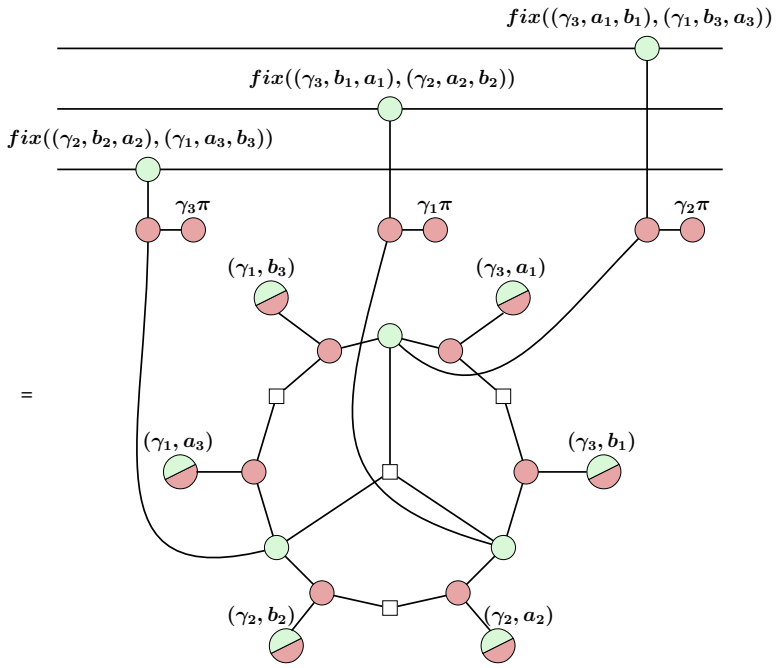
Lemma 3. *Let any arbitrary measurement result $c_1 = \gamma_1, c_2 = \gamma_2, c_3 = \gamma_3$, for $\gamma_1, \gamma_2, \gamma_3 \in \{0, 1\}$, the AutoCCZ diagrams before and after transposing the measurement results by $\sigma = (c_1\ c_2)$ are equivalent.*

Proof. We write $c_1 = \gamma_2, c_2 = \gamma_1, c_3 = \gamma_3$ in the AutoCCZ diagram. Then, we do the same process as in lemma 2:

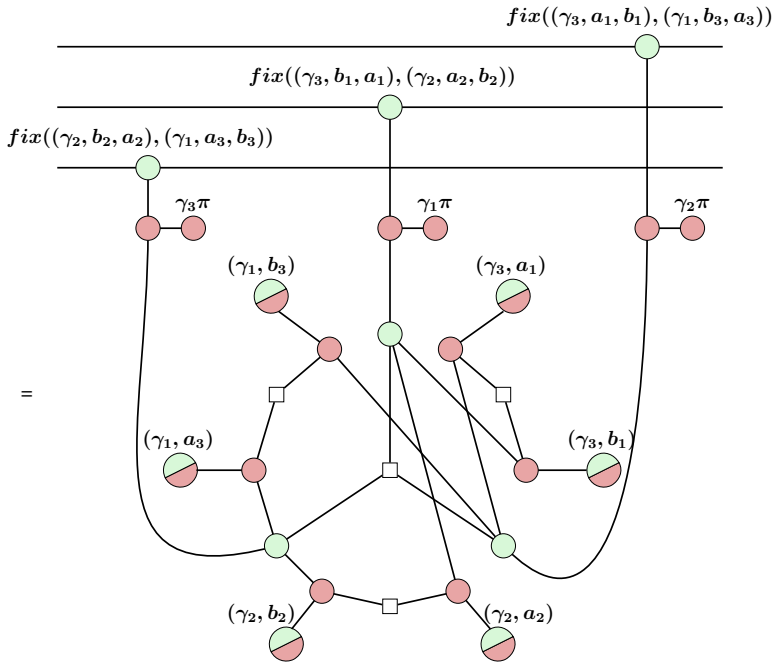


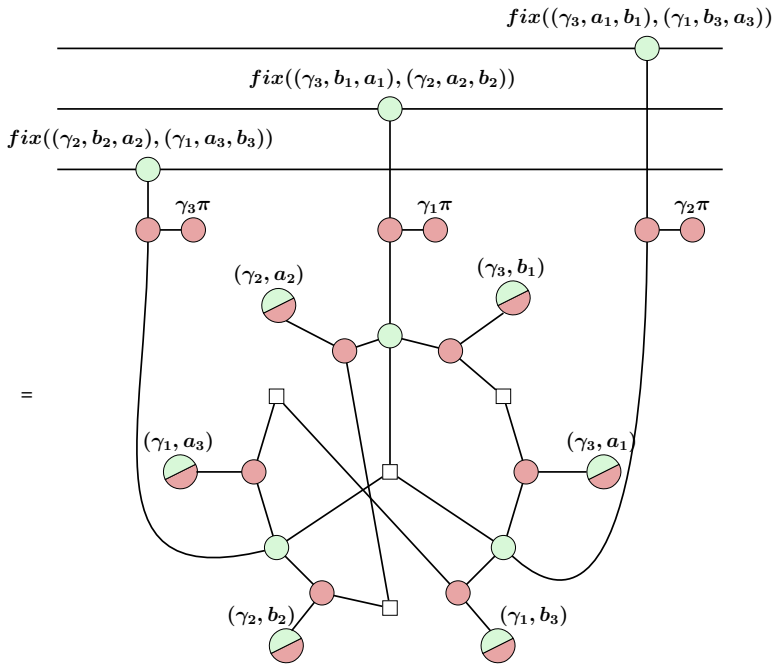
$\xrightarrow{\sigma}$



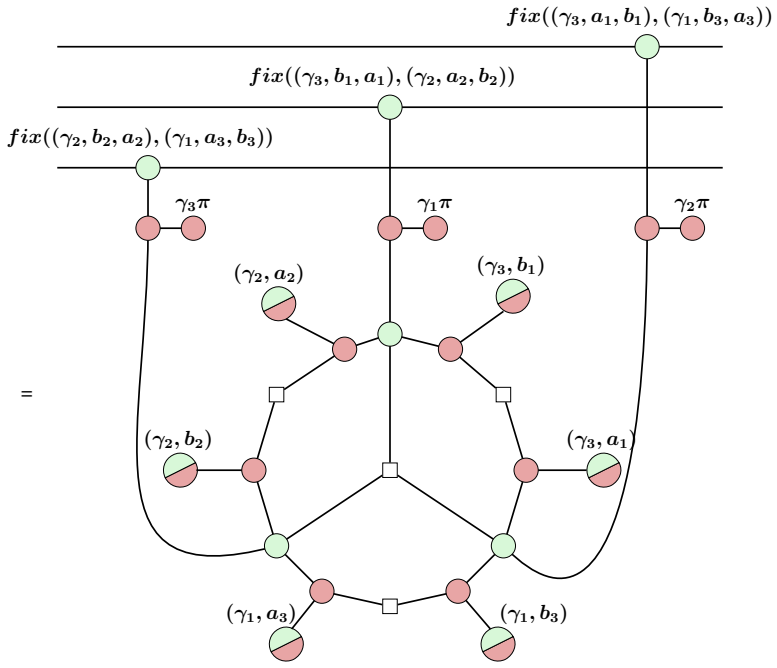

$fix((\gamma_3, a_1, b_1), (\gamma_1, b_3, a_3))$
$fix((\gamma_3, b_1, a_1), (\gamma_2, a_2, b_2))$
$fix((\gamma_2, b_2, a_2), (\gamma_1, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_1, b_3)
(γ_3, a_1)
(γ_1, a_3)
(γ_3, b_1)
(γ_2, b_2)
(γ_2, a_2)

=

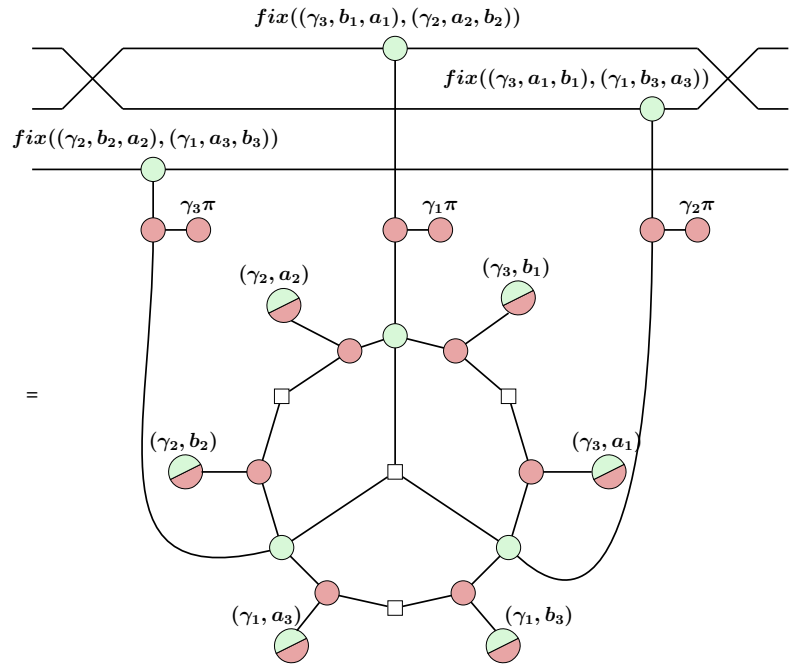

$fix((\gamma_3, a_1, b_1), (\gamma_1, b_3, a_3))$
$fix((\gamma_3, b_1, a_1), (\gamma_2, a_2, b_2))$
$fix((\gamma_2, b_2, a_2), (\gamma_1, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_1, b_3)
(γ_3, a_1)
(γ_1, a_3)
(γ_3, b_1)
(γ_2, b_2)
(γ_2, a_2)


$fix((\gamma_3, a_1, b_1), (\gamma_1, b_3, a_3))$
$fix((\gamma_3, b_1, a_1), (\gamma_2, a_2, b_2))$
$fix((\gamma_2, b_2, a_2), (\gamma_1, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, a_2)
(γ_3, b_1)
(γ_1, a_3)
(γ_3, a_1)
(γ_2, b_2)
(γ_1, b_3)

=

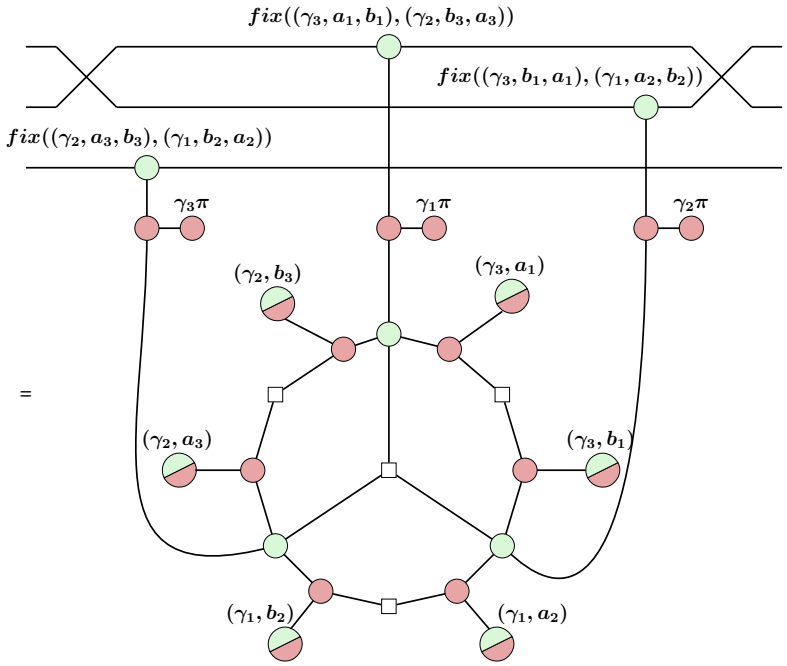

$fix((\gamma_3, a_1, b_1), (\gamma_1, b_3, a_3))$
$fix((\gamma_3, b_1, a_1), (\gamma_2, a_2, b_2))$
$fix((\gamma_2, b_2, a_2), (\gamma_1, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, a_2)
(γ_3, b_1)
(γ_2, b_2)
(γ_3, a_1)
(γ_1, a_3)
(γ_1, b_3)

=


$fix((\gamma_3, b_1, a_1), (\gamma_2, a_2, b_2))$
$fix((\gamma_3, a_1, b_1), (\gamma_1, b_3, a_3))$
$fix((\gamma_2, b_2, a_2), (\gamma_1, a_3, b_3))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, a_2)
(γ_3, b_1)
(γ_2, b_2)
(γ_3, a_1)
(γ_1, a_3)
(γ_1, b_3)

=

(63)


$fix((\gamma_3, a_1, b_1), (\gamma_2, b_3, a_3))$
$fix((\gamma_3, b_1, a_1), (\gamma_1, a_2, b_2))$
$fix((\gamma_2, a_3, b_3), (\gamma_1, b_2, a_2))$
$\gamma_3\pi$
$\gamma_1\pi$
$\gamma_2\pi$
(γ_2, b_3)
(γ_3, a_1)
(γ_2, a_3)
(γ_3, b_1)
(γ_1, b_2)
(γ_1, a_2)

=

□

Theorem 1. *The self-correcting CCZ construction of Gidney and Fowler [16] is equivalent to a CCZ gate for every possible measurement result of the $|CCZ\rangle$ and delayed-choice CZ qubits.*

Proof. In Lemma 1 we explicitly showed the equivalence for four of the eight possible measurement results of the $|CCZ\rangle$ state and any possible result of the conditional measurements. In Lemma 2 and Lemma 3, we proved that the remaining cases are equivalent to the ones shown in Lemma 1. Then, it follows that the self-correcting CCZ from Ref. [16] is equivalent to a CCZ gate. □

7 Conclusion and future work

In this work, we have proven the correctness of several quantum gate teleportation techniques using the ZX-calculus. The list of formally verified operations is the following:

- First, the delayed-choice S gate was proven correct based on how it must behave for each value of the control bit. We primarily showed this to demonstrate how a delayed-choice quantum gate works. After this, we focused our work on verifying the AutoCCZ, which required the following proofs.
- We modeled in the ZX-calculus an unrefined CCZ teleportation protocol, in which we were able to see what byproduct operations could emerge from it. This proved formally the premises on which the AutoCCZ is based.
- Next, we translated the definition of the delayed-choice CZ (in quantum circuit notation) to the ZX-calculus, and with this new construction, we proved its correctness.
- Lastly, we did the same with the AutoCCZ. We translated the definition from circuit notation to ZX-calculus and then proved its correctness.

These ZX-calculus representations of the delayed-choice CZ and AutoCCZ set the ground for future work. For instance, we can use these constructions to design time-efficient algorithms and protocols on the surface code requiring these gates now that we have ensured their correctness. Furthermore, we showed how the ZX-rules express very naturally how Pauli gates in a Pauli Frame can traverse a computation and, in many cases, end up inverting a measurement result. Hence, when laying out computations that use the AutoCCZ, we can now use the ZX-calculus to study which gates with potential byproduct Paulis need to be resolved before the delayed-choice CZ gates [26], as these byproduct Paulis can potentially determine on which basis we need to perform our measurements. An actual quantum device that makes use of the AutoCCZ would certainly require such an algorithm for dependency detection.

References

[1] Dorit Aharonov. A simple proof that toffoli and hadamard are quantum universal, 2003.

[2] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. *Electronic Proceedings in Theoretical Computer Science*, 287:23–42, Jan 2019.

[3] Niel Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to reduce $\pi/4$-parity phase circuits, motivated by the zx calculus, 11 2019.

[4] H. Bombin. Topological order with a twist: Ising anyons from an abelian model. *Physical Review Letters*, 105(3), jul 2010.

[5] S. B. Bravyi and A. Yu. Kitaev. Quantum codes on a lattice with boundary, 1998.

[6] Carlton M. Caves. Stabilizer formalism for qubits, 2014.

[7] Bob Coecke and Ross Duncan. Interacting quantum observables. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, pages 298–310, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

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

[9] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020.

[10] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020.

[11] Austin G. Fowler. Time-optimal quantum computation, 2012.

[12] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), sep 2012.

[13] Austin G. Fowler, David S. Wang, and Lloyd C. L. Hollenberg. Surface code quantum error correction incorporating accurate error propagation. 2010.

[14] Spiro Gicev, Lloyd C. L. Hollenberg, and Muhammad Usman. A scalable and fast artificial neural network syndrome decoder for surface codes, 2021.

[15] Craig Gidney. Spacetime tradeoffs when optimizing large quantum computations. IQFA'11, 2020.

[16] Craig Gidney and Austin G. Fowler. Flexible layout of surface code computations using autoccz states, 2019.

[17] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127–137, jan 1998.

[18] Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Frederic T. Chong. Nisq+: Boosting quantum computing power by approximating quantum error correction, 2020.

[19] Clare Horsman. Quantum picturalism for topological cluster-state computing. *New Journal of Physics - NEW J PHYS*, 13, 09 2011.

[20] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, dec 2012.

[21] Aleks Kissinger and J. J. Van De Wetering. Reducing the number of non-clifford gates in quantum circuits. *Physical Review A*, 102:022406, 2020.

[22] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, Jan 2003.

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

[24] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, mar 2019.

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

[26] Alexandru Paler. Personal communication, 2022.

[27] L. Riesebos, X. Fu, S. Varsamopoulos, C. G. Almudever, and K. Bertels. Pauli frames for quantum computer architectures. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Ryan Sweke, Markus S Kesselring, Evert P L van Nieuwenburg, and Jens Eisert. Reinforcement learning decoders for fault-tolerant quantum computation. *Machine Learning: Science and Technology*, 2(2):025005, dec 2020.

[29] Dominique Unruh. Quantum cryptography lecture notes, spring 2022, 2022.

[30] John van de Wetering. ZX-calculus for the working quantum computer scientist. arXiv:2012.13966, 2020.

[31] Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, feb 2020.

[32] Renaud Vilmart. A near-optimal axiomatisation of zx-calculus for pure qubit quantum mechanics, 2018.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Alejandro David Villoria González**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Verification of Quantum AutoCCZ States Using the ZX-calculus,
   (title of thesis)

   supervised by Dr. Dirk Oliver Theis.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Alejandro David Villoria González
16/05/2022