Preserving MWPM Decodability in Fault-Equivalent Rewrites

Maximilian Tim Schweikart

St Hilda's College

University of Oxford

A thesis submitted for the degree of

MSc Advanced Computer Science

Trinity 2025

Word count: 26,636

Acknowledgements

I would like to thank my supervisors Aleks Kissinger, Benjamin Rodatz and Linnéa Gräns-Samuelsson for answering my endless stream of questions and pushing the boundaries of this project. I would like to express my deepest gratitude for your invaluable guidance, mentorship and collaboration. It was a true privilege to work with you on this project.

Abstract

For the successful application of quantum computers, the suppression of noise has often been dubbed as the most critical problem. Quantum Error Correction (QEC) codes encode the state of logical qubits onto a higher number of physical qubits, adding reduncancy to make limited amounts of noise detectable and correctable in theory. However, computing the right correction for a given syndrome is generally $\mathcal{NP}$-complete. Yet, corrections must be performed at a high frequency to suppress noise successfully. For the surface code, the decoding problem is guaranteed to have a specific structure that enables efficient decoding through the Minimum-Weight Perfect Matching (MWPM) decoder. This property can be lost when constructing implementations for the detector measurements, making the decoding problem hard again.

In this work, we take a circuit-centric perspective to define *matchability-preserving* rewrites. Based on the ZX calculus and the recently introduced fault-equivalent rewrites, we formalise how the decoding problem changes through the application of ZX rewrites to a given diagram and detector basis. This allows us to analyse whether given ZX rewrites preserve the property that enabled efficient MWPM decoding. We demonstrate a set of matchability-preserving rewrites that can be used to build fault-tolerant syndrome measurements for surface codes.

Contents

Chapter 1

Introduction

Quantum computers were first theoretically defined in the 1980s, but only recently have researchers been able to build quantum computing hardware [1]. While quantum computers are anticipated to push the boundaries of what we can compute at reasonable timescales, quantum devices in the near future cannot fulfil this potential, mainly because they exhibit a lot of noise. Further developments in quantum computing hardware are expected to reduce the noise levels significantly, but it is also expected that error rates cannot be decreased to negligible levels only by hardware design, even in the long term [2].

To realise the potential of quantum computers in spite of physical noise, Fault-Tolerant Quantum Computing (FTQC) approaches aim to construct quantum computation procedures that assume a limited amount of noise can occur during computation, and that can produce correct results nonetheless [3]. Many FTQC approaches arise from Quantum Error Correction (QEC) methods which protect a fixed quantum state from noise. Similarly to codes for traditional computing and communication, QEC codes encode the state of noiseless, so-called logical qubits on a larger set of physical, noisy qubits, adding redundancy. Not all states in the larger physical code space belong to a logical state, and we can use code-specific measurements to detect whether a state is a correctly encoded state. The sequence of these detector measurement results is called a syndrome and the syndrome is zero if no error was detected. For good codes, various amounts of noise will disturb the physical state in ways that no longer belong to logical states and that produce a non-zero syndrome, which makes the error detectable [4, 3].

However, detectability alone is insufficient, as the QEC procedure needs to be able to compute which correction to apply to the distorted state. This is the job of the so-called decoder, which takes the measured syndrome as input and produces the correction instructions as output. Even in cases where there is a unique mapping between faults and syndromes, computing the fault that caused the syndrome is generally an $\mathcal{NP}$-complete problem [5, 6].

For real-world implementations of QEC codes, inefficient decoding is a deal breaker as decoders need to be run with a high frequency to keep computations correct. While there is a wide variety of QEC codes available in the literature, efficient decodability is often neglected in the analysis of the codes and set aside as an implementation detail [7].

One class of efficiently decodable QEC codes is that of codes where each atomic fault is *matchable*, that is, it triggers at most two detectors. For these codes, the Minimum-Weight Perfect Matching (MWPM) decoder is a popular choice as it can be implemented efficiently with millions of executions per second [8, 9, 7]. QEC codes are not generally matchable but some topological codes and particularly the surface code family and many of its variations are matchable [9, Sec. 1]. Furthermore, the decoding problem for the colour code can be reduced to matchable surface code decoding problems [10] and the "splitting" technique can be used to approximate some unmatchable decoding problems with matchable ones [11]. This makes matchability and MWPM decoding a prime example of efficient decoding.

Another problem with implementing QEC codes is that they are often specified in terms of the codespace they fix. For stabiliser codes, the codespace is fixed by a finite set of stabiliser generators which also correspond to the code's detector measurements. These measurements need to be decomposed to components from the hardware's native gate set when implementing such a code on real quantum computing hardware [3]. In these decompositions, it is insufficient to find gates whose noiseless maps compose to the original measurement because the multitude of newly introduced gates comes with new potential sources of noise. In such a circuit-centric perspective, noise does not only occur between rounds of syndrome measurements, but within the circuits implementing these measurements [5, 12]. Rodatz et al. formalise this problem with *fault-equivalent rewrites* which relate circuit fragments with each other if they exhibit the same behaviour in noisy environments [13].

This brings us to the problem that will be discussed in this work. Fault-equivalent rewrites enable us to rewrite idealised circuit components into implementations for real hardware while considering the changing noise model but they do not necessarily preserve the matchability property. In fact, we show that some fault-equivalent rewrites will turn matchable codes into unmatchable ones. Our goal is to characterise which fault-equivalent rewrites preserve matchability and to develop a new set of rewrites that are fault-equivalent and matchability-preserving.

To examine matchability preservation, we pick up where Rodatz et al. left off in [13, 14] and apply the ZX calculus, a diagrammatic reasoning framework for quantum circuits, to the problem at hand. The ZX calculus represents circuits by composing so-called spiders and wires to represent the linear map of the quantum circuit. The bare ZX calculus has a complete set of rules that allows rewriting any diagram representation of a linear map into any other diagram representation of the same linear map, but not all of these rewrites produce fault-equivalent diagrams. Rodatz et al.'s *"Fault Tolerance by Construction"* paper provides another set of ZX rewrites that satisfy the fault equivalence constraint [13]. Although the set of fault-equivalent rewrites isn't proven to be complete, it is sufficient for fault-equivalently decomposing stabiliser measurements into single-qubit measurements and CNOTs.

We refine these rewrites by considering how atomic faults on the wires of the rewritten part of the diagram affect the detectors of the spacetime code. Following Bombín et al., we represent detectors as Pauli webs to visually incorporate them into the diagram structure [15] and extend the ZX calculus to rewrite the detector structure along with the diagram. This allows us to analyse in which cases fault-equivalent rewrites also preserve matchability. Where fault-equivalent rewrites do not permit matchability preservation, we are able to suggest novel, alternative rewrites and clear conditions for when these alternatives are applicable. Specifically, the correctness of some of these rewrites can be guaranteed for a sufficiently high distance. This allows our approach to produce fault-equivalent and efficiently decodable implementations of matchable QEC codes.

The main text is organised as follows: In Chapter 2, we discuss the relevant background. While we assume a basic familiarity with quantum computing and the ZX calculus, we briefly introduce them and establish the relevant notation for this text. We then quickly advance to FTQC and QEC, defining our notion of adversarial noise models and covering the relevant properties of QEC codes and decoding problems. After lifting these concepts into the ZX realm with Pauli webs and fault-equivalent rewrites, we proceed to discuss matchability and specifically the MWPM decoder. We also introduce the splitting method, which can recover matchability for some non-matchability-preserving rewrites.

Chapter 3 constitutes the main contribution of this work. We begin by defining *detector-aware rewrites* which allow us to track changes to a detector basis along with the diagram changes of the underlying ZX rewrite. We use this concept to define *matchability-preserving rewrites* which combine the noise model preservation of fault-equivalent rewrites with the detector tracking of detector-aware rewrites. Matchability-preserving rewrites ensure that matchable detector bases of the original diagram are mapped to matchable detector bases of the rewritten diagram. After characterising a few relevant properties of matchability-preserving rewrites, such as compositionality, existence and uniqueness, we show under which conditions the fault-equivalent rewrites from [13] preserve matchability. For cases where these fault-equivalent rewrites do not preserve matchability, we derive alternative fault-equivalent rewrites that can be used to preserve matchability

We demonstrate the application of our approach to syndrome measurements in Chapter 4, where we work through a surface code plaquette measurement example in two variants. The thesis concludes in Chapter 5 where we discuss the limitations of our approach and point out directions for future work in the field of efficient decodability preservation.

Chapter 2

Background

Quantum Error Correction (QEC) is a relatively novel field building on the theoretical foundations of quantum computing to develop tools that allow us to encode qubit data redundantly, to detect errors introduced through noise, and to correct these errors as part of the computation process. This chapter aims to introduce the relevant aspects of QEC that we will use as a foundation for the further chapters. While we introduce quantum computing and the ZX calculus with their basic building blocks in the next sections, some intuition for their use will be beneficial for following this text. We refer to [1] for a comprehensive introduction to (non-ZX) quantum computing, to [16] for a compact overview of the ZX calculus, and to [4] as an in-depth resource for studying quantum computing through the ZX calculus.

The background chapter is structured as follows: In Section 2.1, we give a brief introduction to quantum computing. We cover how qubit-based states, gates and measurements can be composed to circuits and for single qubits specifically, we cover the Pauli rotations on the Bloch sphere. Section 2.2 introduces the ZX calculus. After explaining how the circuit building blocks can be represented as diagrams, we turn our attention to ZX rewrites and how they can be used to change diagrams while maintaining their semantics. From Section 2.3 onwards, we narrow our scope to QEC and Fault-Tolerant Quantum Computing (FTQC). We motivate the relevance of QEC, describe our notion of adversarial noise models and introduce detectors and codes. We also introduce some relevant bits of stabiliser theory and the surface code, which will be our motivating example throughout this work. Section 2.4 describes a specialisation of noise models and detectors to the

ZX calculus. We introduce Pauli webs as diagrammatic representations of stabilisers and detectors and introduce fault-equivalent rewrites for ZX diagrams. In Section 2.5, we introduce the Minimum-Weight Perfect Matching decoder which enables efficient decoding for the *matchable* subclass of error-detector models that includes the surface code family. We also tailor the concept of matchability to ZX diagrams and Pauli webs. Finally, in Section 2.6, we cover *splitting*, a method for turning unmatchable error-detector models into matchable ones.

2.1 Quantum computing

Quantum computers differ fundamentally from traditional computers in the ways they store and process data, but also in how we can insert and extract data to and from them. While classical computers usually work with bit registers, that is, arrays of variables that can either be in the zero or one state, quantum computers relax this requirement and allow so-called superpositions of multiple basis states. The most common data register for universal quantum computers is the *qubit*, whose states can be described as superpositions of two basis states. More generally, *qudit* states are superpositions of d basis states where $d \in \mathbb{N}, d \geq 2$ can be chosen arbitrarily. For the purpose of this work, however, we will focus on quantum computing in qubit-based systems.

In this section, we will first give an intuition for qubit states, operators, and measurements on a single qubit. We will then discuss how this generalises to multi-qubit systems and what new phenomena we can observe on such systems. Furthermore, we will introduce the quantum circuit model which gives us a first, visual representation of quantum programs.

2.1.1 Quantum computing on a single qubit

A *quantum bit (qubit)* is a quantum data register with two basis states. We usually refer to these states as $|0\rangle$ and $|1\rangle$ in analogy to the two possible states of a classical bit. A qubit state, however, can be in any complex, normalised superposition of these two states.

Definition 2.1 (Pure qubit state)**.** *A qubit state* $|\psi\rangle$ *is a complex, normalised superposition of* $|0\rangle$ *and* $|1\rangle$*, i.e.*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

for some $\alpha, \beta \in \mathbb{C}$ *such that* $|\alpha|^2 + |\beta|^2 = 1$*.*

Mathematically, the qubit state lives on the unit sphere surface within a two-dimensional, complex vector space $\mathcal{H} \equiv \mathbb{C}^2$. Moreover, the $|\psi\rangle$ notation is motivated by the inner product and adjoint states in $\mathcal{H}$.

Definition 2.2 (Bra-ket notation)**.** *For a quantum state* $|\psi\rangle$*, we define*

$$\langle\psi| := |\psi\rangle^\dagger \tag{2.2}$$

where $\cdot^\dagger$ *refers to the adjoint of the argument. We refer to* $\langle\psi|$ *as a* bra*, and to* $|\phi\rangle$ *as a* ket *because they make up an inner product bra-ket (bracket) in the following equation:*

$$\langle\psi| \cdot |\phi\rangle = |\psi\rangle^\dagger \cdot |\phi\rangle = \langle\psi|\phi\rangle \tag{2.3}$$

The basis from our qubit state definition (Def. 2.1) is an *orthonormal basis* w.r.t. the inner product. We refer to it as the Z basis (or the *computational basis*) and there are two more orthonormal bases of interest for us.

Definition 2.3 (X, Y, and Z basis)**.**

(i) The X *basis is made up of the* $|+\rangle$ *and the* $|-\rangle$ *states, which we define as follows:*

$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.4}$$

(ii) The Y *basis is made up of the* $|+i\rangle$ *and the* $|-i\rangle$ *states, which we define as follows:*

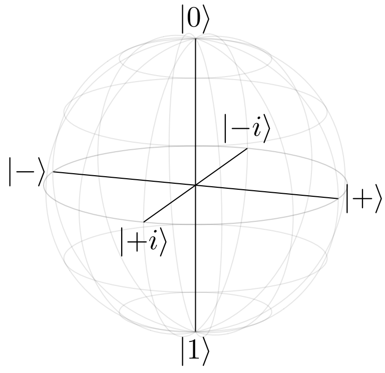
$$|+i\rangle := \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle) \qquad |-i\rangle := \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle) \tag{2.5}$$

(iii) The Z *basis is made up of the* $|0\rangle$ *and the* $|1\rangle$ *state.*

The names of these bases refer to the position of the basis elements on the *Bloch sphere*. The Bloch sphere is an embedding of the qubit state into the three-dimensional real coordinate space that puts emphasis to those observeable degrees of freedom in a quantum state. Specifically, we can also parameterise the qubit state definition from Definition 2.1 as follows, thanks to the normalisation [1, Sec. 1.2]:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \tag{2.6}$$

Here, γ, θ and φ are real numbers. While it is physically impossible to distinguish qubit states that only differ in their γ value, θ and φ are observable and we can interpret them as polar coordinates on a three-dimensional unit sphere. On this Bloch sphere, the basis elements mentioned in the preceeding definition distribute onto the corresponding axes.



(2.7)

This geometric interpretation of quantum states already motivate a few basic operations on single qubit states, the first of which being rotations on the Bloch sphere. For that, let us first characterise what constitutes an operation on a quantum state in our model.

Definition 2.4 (Gate)**.** *A logical operation on a quantum state corresponds to a unitary, linear map $U : \mathcal{H} \to \mathcal{H}$ that acts on the state space $\mathcal{H}$. We refer to these operations as* gates*.*

Rotations of states around the axes of the Bloch sphere correspond to gates. We refer to a rotation with angle ϕ around the X axis as the $X(\phi)$ gate and similarly, we refer to rotations around the other axes as $Y(\phi)$ and $Z(\phi)$.

For single-qubit quantum computing, any two of these gate families are already *uni-*

versal, meaning that we can express any single-qubit logical operation as a composition of these gates. Some of these gates are of particular importance for this work, so we introduce them with their own names.

Definition 2.5 (Common single-qubit gates)**.**

(i) $X(\theta)$*,* $Y(\theta)$*, and* $Z(\theta)$ *are called* Pauli rotation *gates.*

(ii) We refer to $X := X(\pi)$*,* $Y := Y(\pi)$*, and* $Z := Z(\pi)$ *as the* Pauli flips*.*

(iii) The Hadamard *gate* H *is defined as follows:*

$$H|0\rangle = |+\rangle \qquad H|1\rangle = |-\rangle \tag{2.8}$$

We refer to X, Y and Z as Pauli flips because they map the basis states of the other bases onto the opposite element, and act like the identity on states of the same basis. For example, the X gate flips the Y and Z basis states, but keeps the X basis intact.

$$\begin{aligned} X|+\rangle &= |+\rangle \qquad & X|+i\rangle &= |-i\rangle \qquad & X|0\rangle &= |1\rangle \\ X|-\rangle &= |-\rangle \qquad & X|-i\rangle &= |+i\rangle \qquad & X|1\rangle &= |0\rangle \end{aligned} \tag{2.9}$$

We occasionally refer to the X gate as a *NOT* gate because it acts like a bitflip on the computational basis. Meanwhile, the Hadamard gate doesn't switch between states of the same basis but instead acts as a change-of-basis between the X and Z bases.

$$\begin{aligned} H|0\rangle &= |+\rangle \qquad & H|+\rangle &= |0\rangle \\ H|1\rangle &= |-\rangle \qquad & H|-\rangle &= |1\rangle \end{aligned} \tag{2.10}$$

The final missing piece in single-qubit quantum computing is a measurement.

Definition 2.6 (Measurement and Born rule)**.** *A measurement operator for a quantum state corresponds to a set of outcomes* O *and a set of projectors* $\{P_o : \mathcal{H} \to \mathcal{H} \mid o \in O\}$ *that sum to the identity:*

$$\sum_{o \in O} P_o = I \tag{2.11}$$

The probability of an outcome $o \in O$, when measuring state $|\phi\rangle$ is described by the Born rule*:*

$$\text{Prob}_O(o \mid |\phi\rangle) = \langle\phi| P_o |\phi\rangle \tag{2.12}$$

When a state $|\psi\rangle$ is measured, one of the outcomes $o \in O$ will be randomly selected with the probability given by the Born rule. The superposition of the state then collapses: the state is projected to a state $P_o |\psi\rangle$ and renormalised. For a Z-basis measurement, the outcomes are $O = \{0, 1\}$ and the projectors are $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$. Therefore, performing a Z measurement on a Z basis state gives a deterministic outcome:o

$$\begin{aligned}
\text{Prob}_Z(0 \mid |0\rangle) &= \langle 0| P_0 |0\rangle = \langle 0|0\rangle\langle 0|0\rangle = 1 \\
\text{Prob}_Z(1 \mid |0\rangle) &= \langle 0| P_1 |0\rangle = \langle 0|1\rangle\langle 1|0\rangle = 0 \\
P_0 |0\rangle = |0\rangle\langle 0|0\rangle &= |0\rangle \overset{\text{re-normalise}}{\rightsquigarrow} |0\rangle
\end{aligned} \tag{2.13}$$

Measuring an X basis state with the Z basis, on the other hand, gives a uniform distribution over the outcomes.

$$\begin{aligned}
\text{Prob}_Z(0 \mid |+\rangle) &= \langle +| P_0 |+\rangle = \langle +|0\rangle\langle 0|+\rangle = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2} \\
\text{Prob}_Z(1 \mid |+\rangle) &= \langle +| P_1 |+\rangle = \langle +|1\rangle\langle 1|+\rangle = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2} \\
P_0 |+\rangle = |0\rangle\langle 0|+\rangle &= \frac{1}{\sqrt{2}} |0\rangle \overset{\text{re-normalise}}{\rightsquigarrow} |0\rangle \\
P_1 |+\rangle = |1\rangle\langle 1|+\rangle &= \frac{1}{\sqrt{2}} |1\rangle \overset{\text{re-normalise}}{\rightsquigarrow} |1\rangle
\end{aligned} \tag{2.14}$$

Measurements are the only way to extract classical information out of an unknown quantum state. As the measurement collapses the superposition of the state through projection, only a single classical bit of information can be extracted from such a state.

2.1.2 Quantum computing on many qubits

We represent the joint state of separate single-qubit systems as the tensor product of their individual states:

$$|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \tag{2.15}$$

Such states live in the joint Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$ of the individual qubits' Hilbert spaces $\mathcal{H}_i$. Since $H_i \equiv \mathbb{C}^2$ for all $1 \leq i \leq n$, the joint Hilbert space is isomorphic to $\left(\mathbb{C}^2\right)^n$.

We can compose the Z basis states of the individual qubits in parallel to get a multi-qubit Z basis of $\mathcal{H}$.

$$|b_1 \cdots b_n\rangle := |b_1\rangle \otimes \cdots \otimes |b_n\rangle \quad (b_1, \ldots, b_n \in \{0, 1\}) \tag{2.16}$$

This allows us to describe arbitrary multi-qubit states as superpositions of these multi-qubit Z basis states.

Definition 2.7 (Multi-qubit state)**.** *A state $|\psi\rangle$ of n qubits is a complex, normalised superposition of the n-qubit Z basis states, i.e.*

$$|\psi\rangle = \alpha_{0\cdots00} |0 \cdots 00\rangle + \alpha_{0\cdots01} |0 \cdots 01\rangle + \cdots + \alpha_{1\cdots11} |1 \cdots 11\rangle \tag{2.17}$$

where for the $\alpha_i \in \mathbb{C}$, the normalisation condition $\sum_i |\alpha_i|^2 = 1$ holds.

With this definition, our definitions for gates (Definition 2.4), measurements and the Born rule (Definition 2.6) for single-qubit systems are also applicable to multi-qubit systems.

Not every multi-qubit state can be described as a parallel composition of single-qubit states. This non-separability is a key property of quantum systems and we call unseparable multi-qubit states *entangled*. Take for example the Bell state,

$$|\Phi_+\rangle := \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \tag{2.18}$$

which is a superposition of the $|00\rangle$ and the $|11\rangle$ state. Measuring both qubits in the Z basis will yield 0 on both qubits or 1 on both qubits with equal probability but there are no separate states that would have resulted in these random, yet correlated outcomes.

Similarly, there are multi-qubit gates that cannot be described as a parallel composition of single-qubit gates. We have summarised some of the most important unseparable multi-qubit gates in the following definition.

Definition 2.8 (Common multi-qubit gates)**.**

(i) The controlled-not (CNOT) *gate is defined by the following mapping.*

$$\begin{aligned} \mathrm{CNOT}\,|00\rangle &= |00\rangle \qquad & \mathrm{CNOT}\,|10\rangle &= |11\rangle \\ \mathrm{CNOT}\,|01\rangle &= |01\rangle \qquad & \mathrm{CNOT}\,|11\rangle &= |10\rangle \end{aligned} \tag{2.19}$$

(ii) The SWAP *gate is defined by the following mapping.*

$$\begin{aligned} \mathrm{SWAP}\,|00\rangle &= |00\rangle \qquad & \mathrm{SWAP}\,|10\rangle &= |01\rangle \\ \mathrm{SWAP}\,|01\rangle &= |10\rangle \qquad & \mathrm{SWAP}\,|11\rangle &= |11\rangle \end{aligned} \tag{2.20}$$

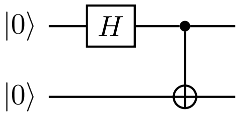
The name of the CNOT gate stems from the fact that if the first qubit is in the $|1\rangle$ state, the NOT/X gate is applied to the second qubit. If the first qubit is in the $|0\rangle$ state, the CNOT acts like an identity. It should be noted that for other states than Z basis states, the CNOT can also affect the state of the first or both qubits.

We can compose a CNOT and a Hadamard gate with two $|0\rangle$ states to create the Bell state:

$$\begin{aligned} &\mathrm{CNOT} \cdot (H \otimes I) \cdot (|0\rangle \otimes |0\rangle) \\ =\ &\mathrm{CNOT} \cdot (|+\rangle \otimes |0\rangle) \\ =\ &\frac{1}{\sqrt{2}}\mathrm{CNOT} \cdot (|0\rangle \otimes |0\rangle) + \frac{1}{\sqrt{2}}\mathrm{CNOT} \cdot (|1\rangle \otimes |0\rangle) \\ =\ &\frac{1}{\sqrt{2}}\left(|0\rangle \otimes |0\rangle\right) + \frac{1}{\sqrt{2}}\left(|1\rangle \otimes |1\rangle\right) \\ =\ &|\Phi_+\rangle \end{aligned} \tag{2.21}$$

For multi-qubit systems, specifying the composition of states, gates and measurements can become convoluted. Inspired from circuit diagrams in electronics, *quantum circuits* can help visualising such compositions more clearly. Quantum circuits represent qubits as horizontal lines and gates as boxes covering the lines of the qubits they target. Measurements and some common gates have specialised notation: Measurements are represented as box with a meter and CNOT gates draw a vertical line between a black dot on the controlling qubit and a $\oplus$ symbol on the target qubit.

For example, we can represent the Bell state preparation from Eq. (2.21) as the following quantum circuit:



(2.22)

2.2 ZX calculus

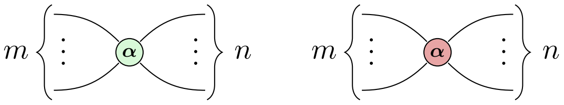
The ZX calculus is a diagrammatic reasoning tool that can be used to represent and manipulate quantum circuits in a way that preservess the semantics of the circuit, i.e. the linear map it represents. It was introduced by Bob Coecke and Ross Duncan in 2007 [17, 18] and has recently gained popularity for its applications in circuit optimisation [19]. The ZX calculus represents linear maps as compositions of just two building blocks: the Z and the X spider.

In this section, we briefly introduce the ZX calculus. After presenting its basic building blocks, we discuss how to represent quantum circuits using ZX diagrams and how to apply ZX rewrite rules to these diagrams.

2.2.1 Representing quantum circuits as ZX diagrams

We begin by defining spiders, the basic building blocks of ZX diagrams:

Definition 2.9 (Spider [4])**.** *We represent Z-spiders as green dots and X-spiders as red dots. Spiders can have an arbitrary number of legs that represent its in- and output ports. They can also have a phase $\alpha \in [0, 2\pi)$, which we assume to be zero if it is omitted.*



(2.23)

Spiders with $m \in \mathbb{N}_0$ input legs and $n \in \mathbb{N}_0$ output legs represent the following linear maps:

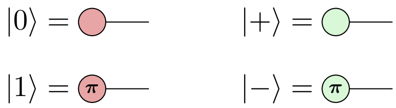
$$\begin{aligned} Z_m^n[\alpha] &= |0\rangle^{\otimes n} \langle 0|^{\otimes m} + e^{i\alpha} |1\rangle^{\otimes n} \langle 1|^{\otimes m} \\ X_m^n[\alpha] &= |+\rangle^{\otimes n} \langle +|^{\otimes m} + e^{i\alpha} |-\rangle^{\otimes n} \langle -|^{\otimes m} \end{aligned} \tag{2.24}$$

We compose these generators in parallel and in sequence using the tensor product ($\otimes$) and function composition ($\circ$ or $\cdot$) operations just like we composed quantum circuits from states, gates and measurements. The resulting structures are called *ZX diagrams*.

As we focus on Clifford circuits in this work, we categorise ZX diagrams into Clifford and non-Clifford diagrams with the following definition. We demonstrate throughout this section that we can represent all Clifford circuits as Clifford diagrams because we can represent each Clifford gate as a Clifford diagram.

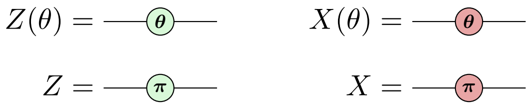
Definition 2.10 (Clifford ZX diagram [4, Sec. 5.1])**.** *A ZX diagram D is called a* Clifford diagram *if all spiders have phases that are multiples of $\frac{\pi}{2}$.*

Spiders alone already allow us to represent some of the quantum circuit building blocks introduced in the last section For instance, we can use spiders with no inputs and only a single output to represent both the Z and the X basis states:



(2.25)

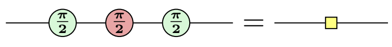
Furthermore, we can use single spiders to represent the Pauli rotation gates for the Z and the X axis, $Z(\theta)$ and $X(\theta)$ for arbitrary angles $\theta \in [0, 2\pi)$. For a half rotation $\theta = \pi$, these gates are just the Z and X flips.



(2.26)

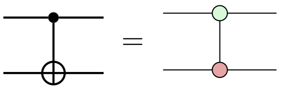
We can use sequential composition to represent the Hadamard gate from three Z and X rotations. As the ZX calculus makes use of the Hadamard gate quite often, we introduce a yellow box as its representation.

$$H = RZ\left(\frac{\pi}{2}\right) \cdot RX\left(\frac{\pi}{2}\right) \cdot RZ\left(\frac{\pi}{2}\right) = \tag{2.27}$$



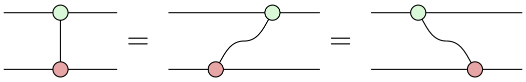
Note here and in many other equations that we use the "=" sign for equality up to a multiplicative non-zero scalar. This is a common practice in the context of the ZX calculus because quantum states are normalised and the global phase of a state is physically unobservable. For exact equalities, we refer to sections 3.1.4 and 3.6.2 in the *"Picturing Quantum Software"* textbook [4].

To represent CNOT gates, we can combine two phaseless spiders as follows.



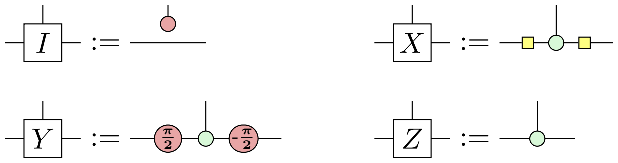
(2.28)

In this diagram, it is at first unclear whether the vertical wire is an input or an output of the two adjacent spiders. We can verify that the equality holds both when considering the vertical wire an output of the green spider and an input of the red spider, but also the other way round. This is a special case of a more general property of ZX diagrams which we discuss further below.



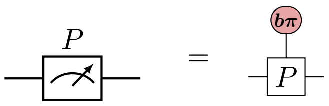
(2.29)

Finally, we need to represent measurements as compositions of spiders. We stated in Section 2.1 that a measurement is defined by a set of outcomes with projectors and that we can measure a qubit with in the three Pauli bases. We can represent such measurements in the ZX calculus through *Pauli boxes* and a binary measurement outcome $b \in \{0, 1\}$. We define the Pauli boxes as follows:



(2.30)

Then for measuring in the $P \in \{X, Y, Z\}$ basis, we use the following ZX diagram:



(2.31)

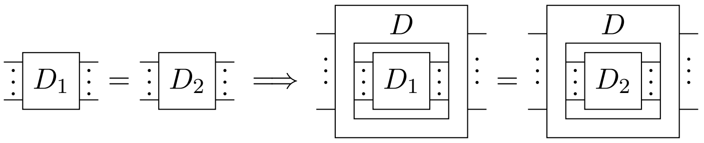
For a fixed outcome b, the RHS diagram corresponds to the projector P_b of the outcome.

2.2.2 Rewriting ZX diagrams

So far, we have only discussed how to represent quantum circuits and quantum states with ZX diagrams. This can be beneficial on its own as it allows us to reason about quantum circuits using structures generated by only two core elements, namely red and green spiders. The key advantage of the ZX calculus, however, comes from *rewriting* ZX diagrams, that is transforming the structure of the diagram while maintaining its semantics.

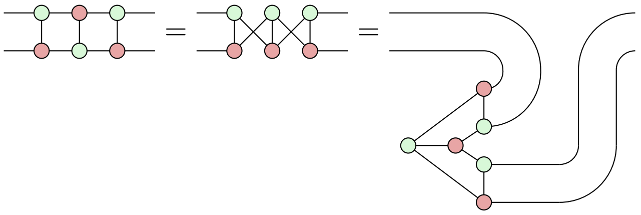
Definition 2.11 (ZX rewrite)**.** *We call equations between ZX diagrams* ZX rewrites*.*

ZX rewrites are naturally compositional: If we replace some part of a ZX diagram using a ZX rewrite, then the resulting diagram will be equivalent to the original, larger diagram.



(2.32)

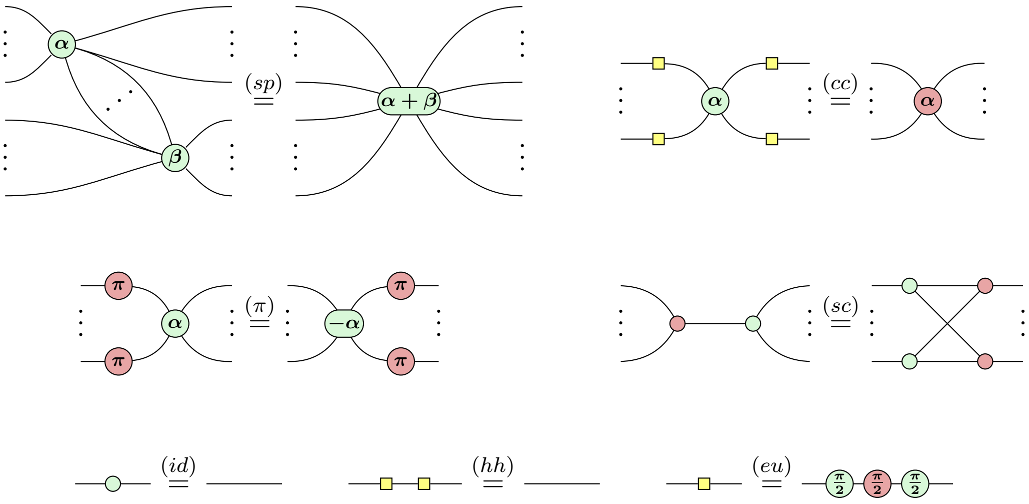
One of the most important rules (that is sometimes not even considered a rule because it is so fundamental to the ZX calculus) is *Only Connectivity Matters (OCM).* OCM states that we can deform any ZX diagram by moving its spiders and wires in almost arbitrary ways. As long as we keep the connectivity between spiders (i.e. which spider is connected to which other spider) and boundary nodes (i.e. the ends of boundary wires must maintain their order) intact, the semantics of the diagram will be preserved [4, Sec. 3.1.3].



(2.33)

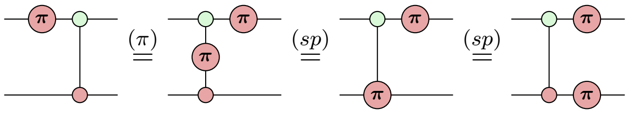
Beyond OCM, the ZX calculus has a complete set of rewrite rules that allow rewriting any two diagrams into each other if they represent the same linear map [20]. This completeness allows us to express any kind of rewrite for a quantum circuit in terms of ZX rewrites, without ever having to compute the underlying linear maps. However, it should be noted that just because it is provably known that any kind of rewrite can be expressed in ZX, it might still be hard to find the right sequence of rewrite rule applications to prove some desired equivalence in ZX.

Since we focus on the Clifford fragment in this work, we present the following rule set which is complete for this fragment [4, Sec. 3.2].



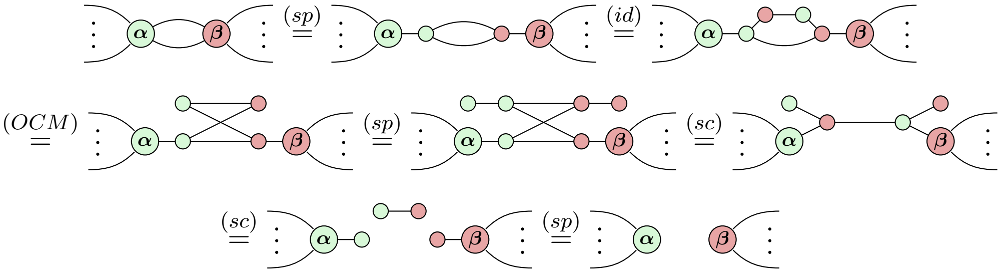
(2.34)

We can apply these rules to prove concrete facts about circuits, for example that an X flip can propagate through a CNOT gate to become two X flips.



(2.35)

We can also use ZX rewrites to derive new ZX rewrites. For example, the *Hopf rewrite* can be used to remove pairs of wires between spiders of opposing colour.



(2.36)

As a result of the colour change (cc) rule, all of these rewrite rules also apply if the colours of spiders are swapped. There are many more useful rewrites and we refer to [4] and [16] for a more detailed discussion.

2.3 Quantum error correction

Universal quantum computing as described in the previous sections is a powerful tool, as it enables quantum algorithms like Shor's integer factoring algorithm [21], Grover's search algorithm [22], and the rather artificial Deutsch-Jozsa algorithm [23]. These algorithms are proven to have a better asymptotical runtime than any possible classical algorithm.

However, their execution is limited by two main factors at the moment. First, their applicability is limited by the number of qubits available on today's hardware systems. For instance, IBM's largest released system has 1,121 physical qubits [24], which is still several orders of magnitude short from the 20 million physical qubits expected to be required for performing Shor's algorithm on a useful scale [25]. Second, these algorithms are not resistent to the noise levels on today's quantum hardware. With each additional qubit, gate, and even idling time step, real quantum computing hardware is subject to *noise*, that is distortions of the quantum state due to interactions with the environment. Quantum hardware is particularly vulnerable to noise because it makes use of quantum-mechanical phenomena only occurring on the tiniest of scales. Controlling a quantum computer through these phenomena requires immense precision, and even then, the quantum state can be distorted by interactions with the environment like electromagnetic radiation. If one were to actually execute Shor's algorithm for useful problems on sufficiently large hardware, the noise introduced by the sheer number of operations required would distort the final state so much that it becomes impossible to read out the correct result [2, Sec. 4.2].

Quantum hardware is expected to improve vastly in the coming years, both in terms of increased qubit counts, and decreased noise levels [26]. However, it is not expected that noise levels can be decreased to a negligible level in the near, or even in the long term [2, 27]. To apply quantum computing successfully, it is therefore imperative to consider quantum programs that work well even in the presence of noise. Turning quantum programs for noiseless, idealised hardware into quantum programs that are resilient to limited amounts of noise is the main goal of *Fault-Tolerant Quantum Computation (FTQC)*. One of its central methods is *Quantum Error Correction (QEC)* which focuses on detecting errors in a fixed quantum memory [27].

In this section, we will first present a formalisation of noise models and errors following [5, 3, 14]. Our perspective will be circuit-centric, i.e. focused around the assumption that some circuit with error correction is given for us to analyse. We then analyse how we can introduce redundancy into quantum states to make noise detectable. Finally, we introduce the surface code which is our main motivating example throughout this work.

2.3.1 Noise models

In this section, we introduce a model for noise on quantum computers. After motivating why we usually restrict our noise models to Pauli errors, we define our noise models as sets of atomic faults and examine its properties.

Quantum states can be distorted by noise in arbitrary ways. They can be changed in the same way as a unitary gate acting on the state, but noise could also act like a measurement projecting the state onto a subspace [3]. This is unlike noise on classical bit registers, where the number of distortions is limited by the finite number of states that the register could transition to. Quantum states can be distorted in infinitely many different ways, even for single-qubit systems.

For classical data registers, we can model any state distortion as a combination of bitflips on the independent bit registers. If an error correction code is able to correct all components of an error simulatenously, it can correct the full, composed error. A similar statement is true for QEC codes and this allows us to limit our noise analysis to a large, but finite set of Pauli flips:

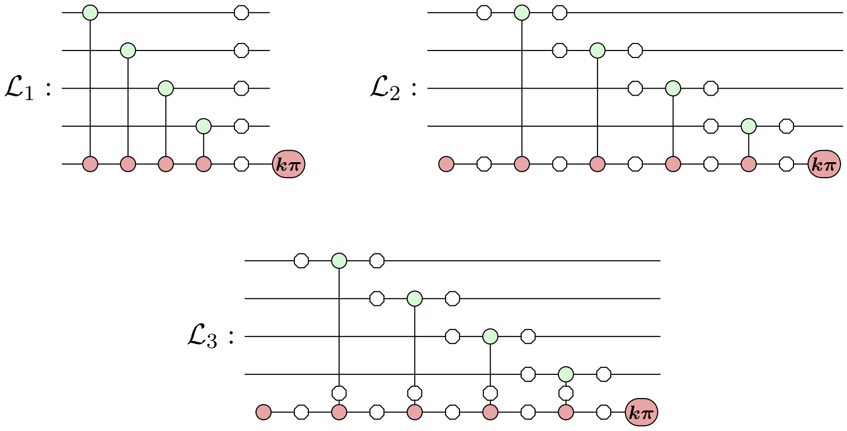
Theorem 2.12 (Pauli flip corrections are enough [27, Cor. 2.5])**.** *If a QEC code can correct all t-qubit Pauli faults, then it can correct any weight t-qubit error.*

Baked into this theorem, we can also find the assumption that there all faults affecting at most some specific number of qubits can be corrected. We call noise models with this property *adversarial noise models* because for this property to hold, even the worst fault affecting at most the specified number of qubits must be correctable. Adversarial noise models stand in contrast to *stochastic noise models* which instead assign probabilities to the potential faults and will usually require us to assess the probability of a randomly drawn fault to be correctable. For the purpose of this thesis, we focus on adversarial noise models, but we briefly revisit this choice in Chapter 5.

We have now seen what shapes individual errors can take but we have not yet discussed where they can take place. Following the formalisation of Rodatz et al. [13], we address this matter abstractly by introducing so-called *fault locations.*

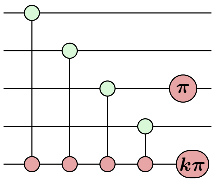
Definition 2.13 (Fault location set)**.** *For a given ZX diagram D, a fault location set $\mathcal{L} \subseteq E(D)$ contains all locations where Pauli flips can arise from noise.*

This definition intentionally leaves the concrete fault locations in the set up for the model to define. This allows the noise model to use *space-time locations* in a concrete circuit, so faults cannot only occur at the end of a circuit but between each time step of the circuit. Take for example the following circuit of four CNOTs. We have annotated this circuit in four different fault location sets $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$:



(2.37)

Each fault location can be instantiated with Pauli flips and $\overline{\mathcal{P}}^{\mathcal{L}}$ contains all of these assignments. We read an assignment to the fault locations as inserting the specified Pauli flips into the diagram at the corresponding locations. For example, the Pauli flip assignment of $\mathcal{L}_1$ that maps the fault location on the third wire to an X flip and all others to the identity I produces the following circuit.



(2.38)

This brings us to our central definition for noise models:

Definition 2.14 (Atomic fault set)**.** *Let $\mathcal{L}$ be a location set for some ZX diagram D. An* atomic fault set *$\mathcal{F} \subseteq \overline{\mathcal{P}}^{\mathcal{L}}$ contains all atomic faults that can occur in the circuit according to the chosen noise model.*

We also refer to atomic fault sets as *noise models* because they specify all relevant information about the noise. The remaining definitions in this subsection are built on top of this structure but they do not add any new data.

Atomic fault sets allow us to define what a "unit" fault is and we can express common adversarial noise models through them. For example, the *circuit-level noise model* [5, 13] can be expressed as an atomic fault set $\mathcal{F}_2 \subseteq \mathcal{P}^{\mathcal{L}_2}$ where $\mathcal{L}_2$ is the location set from Eq. (2.37). As the fault locations in $\mathcal{L}_2$ include all affected wires of the states, gates, and measurements, $\mathcal{F}_2$ then contains all Pauli flips for the individual fault locations (which correspond to single-qubit noise, and also include measurement flips), as well as combinations of Pauli flips on the output wires of the CNOT gates.

Rodatz, Kissinger, and Poór argue in their *"Fault Tolerance by Construction"* paper that many adversarial noise models are subsumed by the *edge flip noise model* which we present in the following.

Definition 2.15 (Edge flip noise model [13])**.** *The* edge flip noise model *for a ZX diagram D allows arbitrary Pauli flips on all edges, i.e.*

$$\mathcal{L} = E(D)\,, \qquad \mathcal{F} = \{(l \mapsto P^{\delta(e,l)}) \mid e \in E(D), P \in \{X, Y, Z\}\}\,. \tag{2.39}$$

Here, $\delta(\cdot,\cdot)$ is the Kronecker delta.

Atomic fault sets have allowed us to define what constitutes a "unit" fault in our noise model. For the final part of this subsection, we formalise how to deal with several faults occurring at the same time.

Definition 2.16 (Fault group, fault)**.** *For an atomic fault set $\mathcal{F} \subseteq \overline{\mathcal{P}}^{\mathcal{L}}$, we define the* Pauli group *$\langle \mathcal{F} \rangle$ as the span of the atomic fault set.*

A fault *$F = F_1 \cdots F_n \in \langle \mathcal{F} \rangle$ can be made up of many atomic faults $F_1, \ldots, F_n \in \mathcal{F}$ and we use the* weight function *$wt(F)$ to refer to the minimum number of atomic faults that compose to F.*

It should be noted that $\langle \mathcal{P} \rangle$ can be a strict subgroup of $\overline{\mathcal{P}}^{\mathcal{L}}$. This is because some assignments of Pauli flips to fault locations might not be realisable in a noise model.

Definition 2.17 (Fault equivalence)**.** *We call two faults $F_1, F_2 \in \langle \mathcal{F} \rangle$ over some fault set $\mathcal{F}$ over a ZX diagram D* equivalent *if their effect on the diagram is the same, i.e.*

$$[\![D^{F_1}]\!] = [\![D^{F_2}]\!]\,. \tag{2.40}$$

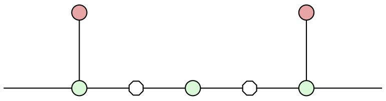
We write $F_1 \equiv F_2$ to denote that F_1 and F_2 are equivalent.

We take this opportunity to discuss the different levels of sameness that faults can achieve:

- **Same representation**: The strongest notion of sameness is if two faults have the same representation. If for some diagram, we have fault locations $\mathcal{L} = \{l_1, l_2\}$ and the atomic fault set is $\mathcal{F} = \{f_1 = (X, I), f_2 = (I, X), f_3 = (X, X)\}$, then $f_1 \cdot f_2$ has the same representation as itself and $f_2 \cdot f_1$, but it has a different representation than f_3.

- **Same fault**: Two faults in a fault group over a fault set are the same if they are the same element of the fault group, no matter their representation. In the previous example, we have $f_1 \cdot f_2 = f_2 \cdot f_1 = f_3$ and therefore all three faults are the same, even though they are composed of different atomic faults.

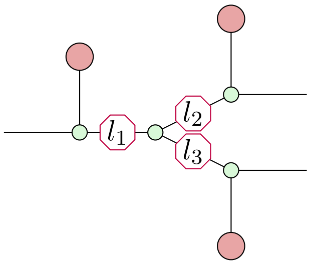
- **Equivalent fault**: Fault equivalence depends on the diagram that the faults lie in. For example, we can have a wire with two fault locations as follows.

  (2.41)

  Having an X fault on the first location is equivalent to having an X fault on the second location because of the π-rule of the ZX calculus. While these two atomic faults are equivalent, they are not the same fault.

  Another example is the following diagram. Here, (X, I, I) and (I, X, X) are not the same fault; but they are equivalent.

  (2.42)

We summarise these levels of equality in the following following hierarchy:

$$\text{same representation} \implies \text{same fault} \implies \text{equivalent fault} \tag{2.43}$$

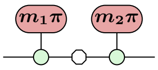
2.3.2 Codes and detectors

So far in this section, we have seen how noise can distort the quantum state, and how we can model such noise in a finite *noise model.* We now turn our attention to the question how we can detect the distortions introduced by noise, and how we can correct it. Researchers have studied this topic for a long time and many ideas to this field were inspired by classical error correction [3]. While the methods in quantum error correction and classical error correction are far apart from each other, the core idea remains the same: Encode the so-called *logical quantum state*, which lives in a 2^k-dimensional Hilbert space, into a *physical quantum state* that lives in a larger, 2^n-dimensional Hilbert spaces. For a linear encoder, the set of encoded states forms a 2^d-dimensional subspace. If we pick our encoder right, this injection into a larger space can add redundancy in such a way that faults with a limited weight can only distort the state so much that we can still identify the undistorted from it.

Quantum computers do not allow us to fully observe the quantum state because observations can collapse superpositions and therefore damage the state irreversibly. However, we can make use of the added redundancy in a different way. Formally, encoded states exhibit parity constraints in the absence of noise:

Definition 2.18 (Detector [5, 12])**.** *A* detector *is a parity constraint on a set of measurement outcomes.*

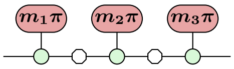
Consider for example the following circuit which is comprised of two measurements with a fault location between them.



(2.44)

In the absence of noise, these measurements should always produce the same result as the first measurement projects the input state to either $|0\rangle$ or $|1\rangle$ and the second measurement is deterministic on Z basis states. $m_1 \oplus m_2$ is a detector of this circuit because in the absence of noise, it has a deterministic outcome ($m_1 \oplus m_2 = 0$). If, however, an X fault occurs in between these measurements, they will produce a different result and the detector is *violated*.

In larger diagrams, there will usually be many different detectors. Consider the following circuit with three measurements



(2.45)

From this circuit, we can form the following three detectors.

$$m_1 \oplus m_2 = 0 \qquad m_2 \oplus m_3 = 0 \qquad m_1 \oplus m_3 = 0 \tag{2.46}$$

However, these detectors are not independent: If any two of these are satisfied, then the third one follows. We call any independent and generating set of detectors for a circuit and noise model a *detector basis*.

Definition 2.19 (Detector basis)**.** *We call a set of detectors S for a diagram D a* detector basis *if its elements are independent and S generates all detectors of D.*

For this concrete example, $S = \{m_1 \oplus m_2 = 0, m_2 \oplus m_3 = 0\}$ is a detector basis (and so is every other pair of these detectors).

To analyse which faults can be detected, it is often useful to create a *Tanner graph* which visualises the relation between atomic faults and detectors

Definition 2.20 (Tanner graph, syndrome)**.** *For a given fault set $\mathcal{F}$ and detector basis S for a diagram C, the* Tanner graph *specifies which atomic faults violate which detectors, i.e.*

$$\begin{gathered} T = (V_T, E_T) \\ V_T = \mathcal{F} \cup \mathcal{D} \\ E_T = \{\{F, d\} \mid F \in \mathcal{F}, d \in \mathcal{D}, F \textit{ violates } d\} \end{gathered} \tag{2.47}$$

We use $syn_T : \langle\mathcal{F}\rangle \to \mathcal{P}(D)$ to refer to the syndrome *of a fault, that is, $syn_T(F)$ is the set of detectors violated by an atomic fault F. For nonatomic faults, we consider a* *detector triggered if the number of atomic faults triggering it is odd:*

$$syn_T(F_1 \cdots F_n) = \{d \in D \mid |\{i \mid d \in syn_T(F_i)\}| \mod 2 = 1\} \tag{2.48}$$

It should be noted that generally, not all syndromes can actually be realised by a fault, so $\mathrm{Im}(syn_F) \subseteq \mathcal{P}(D)$ may be a strict subset.

Note that the Tanner graph only visualises the detector violations of individual, atomic faults. If two atomic faults $F_1, F_2 \in \mathcal{F}$ violate a specific detector $d \in \mathcal{D}$, then the composed fault F_1F_2 does not violate d because even numbers of parity flips cancel out. We say that a fault $F \in \langle\mathcal{F}\rangle$ is *undetectable* if it violates none of the detectors. The weights of undetectable errors give us the following characterisation of QEC codes and circuits:

Definition 2.21 (Distance)**.** *For a given Tanner graph T for a noise model $\mathcal{F}$ for a diagram D, the* distance *is the weight of the smallest non-trivial, undetectable fault, i.e.*

$$\mathrm{dist}_{\mathcal{F}}(D) = \min \left\{\mathrm{wt}_{\mathcal{F}}(F) \mid F \in \langle\mathcal{F}\rangle, \mathrm{wt}_{\mathcal{F}}(F) > 0, syn_T(F) = \emptyset\right\} . \tag{2.49}$$

Notably, the distance of a circuit does not depend on the choice of a detector basis [5, Cor. 8]. The distance is often used as quality metric for QEC codes. Crucially, it not only allows us to reason about the detectability of errors, but also about their correctability:

Proposition 2.22 (Correction by distance [5, p. 8])**.** *In a ZX diagram D with a noise model $\mathcal{F}$, all faults $F \in \langle\mathcal{F}\rangle$ of weight $\mathrm{wt}_{\mathcal{F}}(F) \leq (\mathrm{dist}_{\mathcal{F}}(D) - 1)/2$ are correctable.*

We generally refer to a fault $F \in \langle\mathcal{F}\rangle$ as *correctable* if there is no fault $F' \in \langle\mathcal{F}\rangle$ with the same syndrome as F such that $F' \not\equiv F$ and $\mathrm{wt}_{\mathcal{F}}(F') \leq \mathrm{wt}_{\mathcal{F}}(F)$. This condition ensures that F is the unique (up to equivalence) minimum-weight fault producing its syndrome. If we assume that a correctable fault has occurred, we can then correct it by finding the minimum-weight fault for the measured syndrome, and then applying an operator to the quantum state that is equivalent to this fault[1].

The distance is an important metric for the quality of a code. Codes that encode

[1] In our space-time noise models, faults can occur anywhere in the diagram, but after measuring the syndrome, we are already at the end of the circuit and cannot correct faults in the past. However, we can always "push" Pauli faults to the end of Clifford circuit to compute an equivalent, but applicable correction. This pushing operation is efficiently implementable.

k-qubit logical states onto n-qubit physical states such that they have a distance of d are therefore often referred to as $[\![n, k, d]\!]$ codes [4].

2.3.3 The surface code

We have now seen a formalisation of faults, and we presented how to reason about their detectability and correctability. There is a rich literature of different QEC codes that encode states redundantly in different ways and these approaches vary in their distance, physical qubit counts, and other features. In this subsection, we discuss the *surface code* [28] and how it fits into our error-detector model.

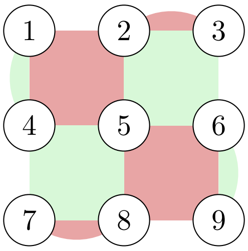
The surface code is a stabiliser code and as such, it specifies the redundant encoding of logical in physical qubits through a set of stabiliser generators. Although we will not go into the details of stabiliser theory here, we will briefly introduce stabilisers to understand how the code is used.

Definition 2.23 (Stabiliser)**.** *A ZX diagram D is a* stabiliser *of a state $|\psi\rangle$ if $|\psi\rangle$ is a $+1$-eigenvector of D, i.e.*

$$D\,|\psi\rangle = |\psi\rangle\ . \tag{2.50}$$

So far with the ZX calculus, we have ignored non-zero scalars for diagrams and diagram equalities. For the stabiliser definition, however, the sign of the eigenvalue is important. It should be noted that that ZX calculus is in principle well capable of handling scalar-accurate rewrites and we refer to [4, Sec. 3.6.2] for such a treatment.

The stabilisers of state form a group which is generated by a finite number of generators and in turn, the Clifford state stabilising a stabiliser group of sufficiently many generators is uniquely fixed [4, Cor. 6.3.1]. Similarly, stabiliser codes fix the codespace though a set of generators that allow just enough freedom for a logical state. For the surface code, the choice of these generators is motivated by the following geometric shape.

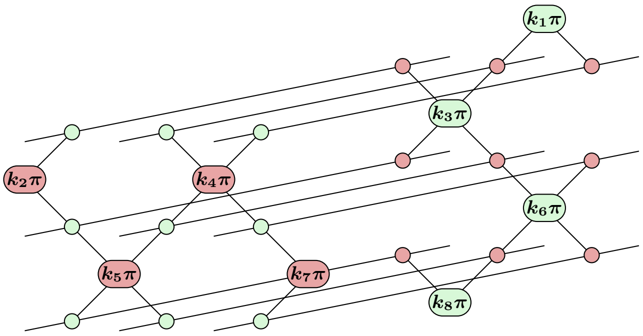


(2.51)

The white circles in this shape correspond to qubits and each of the coloured faces correspond to a stabilisers of the encoded states. Every red face corresponds to an operator that applies X gates to all of the adjacent qubits and symmetrically, every green face corresponds to an operator that applies Z gates to all adjacent qubits. For example, the top-right red face and the bottom-left green face correspond to the following stabilisers:

$$\begin{array}{ccccccccccccccccc} I & \otimes & X & \otimes & X & \otimes & I & \otimes & I & \otimes & I & \otimes & I & \otimes & I & \otimes & I \\ I & \otimes & I & \otimes & I & \otimes & Z & \otimes & Z & \otimes & I & \otimes & Z & \otimes & Z & \otimes & I \end{array} \tag{2.52}$$

The stabiliser generators characterise the redundancy in the codespace and we can formulate them as measurements to get parity constraints:

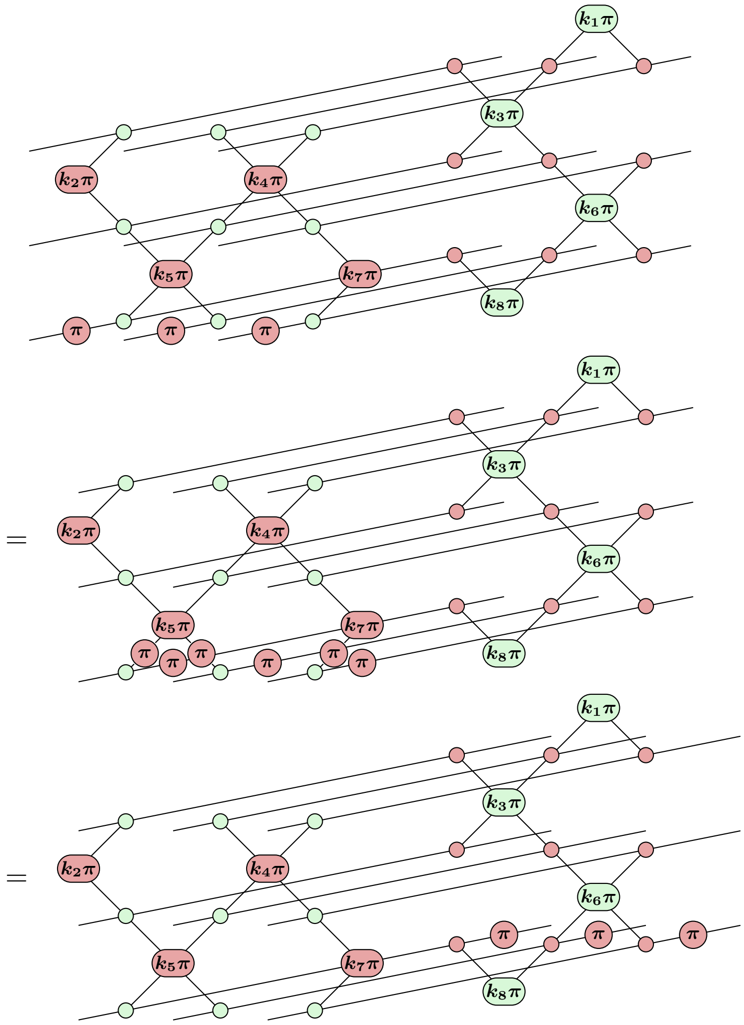


(2.53)

If a state from the codespace of the surface code is used as an input to this diagram, all of these Pauli measurements will be zero. These eight parity constraints form the detector basis S of this surface code.

By design, the surface code only uses stabilisers that are either based only on X flips or only on Z flips, which makes it an instance of Calderbank-Shor-Steane (CSS) codes. For such codes, X flips will only be detected by X Pauli measurements and Z flips will only be detected by Z Pauli measurements but never the other way round. Therefore, we can think of CSS codes as two separate codes for the Z and X components of faults. We can make practical use of this separation when thinking about the detectability or correctability of faults by separating the fault into its X and Z flips, decomposing Y flips to Z and X.

The surface code we discussed had a distance of 3 and we can verify that there is indeed an undetectable error with three Pauli flips by considering any line of Z or X Pauli flips on these qubits:



(2.54)

Given the availability of sufficiently many physical qubits, surface codes can be constructed to have arbitrary distances. Physical implementations of the surface codes benefit from the local connectivity of its circuits, which avoids drastic increases to the gate count for routing gates between distant qubits [4]. Surface codes also have a high *threshold*, a metric which characterises how low the physical error rates in a quantum need to be for the QEC code to improve the noise level [3]. And lastly, surface codes admit *matchable decoders*, which we will discuss in further detail in Section 2.5.

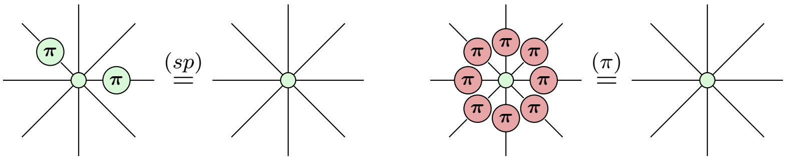
2.4 ZX perspective on fault tolerance

In the previous sections, we have seen how noise poses a significant challenge to quantum computing and we have seen how QEC enables us to detect and correct some of this noise.

We have presented these concepts with their ZX calculus formalisation but we could have just as well presented them in the circuit-based model. In fact, these ideas are older than the ZX calculus itself, and their original presentation was based on the linear algebra or circuit-based quantum computing formalism. In this section, we present *Pauli webs* and *fault-equivalent rewrites*, two concepts in the fault tolerance regime that make more explicit use of the diagram structure of the ZX calculus.

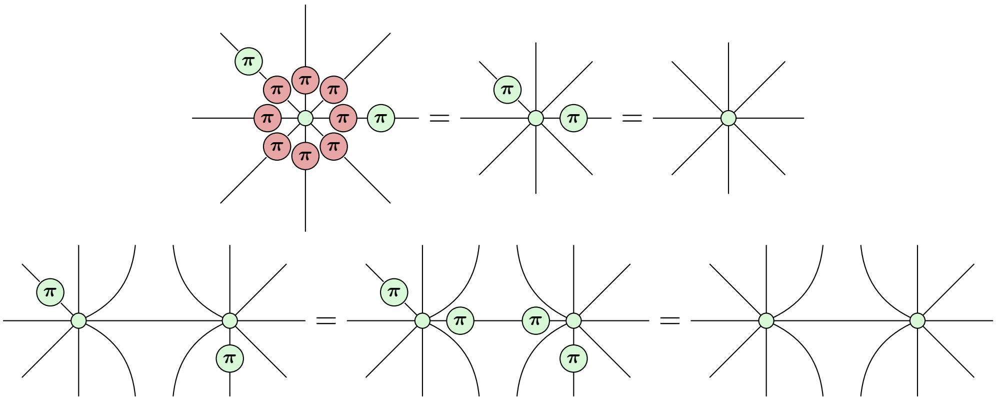
2.4.1 Pauli webs

In their paper *"Unifying flavors of fault-tolerance with the ZX calculus"*, Bombín et al. present a method to express stabilisers and detectors for Clifford ZX diagrams as a decoration to the wires of the diagrams [15]. Pauli webs are motivated by the following observation regarding the stabilisers of single spiders: Phaseless Z spiders stabilise any pair of Z flips on their wires and they also stabilise X flips on all of their wires. Symmetrically, phaseless X spiders stabilise any pair of X flips and the combination of Z flips on all wires:



(2.55)

These stabilisers are composeable in several ways. First, we can combine several stabilisers of the same diagram to get another stabiliser for the diagram. Second, we can wire up multiple spiders with stabilisers and if they cover the connecting wires with edge flips in the same way, then we can read off a stabiliser of the composed diagram:



(2.56)

Pauli webs express this compositionality of stabilisers as a decoration to the edges of

ZX diagrams. We formally define them as follows.

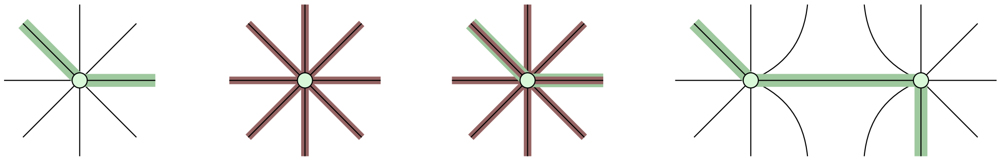
Definition 2.24 (Pauli web [15])**.** *A* Pauli web *is a decoration for the wires of a diagram. More specifically, $P : E(D) \to \overline{\mathcal{P}}$ is a Pauli web of a ZX diagram D if it satisfies the following conditions:*

- *$k\pi$-spiders have both*
  - *an even number of legs in their own colour*
  - *all or no legs in the opposite colour*
- *$\pm\frac{\pi}{2}$-spiders have either*
  - *an even number of legs in their own colour and no legs in the opposite colour*
  - *an odd number of legs in their own colour and all legs in the opposite colour*

We call P trivial *if it does not colour any edges in the diagram and we call it a* detecting Pauli web *or* detecting region *when none of D's boundary wires are coloured.*

Furthermore, Pauli webs are sometimes also classified as "stabilising", "co-stabilising" or "logical" Pauli webs [13]. We will not make this distinction and refer to stabilisers corresponding to Pauli webs as demonstrated by the examples in this section.

For this definition, we should clarify that we consider red and green to be opposite colours. However, they are not mutually exclusive and edges can have no colour too. We can use Pauli webs to represent the stabilisers from Eqs. (2.55) and (2.56) as follows:



(2.57)

Since Pauli webs correspond to stabilisers and stabilisers are closed under multiplication the (element-wise) products of Pauli webs are Pauli webs too.

So far with the ZX calculus, we have ignored non-zero scalar for diagram equalities. However in stabiliser theory, the sign of stabilisers can be a relevant property. In our application of this property, we can limit the need for sign considerations to the commutation and anticommutation of Pauli flips and Pauli webs.

Definition 2.25 (Commutation and anticommutation)**.**

- *Two Pauli flips $P_1, P_2 \in \overline{\mathcal{P}}$* commute *($P_1P_2 = P_2P_1$) if they are identical ($P_1 = P_2$) or either of them is the identity ($P_1 = I$ or $P_2 = I$). Otherwise, they* anticommute *($P_1P_2 = -P_2P_1$).*
- *Two Pauli webs P_1, P_2 of the same ZX diagram D commute if the numer of edges for which their Pauli flips anticommute is even, i.e.*

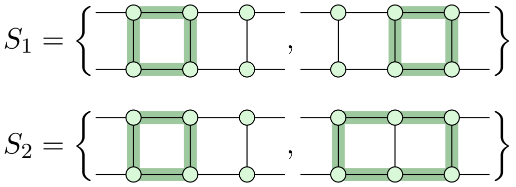
$$|\{e \in E(D) \mid P_1(e)P_2(e) = -P_2(e)P_1(e)\}| \mod 2 = 0\,. \tag{2.58}$$

Otherwise, they anticommute.

Detecting regions are of particular interest in the discussion of error-detector models because of their relation to detectors. More specifically, detecting regions visualise restrictions on the measurement outcomes in ZX diagrams similar to the parity constraints that we called detectors in the previous section [15, Sec. 3]. If these parity constaints are violated, we consider a fault detectable. Following Rodatz et al. [13, Sec. 5.2], we thus model fault detection for ZX diagrams under edge flip noise as follows:

Proposition 2.26 (Fault detection for Pauli webs [13, Def. 5.7])**.** *A fault $F \in \langle \mathcal{F} \rangle$ violates a detecting Pauli web P of a ZX diagram D if its edge flips anticommute with P on oddly many edges.*

Similar to Definition 2.19, we can find a detector basis of all detecting regions in a diagram. For example, the following two sets S_1, S_2 are both valid detector bases because they are independent and they generate all possible detectors:



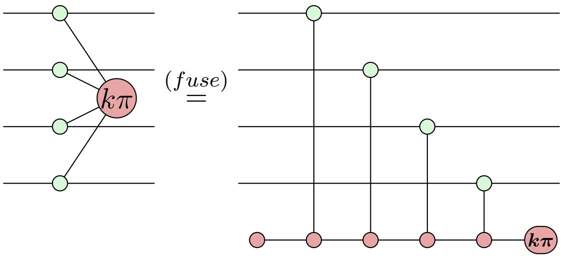
(2.59)

The visual representation of detectors is therefore also visual representation of detectability: By looking at the detector bases in the above equation, we can see that no single Z flip is detectable in this diagram, and single X and Y flips are only detectable on the internal wires of the diagram.

2.4.2 Fault equivalence

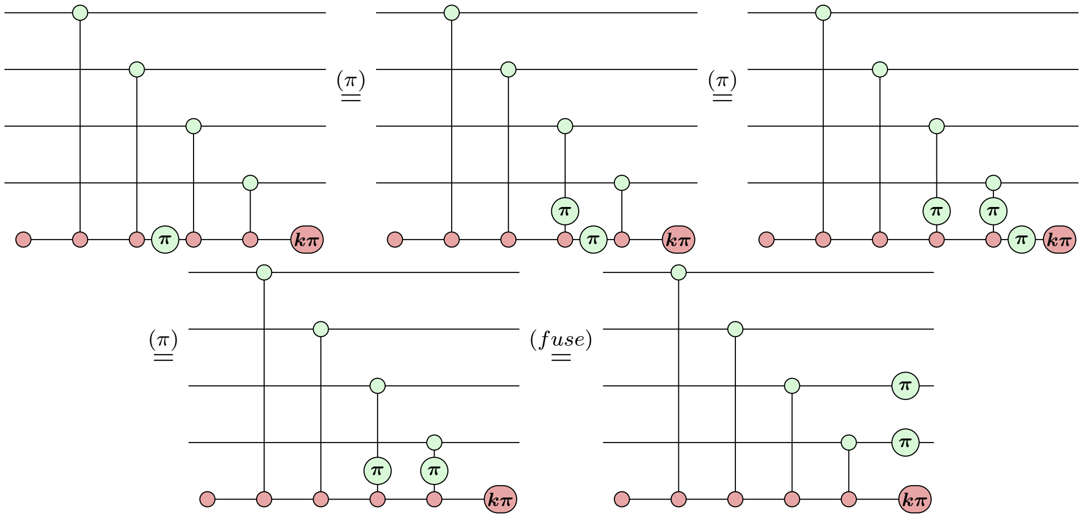
In this section, we will explore another notion of fault tolerance in the ZX realm, namely *fault-equivalent rewrites*, which allows us to preserve the noise model in ZX rewrites. We will first motivate why standard rewrites do not preserve noise models and then introduce the system of rewrites presented by Rodatz et al. [14, 13].

Rewriting diagrams and circuits is a key feature of the ZX calculus and by our definition of ZX rewrites (Definition 2.11), any application of a ZX rewrite preserves the semantics of the diagram. This means that the two diagrams produce the same result under the same input, assuming they can be executed without error. In the context of noisy quantum computing, however, this assumption is no longer justified as we might introduce or remove potential error locations. Consider for example the following sequence of ZX rewrites which turns a $ZZZZ$ measurement into a sequence of CNOTs, a $|0\rangle$ state initialisation and a destructive, single-qubit Z measurement.



(2.60)

These rewrite steps show that the diagrams on the left-hand side and on the right-hand side are semantically equivalent. However, noise can have different effects on these diagrams under an edge-flip noise model, and even a circuit-level noise model. In the RHS diagram, a single Z flip on the ancilla qubit after the second CNOT gate will propagate through the remaining CNOT gates to become three Z flips: two on the lower two data qubits and one on the ancilla qubit.



(2.61)

No single edge flip on the LHS of Eq. (2.60) has the same effect on this diagram. We can therefore conclude the applied ZX rewrites have changed the noise model. This is alarming because it can destroy the error-correcting capabilities of a QEC code. More specifically, if the above implementation was part of a bigger diagram with circuit distance $d = 2$, and the $I \otimes I \otimes Z \otimes Z$ fault was undetectable, then our rewrite would have decreased the circuit distance to one because a single edge flip was already undetectable. This issue motivates the notion of *fault-equivalent* rewrites.

Definition 2.27 (Fault-equivalent rewrite [13])**.** *Let D_1, D_2 be equivalent ZX diagrams and let $\mathcal{F}_1, \mathcal{F}_2$ be noise models for D_1, D_2. We call D_1 under $\mathcal{F}_1$* fault-equivalent *to $\mathcal{D}_2$ under $\mathcal{F}_2$ if every undetectable fault $F_1 \in \langle \mathcal{F}_1 \rangle$, there exists a fault $F_2 \in \langle \mathcal{F}_2 \rangle$ such that $D_1^{F_1} = D_2^{F_2}$ and $\mathrm{wt}_{\mathcal{F}_2}(F_2) \leq \mathrm{wt}_{\mathcal{F}_1}(F_1)$, and vice versa.*

We denote fault equivalence with $(D_1, \mathcal{F}_1) \hat{=} (D_2, \mathcal{F}_2)$, or $D_1 \hat{=} D_2$ if the noise models are clear from context.

Fault-equivalent rewrites ensure that the problem from our example in Eq. (2.60) cannot occur by ensuring every undetectable fault must have an equivalent fault of at most the same weight in the other diagram. They are a specialisation of so-called *distance-preserving rewrites* which was presented in an earlier work [14]. As such, fault-equivalent rewrites have the distance preservation property as well:

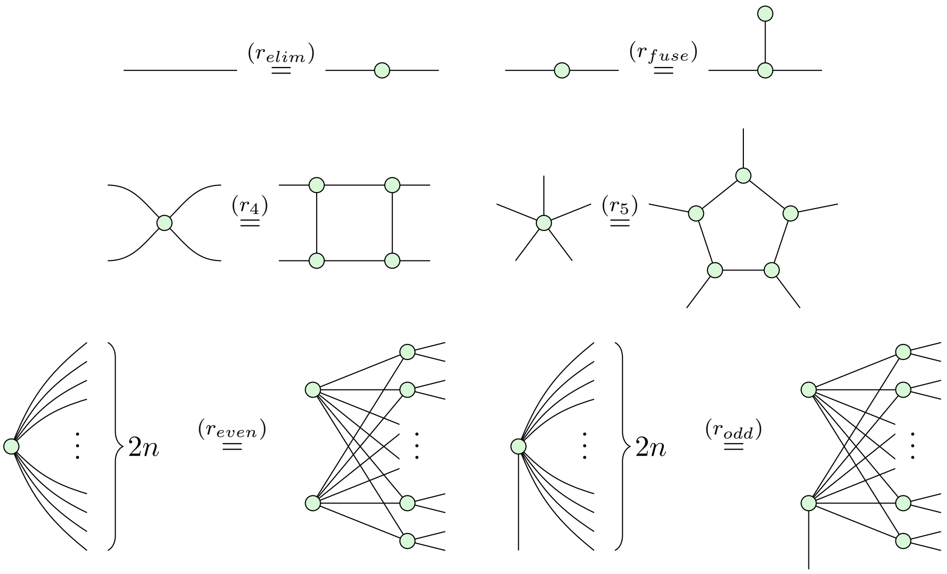
Proposition 2.28 (Fault-equivalent rewrites preserve the distance [13, Cor. 3.13])**.** *If two*

ZX diagrams D_1, D_2 under noise models $\mathcal{F}_1, \mathcal{F}_2$ are fault-equivalent, then $\mathrm{dist}_{\mathcal{F}_1}(D_1) = \mathrm{dist}_{\mathcal{F}_2}(D_2)$.

Furthermore, fault equivalence is compositional, that is the parallel and sequential composition of fault-equivalent diagrams are fault-equivalent too and fault equivalence is a transitive property [13, Prop. 3.11 - 12].

While Rodatz et al. present fault equivalence as a general concept, applicable to arbitrary adversarial noise models and circuits, their usefulness really comes to shine on ZX diagrams under edge flip noise models. The authors demonstrate that arbitrary circuits under arbitrary noise models can be represented fault-equivalently under subsets of the edge-flip noise model. Together, compositionality and the locality of edge-flip noise ensure that we can apply fault-equivalent rewrites to subdiagrams of a larger diagram to get a fault-equivalent rewrite of the larger diagram.

Rodatz et al. present the following system of fault-equivalent rewrites in [14]. The edge-flip noise model is assumed for all of these ZX diagrams.


$\overset{(r_{elim})}{=}$
$\overset{(r_{fuse})}{=}$
$\overset{(r_4)}{=}$
$\overset{(r_5)}{=}$
$2n$
$\overset{(r_{even})}{=}$
$2n$
$\overset{(r_{odd})}{=}$

(2.62)

While this system is not known to be complete, it is sufficient for decomposing arbitrary, phaseless ZX diagrams into phaseless ZX diagrams where all spiders have at most three legs. For phaseless diagrams like surface code implementations, this property is sufficient for extracting a CNOT circuit from the diagram, that is a circuit with only state preparations, CNOT gates and measurements [14]. A complete set of fault-equivalent rewrites is currently being developed in a parallel work [29].

2.5 Matching decoders

When looking at the correcting capabilities of QEC codes so far, we have mostly focused on which faults we can be detected and corrected in principle. In these considerations, we have neglected the feasibility for implementing the *decoder*, that is the component that calculates which correction to apply based on the syndrome. In this section, we discuss the hardness of the decoding problem and we present *matching decoders* which can solve the decoding problem efficiently for a specific class of QEC codes. Furthermore, we characterise how *matchability*, the property that enables the use of matching decoders, can be expressed on ZX diagrams.

Decoherence, one of the main sources of noise, disturbs the quantum state perpetually. QEC procedures must therefore be run at a high frequency to avoid accumulating noise to an irreversible degree [7, 8]. For this to work, the error correction procedure must run efficiently. However, Hsieh and Le Gall showed in 2011 that the decoding problem for QEC codes is generally $\mathcal{NP}$-hard [6]. In spite of enormous research efforts on the "$\mathcal{P}$ vs $\mathcal{NP}$" problem, no efficient algorithm was ever found for $\mathcal{NP}$-hard problems, which suggests that finding an efficient algorithm for the decoding problem too will be unlikely for the forseeable future [30]. The hardness of the decoding problem seems prohibitive for the successful application of QEC at first.

However, in their *"Topological quantum memory"* publication from 2002 [31], Dennis et al. discuss a class of QEC codes whose error-detector models exhibit a specific structure admitting efficient decoding. We characterise this structure in the following definition.

Definition 2.29 (Matchability of a Tanner graph)**.** *We call a Tanner graph T for a noise model $\mathcal{F}$* matchable *if every atomic fault violates at most two detectors, i.e.*

$$\forall F \in \mathcal{F}: \quad |syn_T(F)| \leq 2\,. \tag{2.63}$$

We also call individual atomic faults matchable *if they violate at most two detectors.*

In the literature, this property is occasionally referred to as *graph-like*[2] because it

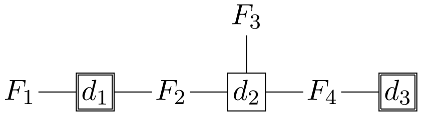
[2] We decided to give this property a new name because "graph-like" is an overloaded term in the context of the ZX calculus and QEC. For example, it refers to a specific diagram structure used in ZX diagram simplification routines [32]. Furthermore, many different types graphs are involved in ZX-based QEC, from

allows constructing a *decoding graph*, an equivalent formulation of error-detector models. For a given syndrome and decoding graph, the decoding problem can be reduced to to the *Minimum Weight Perfect Matching (MWPM)* problem which is efficiently solvable. The surface code, which we presented as an example QEC code in the previous section, is one of the most widely studied codes with this property.

2.5.1 The MWPM decoder

In this subsection, we demonstrate how to reduce the decoding problem for matchable error-detector models to the *Minimum-Weight Perfect Matching (MWPM)* problem. This reduction is comprised of two stages: constructing a so-called matching graph and then constructing a syndrome graph, which will is an MWPM instance. We refer to decoding algorithms that make use of this reduction as *matching decoders*.

In Section 2.3, we introduced the Tanner graph as a representation of a error-detector models. Consider the following example of a Tanner graph T which has four atomic faults and three detectors, and suppose that we want to find the minimum-weight fault for the syndrome $s = \{d_1, d_3\}$.



(2.64)

We know that d_3 needs to be triggered and F_4 is the only atomic fault triggering this detector, so F_4 must definitely be part of the fault. But F_4 also triggers d_2, which is not part of the syndrome. Therefore, we need to add another atomic fault to cancel out the violation of d_2. We can now pick between F_3, which would just turn off d_2, and F_2, which turns off d_2 and also violates d_1. As F_4F_2 already produces the correct syndrome and F_4F_3 does not, we can conclude that F_4F_2 is a minimum-weight decoding of the given syndrome s.

Observe that while reasoning about this decoding, we “walked” through the Tanner graph starting at one of the triggered syndrome bits and ending at another one. In each step where we picked an atomic fault, we walked over all of its edges to get to another detector node. Walking into a detector node first triggered it but walking through it

ZX diagrams over Tanner graphs to surface code visalisations. The term “matchability” aims to point out that something is decodable using matching decoders.

un-triggered it again. The intuition that atomic faults form edges and paths between detectors is formalised in *matching graphs*, which we define as follows.

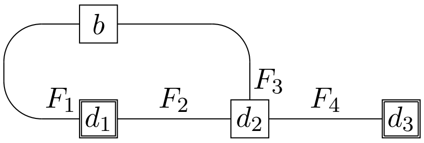
Definition 2.30 (Matching graph [9])**.** *Let T be a matchable Tanner graph for some noise model $\mathcal{F}$ and detector set S. We define the* matching graph $G(T)$ *of this Tanner graph as follows.*

$$\begin{aligned} G(T) &:= (V_{G(T)}, E_{G(T)}) \\ V_{G(T)} &= S \uplus \{b\} \\ E_{G(T)} &= \{e(F) \mid F \in \mathcal{F}\} \end{aligned} \tag{2.65}$$

$$e(F) = \begin{cases} syn_T(F) & \text{if } |syn_T(F)| = 2 \\ syn_T(F) \cup \{b\} & \text{if } |syn_T(F)| = 1 \end{cases}$$

In the literature, matching graphs are sometimes also referred to as *decoding graphs*. Note that matching graphs introduce a boundary node b which is used in edges corresponding to faults that only trigger a single detector. This extra node has no physical meaning, it merely allows us to represent all faults as edges betweem two nodes. We can also observe here why matchable error-detector models are sometimes called "graph-like" in the literature: If there were any atomic faults triggering more than two detectors, they would turn into hyperedges in the matching graph.

For the example Tanner graph T from Eq. (2.64), the matching graph $G(T)$ has the following structure. Here, we have labelled the edges with the corresponding faults.



(2.66)

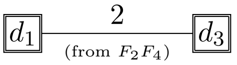
Matching graphs are equivalent formulations of the error-detector model. For a given syndrome, we can use the matching graph to construct an MWPM instance as follows.

Definition 2.31 (Syndrome graph)**.** *Let $G(T)$ be a matching graph for a matchable Tanner graph $\mathcal{T}$ of a noise model $\mathcal{F}$ and detector basis S. For any syndrome $s \subseteq S$, we define* *the* syndrome graph $V(T,s)$ *as the following weighted graph.*

$$\begin{aligned} V(T,s) &:= (V_{V(T,s)}, E_{V(T,S)}, w_{V(T,s)}) \\ V_{V(T,s)} &= \begin{cases} s & \textit{if } |s| \textit{ is even} \\ \\ s \cup \{b\} & \textit{if } |s| \textit{ is odd} \end{cases} \\ E_{V(T,s)} &= V^2_{V(T,s)} \\ w_{V(T,s)}(\{u,v\}) &= \textit{shortest-path-weight}_{G(T)}(u,v) \end{aligned} \tag{2.67}$$

The nodes of this graph are the violated detectors s and if $|s|$ is odd, we also take the boundary node b of $G(T)$ as a node. $V(T,s)$ is fully connected and each edge takes weight of the shortest path between the corresponding nodes in $G(T)$.

For the running example, the syndrome graph $V(T,s)$ is the following one.



(2.68)

Now selecting an edge in this syndrome graph is equivalent selecting shortest paths in the matching graphs or fault chains in the Tanner graph. In fact, picking a minimum-weight set of edges from the syndrome graph such that each node is covered by exactly one selected edge corresponds to the minimum-weight solution to the decoding problem:

Proposition 2.32 (MWPM decoding [9])**.** *Let T be the Tanner graph for a noise model $\mathcal{F}$ and a detector basis S. Then for any syndrome $s \subseteq S$, the minimum-weight perfect matching of $V(T,s)$ induces a minimum-weight decoding of s in T.*

This construction allows us to reduce the decoding problem for matchable error-detector models to an MWPM instance. The MWPM instance can then be solved efficiently using Edmond's algorithm [33]. We summarize this reduction in Algorithm 1.

MWPM decoders for QEC codes have been studied thoroughly in the literature and there exist numerous implementations optimising the runtime of the algorithm [9, 8, 34, 35]. Several implementations have demonstrated decoding millions of errors per second for matchable codes at useful distances [8, 34]. This makes MWPM decoders feasible for use in practical quantum computing [8, 7].

Algorithm 1: MWPM decoder, adversarial variant [9, 11]

```
Data: Matchable error-detector model $T = (V_T, E_T)$, syndrome $s \subseteq V_T$
Result: Minimum-weight fault $F$ such that $syn_T(F) = s$
compute $G(T)$ ;                                   /* build matching graph */
compute $V(T, s)$ ;                                /* build syndrome graph */
$\{\{u_1, v_1\}, \ldots, \{u_n, v_n\}\} \leftarrow \mathrm{MWPM}(V(T, s))$ ; /* solve with Edmond's algo */
for  $i = 1..n$ do
    find shortest path $E_i \subseteq E_{G(T)}$ between $u_i$ and $v_i$;
    $F_i \leftarrow \prod_{\{u,v\} \in E_i} e^{-1}(\{u, v\})$ ;        /* recover fault from path */
end
$F \leftarrow \prod_{i=1}^{n} F_i$ ;                              /* return combined fault */
```

2.5.2 ZX perspective on matchability

We have seen in the previous subsection that MWPM decoders rely on the matchability property of error-detector models to provide an efficient decoding technique. Definition 2.29 characterised this property for Tanner graphs. In this subsection, we specialise matchability to ZX diagrams with edge-flip noise models.

Recall that we can express the detectors of a ZX diagrams as detecting Pauli webs, that is, Pauli webs that do not highlight any of the boundary wires. Furthermore, we could express the detector basis for such a noise model as set of detecting Pauli webs. The atomic faults of edge flips noise models are single Pauli edge flips and an edge flip violates a detecting region if it anticommutes with the colouring of the edge. This allows us to express the matchbility property as follows:

Proposition 2.33 (Matchable edge flip noise models)**.** *The error-detector model for the edge flip noise model and a detecting region basis S of a given diagram is matchable if and only if each edge of the diagram is covered nontrivially by at most two detectors $P \in S$, i.e.*

$$\forall e \in E(D) : |\{P \in S \mid P(e) \neq I\}| \leq 2 \tag{2.69}$$

Proof. This equivalence follows immediately from the definitions of edge flip noise models and matchability and from the fact that Pauli web detectors are violated by anticommuting faults. □

In analogy to CSS codes, we can relax this requirement a bit for phaseless ZX diagrams with detector bases where all generating detectors are either green on all coloured wires or red on all coloured wires. Then because Z flips and X flips are detected independently of each other, we can always decompose Y faults into their Z and X components and simplify the matchability criterion as follows:

Proposition 2.34 (Matchable CSS edge flip noise models)**.** *Let T be the error-detector model for the edge flip noise model without Y flips on a phaseless ZX diagram D and a detecting region basis S where all detectors are either all green or all red. T is matchable if and only if each edge of the diagram is covered by at most two green detectors and at most two red detectors, i.e.*

$$\begin{aligned} \forall e \in E(D): \quad & |\{P \in S \mid P(e) = Z\}| \leq 2 \\ & \wedge |\{P \in S \mid P(e) = X\}| \leq 2 \end{aligned} \tag{2.70}$$

Proof. The proof for this corollary is analogous to that of Proposition 2.33. □

2.6 Splitting

In the previous section, we presented matchability as a helpful property of error-detector models that enables efficient decoding through the MWPM decoder. While very practical, matchability is not a common property among QEC codes. *Splitting* is a method introduced by Nicolas Delfosse et al. in 2023 that changes the noise model of unmatchable error-detector models in a way that ensures matchability.

In their original presentation, Delfosse et al. classify atomic faults[3] into *primitive* and *non-primitive* faults.

Definition 2.35 (Primitive fault)**.** *Let T be a Tanner graph for a noise model $\mathcal{F}$. We call an atomic fault $F \in \mathcal{F}$* primitive w.r.t. *T if it satisfies either of the following conditions:*

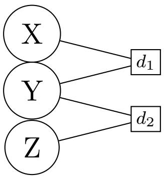
- $|syn_T(F)| = 1$*, i.e. the atomic fault triggers exactly one detector in T or*
- $|syn_T(F)| = 2$ *and there are* ***no*** *two atomic faults $F_1, F_2 \in \mathcal{F}$ with* $|syn_T(F_1)| = |syn_T(F_2)| = 1$ *such that* $F_1 F_2 \equiv F$*.*

[3] The *"Splitting decoders for correcting hypergraph faults"* paper [11] uses a different, and slightly inconsistent terminology. What we call an "atomic fault", they refer to as a "fault", and what we call a "fault", they refer to as a "fault configuration". However, these terms seem to be mixed up in a few places like the splitting pseudocode algorithms, so we will stick with our terminology here to give a consistent presentation.

An atomic fault that is not primitive is called non-primitive.

From this definition, it is immediately clear that primitive faults are matchable, and that any atomic fault set $\mathcal{F}$ that has only primitive faults under some Tanner graph T is also matchable. In fact, primitiveness is a stricter property than matchability.

Take for example a noise model that has an X, a Y, and a Z atomic fault, and a Tanner graph with two detectors where the first one detects the X and the Y fault and the second one detects the Y and the Z fault.



(2.71)

This noise model is matchable even though it has a non-primitive fault (Y) and in this case, it is perfectly valid to split it onto the other two faults (X and Z).

Delfosse et al. proposed two algorithms that turn stochastic noise models into stochastic noise models that only have primitive faults. Here, we will look at one of them: *Decoder-based splitting.* Taking an arbitrary noise model as an input, this algorithm considers every non-primitive fault F and uses a MWPM decoder to find the highest-probability fault $F_1 \cdots F_n$ that is composed of only primitive faults and has the same syndrome as F. F is then removed from the noise model and its probability is added to each of the component faults $F_1, \ldots, F_n$. If no such primitive faults $F_1, \ldots, F_n$ can be found, the splitting fails.

For adversarial noise models, this procedure simplifies to finding a minimum-weight decomposition $F \equiv F_1 \cdots F_n$ and removing F from the noise model. The adversarial variant of Decoder-based splitting is summarized in Algorithm 2.

Decoder-based splitting can fail at the splitting step, not finding a syndrome-equivalent composed fault, but also in the decoding process itself. This behaviour is pointed out in the original paper [11], and a pathological example can be found in a recent paper by Grans-Samuelsson et al. [36] where splitting was used on a surface code example where the detector measurement implementation made the Tanner graph unmatchable. The specific conditions for the success of splitting remain an open topic for future work and

Algorithm 2: Decoder-based splitting [11], adversarial variant

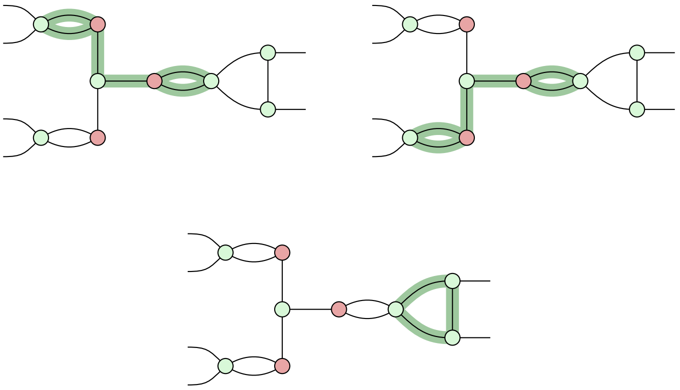
```
Data: Atomic fault set $\mathcal{F}$ and Tanner graph $T$
Result: Atomic fault set $\mathcal{F}''$ where all atomic faults are primitive
$\mathcal{F}' \leftarrow \{F \in \mathcal{F} \mid F$ is primitive$\}$;
$\mathcal{F}'' \leftarrow \mathcal{F}'$ ;                                    /* output accumulator */
for each $F \in \mathcal{F} \setminus \mathcal{F}'$ ;                /* iterate non-primitive faults */
do
    if exists atomic faults $F_1, \ldots, F_n \in \mathcal{F}''$ such that $syn_T(F_1 \cdots F_n) = syn_T(F)$
     then
        continue ;   /* no need to add $F$ to $\mathcal{F}''$, it can be expressed as
          $F_1 \cdots F_n$ */
    else
        fail ;                                           /* $F$ is unsplittable */
    end
end
```

we discuss some incomplete results in Appendix A.

Chapter 3

Matchability preservation

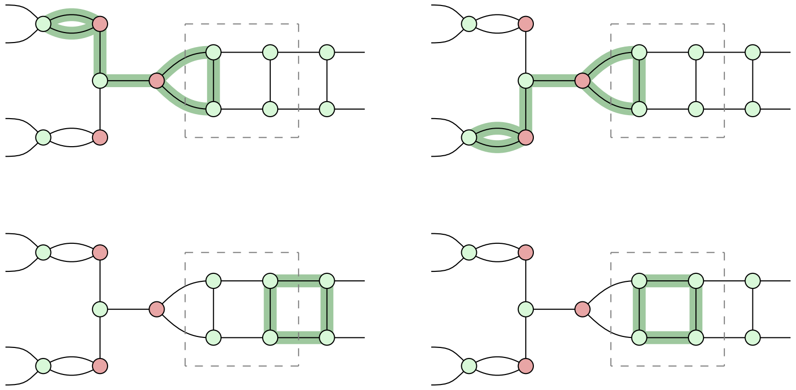
We have seen in Section 2.4 that we can use fault-equivalent rewrites to systematically construct implementations of stabiliser measurements. We have also seen in Section 2.5 that matchable codes are a prominent way to enable efficient decoding. However, these two notions are not necessarily compatible. Take for example the following ZX diagram, which has a detector basis with three elements.



(3.1)

This set of detecting regions covers each wire with at most two generating detectors. Under the edge flip noise model, any atomic fault can therefore trigger at most two detectors, so the error-detector model for this diagram is matchable

If we apply the fault-equivalent r_4 rewrite to the middle-right, green spider, however, we get the following diagram, which has four generating detectors.



(3.2)

In these diagrams, we have highlighted the rewritten part with a dashed box. Apparently, the r_4 rewrite has introduced a new detecting region to our diagram. New detectors are not necessarily problematic, but in this particular case, the rewrite has created a wire (more specifically, the middle vertical wire) that is covered by three green detecting regions. We have therefore lost the matchability property through fault-equivalent rewriting.

This is just a counterexample that should motivate the following formalisation and contributions. We have left a number of questions open here: What if we choose to route the detecting regions differently in the rewritten ZX diagram? Do we run into similar problems when applying the r_4 rewrite in a different context? And how can we avoid running into this problem at all?

In Section 3.1, we first formalise what it means to apply a ZX rewrite in the context of a chosen detector basis and when such rewrites preserve the matchability of the detector basis. Then, in Section 3.2, we show that matchability is always preserved for fault-equivalent rewrites that do not add new detecting regions, namely r_{fuse} and r_{elim}. For four- to six-legged spiders, we demonstrate in Section 3.3 how the fault-equivalent r_4, r_5, and r_6 rewrites preserve matchability for detector bases that cover the spider in a specific shape. In Section 3.4, we demonstrate a trick that allows us to decompose spiders that are not covered by detectors in this shape. Finally, we show in Section 3.5 that the fault-equivalent rewrites r_{even} and r_{odd} for many-legged spiders do not generally preserve matchability, and how we can generalise r_4, r_5 and r_6 to decompose many-legged spiders under matchability preservation.

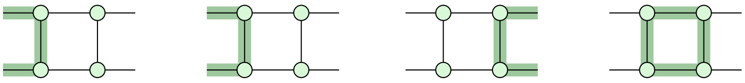
3.1 Detector-aware rewrites

In the introduction to this chapter, we have already rewritten a ZX diagram and compared the detector bases before and after the rewrite. We observed that the application of a rewrite extended the size of the detector basis, and we might even have noticed that some detectors remained largely unchanged. In this section, we formalise what it means to apply a ZX rewrite in such a context and when these rewrites preserve the matchability property.

3.1.1 Defining detector-aware rewrites

ZX rewrites are usually local constructs that are applied to some part of a larger diagram to rewrite it. A ZX rewrite can be specified by providing two equivalent diagrams, and we can apply this rewrite to a larger diagram by replacing an occurrence of one of the rewrite's diagrams with the other one. We call such rewrites local because they only specify the structure of the diagram part to be replaced, and the diagram that this part will be replaced with. Crucially, the rewrite does not depend on global properties of the diagram it is being applied to.

Motivated by this observation, we aim to specify changes to the error-detector model of a ZX diagram in a similar, local fashion. The error-detector model is not directly visible in a ZX diagram or a ZX rewrite, but as we have seen in Section 2.4, we can use Pauli webs to represent detectors as decorations of ZX diagrams. However, a basis of detecting Pauli webs only make sense in the context of the global ZX diagram. For a partial diagram within this global ZX diagram, a detecting Pauli web of the global diagram may or may not be a detecting Pauli web of the local diagram. For example, the detectors from the rewritten diagram in the introduction to this chapter (in Eq. (3.2)) cover the rewritten part of the diagram as follows.



(3.3)

For a ZX rewrite that considers the detector basis of a global diagram, it is therefore insufficient to specify changes to the detectors of the local diagrams. Instead, it needs to consider all detectors that overlap with the local diagram.

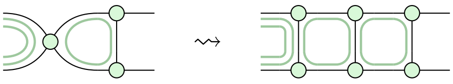
We begin to formalise these detector changes by defining what we mean with the restriction of a Pauli web to a part of a diagram. Note that the following definition allows non-detecting Pauli webs both as inputs and outputs of the restriction. This enables us to concatenate local restrictions.

Definition 3.1 (Local restriction of a Pauli web)**.** *Given a Pauli web P for a ZX diagram D and a subdiagram D' of D, we use $P[D']$ to denote the local restriction of P to D'. Formally, $P[D']$ is a Pauli web on D' that agrees with P on all wires, i.e.*

$$\forall e \in E(D') : \quad P[D'](e) = P(e)\,. \tag{3.4}$$

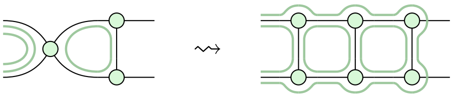
Local restrictions are not injective, that is, different Pauli webs on a diagram can have the same local restriction on some subdiagram. We have already seen this in the motivating example in Eq. (3.3). Non-injectivity is not a problem for local restrictions of single Pauli webs but we need to be careful about what we consider local restrictions for sets of Pauli webs. If we defined detector-aware rewrites on sets of local Pauli webs, we would be unable to distinguish between global detectors that have the same local restriction. Consequently, such a rewrite would be unable to modify detectors with the same local restriction in different ways.

We can exemplify why this is problematic by considering our motivating example again. This time, we consider a larger part of the diagram. Specifically we extend our consideration by the two nodes on the right side of the diagram. If we did not allow our rewrite to change detectors with the same local restriction in different ways, then all possible detector rewrites (with the same underlying ZX rewrite) between these two diagrams will become unmatchable, for example:



(3.5)

But if we allow different mappings for detectors with the same local restriction, we can recover matchability for this rewrite:



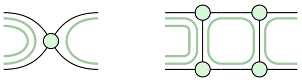
(3.6)

This demonstrates why it can be helpful for detector-aware rewrites to distinguish between detectors with the same local restrictions. As a compromise between locality and this ability to distinguish detectors as in this case, we resort to multisets of local Pauli web restrictions. We capture this in the following definition.

Definition 3.2 (Local restriction of a Pauli web multiset)**.** *Given a ZX diagram D with a Pauli web multiset S, and a subdiagram D', we use $S[D']$ to denote the local restriction of S to D'.*

$$S[D'] := \{\{P[D'] \mid P \in S, \quad P[D'] \text{ is non-trivial } \}\} \tag{3.7}$$

We will work with Pauli web multisets a lot in this chapter. In many cases, it will be tedious to list all Pauli webs in a multiset as separate diagrams, so we occasionally use the following shorthand notation which draws many Pauli webs into the same diagram by drawing the wire colouring as continuous lines next to the wires, instead of covering them. For example the local restriction Pauli web multiset of the introductory example LHS and RHS can be drawn in a single diagram as follows:



(3.8)

Here, each of the green lines represents a separate Pauli web. For too many Pauli webs, or Pauli webs with many wires, this notation can become convoluted, so we will alternate between the standard Pauli web notation and this Pauli web multiset notation depending on suitability to the problem at hand.

The embedding of local detector rewrites into their context imposes two boundary conditions. First, the Pauli web multisets covering the diagrams of the rewrite must be able to complete the same detecting regions that were partly cut off through the local restriction, after the rewrite[1]. Take for example the top-left detector from the introductory example which covered the local diagram only partly. If the detector rewrite does not provide a suitable replacement for the part that was cut off, then the remaining part becomes an incomplete Pauli web through the rewrite and we cannot have this as part of

[1] One may wonder if the sign of the Pauli webs is relevant here because detecting regions with negative signs zero out the diagram [13, Sec. 5.2]. However, as ZX rewrites are already between equivalent diagrams, both sides of the rewrites must have the same stabilisers and a diagram cannot stabilise both S and $-S$, so we will not need to pay attention to stabiliser signs here.

our new detector basis. We capture this condition in the following definition:

Definition 3.3 (Same boundary coverage)**.**

- *Two Pauli webs P_1, P_2 for equivalent diagrams D_1, D_2 have the* same boundary coverage *if they agree on all boundary wires, i.e.*

$$\forall e \in E(D_1): \quad e \text{ is boundary} \implies P_1(e) = P_2(e) \tag{3.9}$$

- *Two Pauli web multisets S_1, S_2 for equivalent diagrams D_1, D_2 have the* same boundary coverage *if there is multiset of pairs*

$$R = \big\{\{(P_1, P_1'), \ldots, (P_n, P_n')\}\big\} \tag{3.10}$$

  such that $S_1 = \{\{P_i \mid (P_i, P_i') \in R\}\}$, $S_2 = \{\{P_i' \mid (P_i, P_i') \in R\}\}$, and P_i has the same boundary coverage as P_i' for all $(P_i, P_i') \in R$. [2]

The second condition that we can observe from our running example is the fact that ZX rewrites might allow for entirely new detecting regions that cannot be generated from the detecting regions we have rewritten. If our rewrites do not add such detecting regions to the transformed basis, then the transformed basis does not generate the set of detectors for the rewritten diagram anymore. Therefore, a detector rewrite that transforms bases into other bases needs to add such local detectors, or more precisely, a basis of detectors of the local ZX diagram. On the other hand, of course, we need to remove local basis elements from the old local part of the diagram. We give another definition to formalise this concept.

Definition 3.4 (Local detector basis)**.** *A Pauli web multiset S of a ZX diagram D* has a local detector basis *if its detectors*

$$S_d = \{\{P \in S \mid P \text{ is a detector}\}\} \tag{3.11}$$

[2] These conditions aim generalise the set-theoretic concept of a bijective relation to multisets. A proper formalisation of bijective relations between multisets is studied in [37]. We have refrained from introducing multiset theory here because the increased precision is negligible in comparison to the notational overhead.

form a detecting region basis for D. In particular, all elements in S_d have multiplicity one.

With these preliminaries out of the way, we can now formalise our intuition about detector rewriting:

Definition 3.5 (Detector-aware rewrite)**.** *Let $D_1 = D_2$ be a ZX rewrite, and let S_1, S_2 be multisets of Pauli webs for D_1, D_2, respectively. We say that S_1 and S_2 form a* detector-aware rewrite *if the following conditions hold:*

- *S_1 and S_2 have the same boundary coverage*
- *S_1 and S_2 each have a local detector basis*

We denote this with $(D_1, S_1) \mathrel{\mathring{=}} (D_2, S_2)$. As the diagrams D_1 and D_2 are implicitly specified by the Pauli webs, we sometimes write $S_1 \mathrel{\mathring{=}} S_2$ if the multisets are nonempty.

Note that this definition does not require fault equivalence. While we will later require fault equivalence for matchability-preserving rewrites, detector-aware rewrites are in principle applicable to arbitrary ZX rewrites.

3.1.2 Properties of detector-aware rewrites

In this section, we state and prove a few essential properties of detector-aware rewrites. The main theorem in this subsection states how we can apply detector-aware rewrites to larger diagrams. Our proof is constructive which allows us to obtain a new detector basis for the diagram that was rewritten using a detector-aware rewrite. Furthermore, we prove that detector-aware rewrites always exist.

Before we get to the main result, we introduce another property for Pauli web multisets that will simplify the proof of this result.

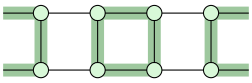
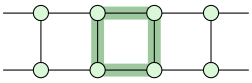
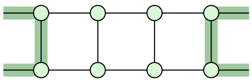
Definition 3.6 (Connected Pauli webs)**.** *Let P be a Pauli web for a ZX diagram D.*

- *A detecting Pauli web Q of D is a* detector component *of P if the following implication holds:*

$$\forall e \in E(D) : \quad Q(e) \neq I \implies P(e) = Q(e) \tag{3.12}$$

- P *is* disconnected *if there exists a nontrivial detector component $Q \neq P$ of P. P is* connected *if it is not disconnected and a Pauli web multiset is connected if all contained Pauli webs are connected.*

We illustrate this definition with an example: The following Pauli web P has two detector components, Q and the trivial Pauli web. As Q is neither trivial nor identical to P, P is disconnected.

$P =$  $Q =$  $PQ =$  (3.13)

Furthermore, Q is a detector component of itself. But since it has no nontrivial detector components besides itself, Q is connected. The only detector component of PQ is the trivial Pauli web, so PQ is connected too.

We will generally assume that Pauli web multisets are connected. This assumption is without loss of generality because we can always eliminate non-trivial, non-identical detector components from detector bases:

Lemma 3.7 (Removing disconnected detector components)**.** *For any detector basis $S = \{P_1, \ldots, P_n\}$ of a ZX diagram D, there is connected a detector basis $S' = \{P'_1, \ldots, P'_n\}$ such that P'_i is a detector component of P_i for all $1 \leq i \leq n$.*

Proof. Let S, P_i, D be as in the claim. If S is connected, then we are already done.

If S is disconnected, then we can construct a basis S' iteratively by eliminating disconnected components from the basis elements, similar to Gaussian elimination. For that, let P_i be a disconnected basis element for some $1 \leq i \leq n$. As P_i is disconnected, there must exist a nontrivial detector component Q of P_i with $Q \neq P_i$.

We first prove that $Q' := P_iQ$ is also a detector component of P_i from the fact that Q is a detector component:

$$\begin{aligned} I \neq P'_i(e) &\overset{def}{=} P_i(e)Q(e) \\ &\overset{\cdot P_i(e)}{\Longrightarrow} P_i(e) \neq Q(e) \\ &\Longrightarrow Q(e) = I \\ \Longrightarrow P'_i(e) = P_i(e)Q(e) &= P_i(e)I = P_i(e) \end{aligned} \tag{3.14}$$

As $Q \neq P_i$ holds, so does $Q' \neq P_i$.

Q is a detecting Pauli web, so we can compose $Q = Q_1 \cdots Q_m$ from basis elements $Q_1, \ldots, Q_m \in S$. Without loss of generality, we assume that $P_i \neq Q_j$ holds for all $1 \leq j \leq m$ (otherwise, we could have picked Q' instead of Q). Then we can construct a new detector basis as follows:

$$S' := \{P'_1, \ldots, P'_n\}$$
$$\forall 1 \leq j \leq n : \quad P'_j := \begin{cases} P_j & \text{if } j \neq i \\ P_i Q_1 \cdots Q_m & \text{if } j = i \end{cases} \tag{3.15}$$

For $i \neq j$, P'_j is a detector component of P_j because $P'_j = P_j$ holds. For $i = j$, we proved that $P'_i = P_i Q_1 \cdots Q_m = P_i Q = Q'$ is a detector component in Eq. (3.14).

We repeat this procedure with $S \leftarrow S'$ until we find have removed all detector components that made some basis element disconnected. The algorithm must terminate because in each step, we removed some edge colourings from a basis element and there are only finitely many edges and basis elements. □

We now get to the central theorem about detector-aware rewrites:

Theorem 3.8 (Applying detector-aware rewrites)**.** *Let $D(D_1)$ be a ZX diagram with a subdiagram D_1, let S_1 be a connected Pauli web multiset of $D(D_1)$ that has a local detector basis.*

If $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$ is a detector-aware rewrite, then there exists a Pauli web multiset S_2 for $D(D_2)$ such that $S_2[D] = S_1[D]$, $S_2[D_2] = T$, and $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_2), S_2)$ is a detector-aware rewrite.

Proof. Let D, D_1, D_2, S_1 and T be as assumed in the claim.

This proof is structured as follows: We first provide a construction for a Pauli web multiset S_2 for $D(D_2)$ based on the Pauli webs in S_1 and the detector-aware rewrite between D_1 and D_2. Afterwards, we prove for this construction that it has the same boundary coverage as S_1 and that $S_2[D] = S_1[D]$, $S_2[D_2] = T$ hold. In the final step, we prove that S_2 has a local detector basis for $D(D_2)$ by showing that the detecting Pauli webs in S_2 generate all detectors of $D(D_2)$ and that they are independent.

Construction: Let us first partition S_1 into three disjoint multisets:

$$\begin{aligned} S_{1,out} &:= \left\{\!\left\{ P \in S_1 \mid P[D_1] = I^{E(D_1)} \right\}\!\right\} , \\ S_{1,both} &:= \left\{\!\left\{ P \in S_1 \mid P[D_1] \neq I^{E(D_1)}, P[D] \neq I^{E(D)} \right\}\!\right\} , \\ S_{1,in} &:= \left\{\!\left\{ P \in S_1 \mid P[D] = I^{E(D)} \right\}\!\right\} . \end{aligned} \tag{3.16}$$

$S_{1,in}$ contains the detector basis of D_1 that is contained in S_1. $S_{1,both}$ contains the Pauli webs from S_1 that cover both D and D_1 and thanks to our connectivity assumption, $S_{1,both}$ contains exactly those Pauli webs in S_1 that cover the boundary edges between D and D_1 nontrivially. $S_{1,out}$ contains the rest of S_1's Pauli webs and these do not cover D_1 but they can cover boundary edges of D that are not connected to D_1. By definition, $S_{1,out} \cup S_{1,both} \cup S_{1,in} = S_1$ must hold. We now construct a Pauli web multiset S_2 for $D(D_2)$ by translating each of these three partitions from $D(D_1)$ to $D(D_2)$.

We translate $S_{1,out} = \{\{P_1, \ldots, P_k\}\}$ to a new multiset $S_{2,out} := \{\{P'_1, \ldots, P'_k\}\}$. The P'_i are constructed from their counterparts in $S_{1,out}$:

$$\forall 1 \leq i \leq k, e \in E(D(D_2)) : \quad P'_i(e) := \begin{cases} P_i(e) & \text{if } e \in E(D) \\ I & \text{if } e \in E(D_2) \end{cases} \tag{3.17}$$

$S_{1,both} = \{\{Q_1, \ldots, Q_m\}\}$ contains exactly those Pauli webs in S_1 that cover the boundary wires between D and D_1. We transform these Pauli webs to $D(D_2)$ by keeping the part in D intact and replacing the part in D_1 with a corresponding part using the same-boundary-coverage relation between the diagrams. For that, we first match the D_1 coverages of the Pauli webs in $S_{1,both}$ with their counterparts, following the same-boundary-coverage relation R' if the inner detector-aware rewrite $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$:

$$R' = \{\{(Q_1[D_2], q_1), \ldots, (Q_m[D_2], q_m)\}\} \tag{3.18}$$

We then construct $S_{2,both} := \{\{Q'_1, \ldots, Q'_m\}\}$ as follows.

$$\forall 1 \leq i \leq m, e \in E(D(D_2)) : \quad Q'_i(e) := \begin{cases} Q_i(e) & \text{if } e \in E(D) \\ q_i(e) & \text{if } e \in E(D_2) \end{cases} \tag{3.19}$$

Lastly, we construct the set $S_{2,in}$. This set does not depend on $S_{1,in}$ because just like the detector-aware rewrite $(D_1, S_1[D_1]) \mathrel{\overset{\circ}{=}} (D_2, T)$, we want to remove local detectors of D_1 and add detectors for D_2. We do this by embedding the local detector basis of $T = \{R_1, \dots, R_n\}$ into the larger diagram $D(D_2)$.

$$\begin{gathered} S_{2,in} := \big\{\{R'_1, \dots, R'_n\}\big\} \\ \forall 1 \leq i \leq n, e \in E(D(D_2)) : \quad R'_i(e) := \begin{cases} R_i(e) & \text{if } e \in E(D_2) \\ I & \text{if } e \in E(D) \end{cases} \end{gathered} \tag{3.20}$$

Same boundary coverage, sameness on D and D_2: For the sameness of the boundary coverage, we first realise that boundary-covering Pauli webs in S_i can only be in the $S_{i,out}$ and $S_{i,both}$ multisets. We can then assemble the multiset relation between the boundary-covering Pauli webs from our constructions above.

$$\begin{aligned} R = \quad & \big\{\{(P_i, P'_i) \mid P_i \in S_{1,out},\ P_i \text{ is not a detector }\}\big\} \\ & \cup \big\{\{(Q_i, Q'_i) \mid Q_i \in S_{1,both},\ Q_i \text{ is not a detector }\}\big\} \end{aligned} \tag{3.21}$$

The sameness of S_1's and S_2's D coverage follows directly from our construction of $S_{2,both}$ and $S_{2,out}$. $S_{i,in}$ does not contribute to the D coverage at all, by definition.

$$\begin{aligned} S_2[D] &= (S_{2,out} \cup S_{2,both} \cup S_{2,in})[D] \\ &= S_{2,out}[D] \cup S_{2,both}[D] \cup \underbrace{S_{2,in}[D]}_{=\emptyset} \\ &= S_{1,out}[D] \cup S_{1,both}[D] \cup \underbrace{S_{1,in}[D]}_{=\emptyset} \\ &= (S_{1,out} \cup S_{1,both} \cup S_{1,in})[D] \\ &= S_1[D] \end{aligned} \tag{3.22}$$

Finally, $S_2[D_2] = T$ follows from the fact that $S_{2,out}$ does not cover D_2 at all, $S_{2,both}$ covers D_2 the same way as the non-detecting Pauli webs in T and $S_{2,in}$ is an embedding of the detecting Pauli webs in T into $D(D_2)$.

$$\begin{aligned}
S_2[D_2] &= (S_{2,out} \cup S_{2,both} \cup S_{2,in})[D_2] \\
&= \underbrace{S_{2,out}[D_2]}_{=\emptyset} \cup S_{2,both}[D_2] \cup S_{2,in}[D_2] \\
&= \{\{P \in T \mid \ P \text{ is not a detector}\}\} \cup \{\{P \in T \mid \ P \text{ is a detector }\}\} \\
&= T
\end{aligned} \tag{3.23}$$

Generating set: Consider any nontrivial, local detecting Pauli web P of $D(D_2)$. We show that the detectors in S_2 generate P by translating P back to $D(D_1)$ and generating that detector from the local detector basis of S_1.

As $D_1 = D_2$, there exists a Pauli web P' for D_1 with the same boundary coverage as $P[D_2]$ on D_2. We can use P' to construct a local detector P^- of $D(D_1)$ as follows:

$$\forall e \in E(D_1): \quad P^-(e) = \begin{cases} P(e) & \text{if } e \in E(D) \\ P'(e) & \text{if } e \in E(D_1) \end{cases} \tag{3.24}$$

By construction, $P^-[D] = P[D]$ must hold.

As S_1 has a local detector basis, we can find detecting Pauli webs $P_1, \ldots, P_n \in S_1$ such that $P^- = P_1 \cdots P_n$. Without loss of generality, we assume that the P_i are ordered such that $P_1, \ldots, P_m$ cover D nontrivially and $P_{m+1}, \ldots, P_n$ cover D trivially for some $0 \leq m \leq n$. Then $P_1 \cdots P_m$ cover D the same way as P^- and P, that is

$$(P_1 \cdots P_m)[D] = (P_1 \cdots P_m \cdot \underbrace{P_{m+1} \cdots P_n}_{=I^{E(D)}})[D] = P^-[D] = P[D]\,. \tag{3.25}$$

By our construction of S_2, each $P_i \in S_1$ (for $1 \leq i \leq m$) has a corresponding $P_i' \in S_2$ such that $P_i[D] = P_i'[D]$. Then $(P_1' \cdots P_m')[D] = (P_1 \cdots P_m)[D] = P[D]$, so $P_1' \cdots P_m'$ and P can only differ by a local detector of D_2. But our construction ensured that S_2 had a local detector basis for D_2, so we can find $Q_1, \ldots, Q_k \in S_2$ such that $Q_1 \cdots Q_k = P_1' \cdots P_m' \cdot P$. Therefore P is generated by S_2 as follows

$$P = P_1' \cdots P_m' \cdot Q_1 \cdots Q_k \tag{3.26}$$

and we conclude that S_2 has a local detector basis for $D(D_2)$.

Independence: Assume that the detecting Pauli webs in S_2 are not independent. Then there must exist a nonempty set of detecting Pauli webs $\{P'_1, \dots, P'_n\} \subseteq S_2$ such that $P'_1 \dots P'_n$ is trivial. We will lead this assumption to contradiction by translating this set to a nonindependent set of local detectors in S_2.

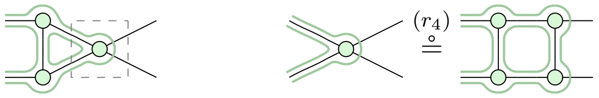
Without loss of generality, we assume that the P_i are ordered such that $P'_1, \dots, P'_m$ cover D nontrivially and $P'_{m+1}, \dots, P'_n$ cover D trivially for some $0 \leq m \leq n$. By our construction, all $P'_i \in S_2$ that cover D nontrivially are in correspondence with $P_i \in S_1$ such that $P_i[D] = P'_i[D]$. Therefore, we have

$$\begin{aligned}
(P_1 \cdots P_m)[D] &= P_1[D] \cdots P_m[D] \\
&= P'_1[D] \cdots P'_m[D] \\
&= P'_1[D] \cdots P'_m[D] \cdot \underbrace{P'_{m+1}[D]}_{=I^{E(D)}} \cdots \underbrace{P'_n[D]}_{=I^{E(D)}} \\
&= (P'_1 \cdots P'_m \cdot P'_{m+1} \cdots P'_n)[D] \\
&= I^{E(D)},
\end{aligned} \tag{3.27}$$

so $P_1 \cdots P_m$ must be a local detector of D_1.

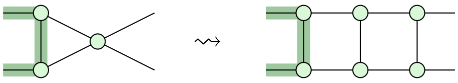
But S_2 contains a local detector basis for D_1, so there exist independent $Q_1, \dots, Q_k \in S_1$ such that $P_1 \cdots P_m = Q_1 \cdots Q_k$. Therefore, $P_1 \cdots P_m \cdot Q_1 \cdots Q_k$ is trivial and therefore, the detecting Pauli webs in S_1 cannot have been independent. This contradicts our assumption that S_1 had a detector basis and we conclude that the detectors in S_2 must be independent. □

Let us now consider an example detector-aware rewrite to demonstrate the construction of the new Pauli web multiset. We want to apply the following detector-aware r_4 rewrite to the dashed box in the left diagram.



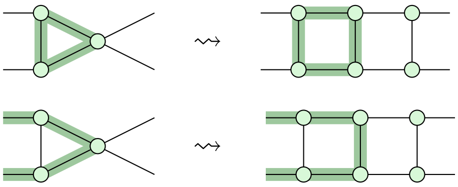
(3.28)

We first take care of the $S_{1,out}$ Pauli webs. There is just one such Pauli web, namely the left one and we can copy it to the new diagram on the outer diagram, leaving the rewritten part uncovered.



(3.29)

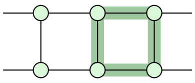
There are two webs in $S_{1,both}$, namely the only detector of the LHS and the remaining stabilising Pauli web of the LHS. To transform them for the LHS, we keep the outer part intact and replace the rewritten part with an RHS web from the detector-aware rewrite that we are applying. In this case, both webs we want to transform have the same boundary coverage for the inner diagram. It does not matter how we match them up with the RHS webs, as long as the LHS and RHS webs of the inner, detector-aware rewrite are in bijection and the boundary coverage of the pairs is the same. We choose the following RHS webs.



(3.30)

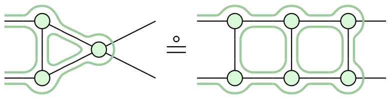
Finally, we need to discard all local detectors of the inner, detector-aware rewrite on the LHS and introduce the local detector basis of the RHS. In this case, we do not need to remove anything and need to add just one detector.

(none) $\rightsquigarrow$



(3.31)

This leaves us with the following rewrite on the bigger diagram.



(3.32)

Theorem 3.8 tells us how to *apply* detector-aware rewrites to larger diagrams and how to construct the resulting Pauli web multiset from the previous one. However, our motivation for detector-aware rewrites was not to rewrite Pauli web multisets but detector bases. And indeed, detector bases are always mapped to detector bases by detector-aware rewrites.

Corollary 3.9 (Detector-aware rewrites keep detector bases intact)**.** *When applying a detector-aware rewrite* $(D_1, T_1) \mathrel{\hat{=}} (D_2, T_2)$ *to a diagram* $D(D_1)$ *with a* detector basis S_1, *the resulting Pauli web multiset* S_2 *for* $D(D_2)$ *is a detector basis of* $D(D_2)$.

Proof. Let D_1, D_2, D be ZX diagrams, let S_1 be a detector basis for $D(D_1)$, let T be a Pauli web multiset for D_2 such that $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$ is a detector-aware rewrite.

Then Theorem 3.8 allows us to construct a Pauli web multiset S_2 for $D(D_2)$ such that $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_2), S_2)$ is a detector-aware rewrite and $S_2[D] = S_1[D]$ as well as $S_2[D_2] = T$ hold. S_1 and S_2 must have the same boundary coverage, so S_1 containing only detectors implies S_2 containing only detectors. Furthermore, S_2 has a local detector basis and as there cannot be any other Pauli webs in S_2, S_2 is a detector basis for $D(D_2)$. □

Before we get to defining matchability-preserving rewrites, we will prove one last property about detector-aware rewrites:

Lemma 3.10 (Detector-aware rewrites always exist)**.** *Let D_1, D_2 be equivalent ZX diagrams and let S_1 be a Pauli web multiset with a local detector basis for D_1. Then there exists a Pauli web multiset S_2 for D_2 such that $(D_1, S_1) \mathrel{\hat{=}} (D_2, S_2)$ is a detector-aware rewrite.*

Proof. Let D_1, D_2, S_1 be as in the claim. We separate S_1 into the set of local detectors $S_{1,d}$ and the set of boundary-covering webs $S_{1,b}$ with $S_1 = S_{1,d} \cup S_{1,b}$. Each $P \in S_{1,b}$ corresponds to a stabiliser S_P of D_1 and as D_1 and D_2 are equivalent, S_P must be a stabiliser of D_2 too. Therefore, there must exist a stabilising Pauli web P' corresponding to S_P in D_2. We use this correspondence to construct a Pauli web multiset with the same boundary coverage as $S_{1,b}$ as follows:

$$S_{2,b} := \{\{P' \mid P \in S_{1,b}\}\} \tag{3.33}$$

Furthermore, we can pick an arbitrary local detector basis $S_{2,d}$ of D_2. Then $S_2 := S_{2,d} \cup S_{1,b}$ is a Pauli web multiset for D_2 that has a local detector basis and the same boundary coverage as S_1 by construction. Therefore, $(D_1, S_1) \mathrel{\hat{=}} (D_2, S_2)$ is a detector-aware rewrite. □

3.1.3 Matchability-preserving rewrites

So far in this section, we have defined detector-aware rewrites and we have characterised how we can apply them to rewrite detector bases along with the ZX diagram. For the final part of this section, we will combine fault equivalence, detector-awareness and matchability to define *matchability-preserving rewrites.* We will see that many relevant characteristics of these rewrites follow straightforwardly from these defining properties.

To get started, let us first generalise the notion of matchable detector bases to Pauli web multisets.

Definition 3.11 (Matchable Pauli web multiset)**.** *We call a Pauli web multiset S for a ZX diagram D* matchable *if each wire in D is covered by at most two Pauli webs in S, i.e.*

$$\forall e \in E(D) : \quad |\{\{P \in S \mid P(e) \neq I\}\}| \leq 2 \tag{3.34}$$

It's easy to see that for a detector basis S, this definition coincides with the matchability condition presented in Section 2.5. We can use this notion to reason about matchability of a detector basis locally. This allows a natural restriction of detector-aware rewrites to *matchability-preserving* rewrites, which additionally to diagram and detector rewriting also ensure that the rewrite does not introduce any faults that would violate matchability.

Definition 3.12 (Matchability-preserving rewrite)**.** *We call $(D_1, S_1) \mathrel{\hat{=}} (D_2, S_2)$ a* matchability-preserving rewrite *if $D_1 \hat{=} D_2$ is a fault-equivalent ZX rewrite, $(D_1, S_1) \mathrel{\overset{\circ}{=}} (D_2, S_2)$ is a detector-aware rewrite, and both S_1 and S_2 are matchable Pauli web multisets.*

Note that this definition also requires matchability-preserving rewrites to be fault-equivalent rewrites. This is technically not required for defining matchability-preservation but we find that matchability preservation will rarely be helpful without fault equivalence because efficient decoding cannot recover decreases of the distance, for example.

Applying matchability-preserving rewrites to a ZX diagram and Pauli web multiset works the same way as applying detector-aware rewrites. We capture this in the following corollary:

Corollary 3.13 (Matchability-preserving rewrites work as expected)**.** *Let $D(D_1)$ be a ZX diagram with a subdiagram D_1 and let S_1 be a connected,* matchable *Pauli web multiset of $D(D_1)$ that has a local detector basis.*

(i) If $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$ *is a* matchability-preserving *rewrite, then there exists a Pauli web multiset* S_2 *for* $D(D_2)$ *such that* $S_2[D] = S_1[D]$*,* $S_2[D_2] = T$*, and* $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_2), S_2)$ *is a* matchability-preserving rewrite*.*

(ii) If additionally, S_1 *is a matchable detector basis of* $D(D_1)$*, then the resulting* S_2 *is a matchable detector basis of* $D(D_2)$*.*

Proof. Let D, D_1, D_2, S_1, T be as assumed in the claim.

(i) Following Theorem 3.8, the theorem about applying detector-aware rewrites, we can construct a Pauli web multiset S_2 for $D(D_2)$ such that $S_2[D] = S_1[D]$, $S_2[D] = T$ and $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_1), S_2)$ is a detector-aware rewrite. It remains to be shown that $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_1), S_2)$ is matchability-preserving, that is, it must be fault-equivalent and S_2 must be matchable.

The fault equivalence $D(D_1) \hat{=} D(D_2)$ follows from the compositionality of fault equivalence [13, Prop. 3.11]. For the matchability of S_2, we observe that $S_2[D] = S_1[D]$ is matchable by assumption and $S_2[D_2] = T$ is matchable because $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$ is assumed to be a matchability-preserving rewrite.

(ii) Let S_1 be a matchable detector basis. Then (i) states that the constructed S_2 must be matchable and Corollary 3.9 states that S_2 must be a detector basis.

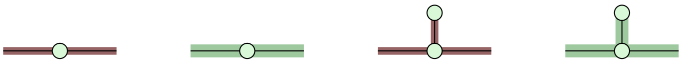
□

We will discuss various examples for matchability preserving rewrites in the upcoming sections.

3.2 Uniquely detector-aware rewrites

After introducing detector-aware rewrites in the previous section, we will look at a class of detector-aware rewrites that are particularly easy to reason about in this section. Namely, we will introduce the concept of *uniquely detector-aware* rewrites. ZX rewrites that admit no local detecting regions in their target are prominent examples of such rewrites and we will apply this property to characterise the detector-aware and matchability-preserving variants of the fault-equivalent r_{elim} and r_{fuse} rewrites.

Let us consider the r_{elim} rewrite as an example first. The LHS and RHS diagrams of this rewrite only admit two different non-trivial Pauli webs, one green and one red.



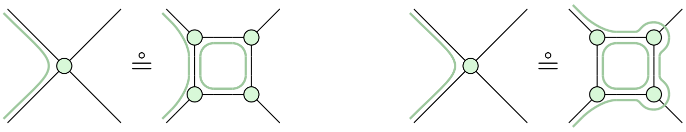
(3.35)

Notably, neither the LHS nor the RHS of this rewrite allow for any non-trivial detecting region and for each stabiliser of these diagrams, there is only a single Pauli web realising this stabiliser. This property fully defines a boundary-coverage-preserving one-to-one correspondence between the Pauli webs on the LHS and those on the RHS. Therefore, any Pauli web multiset for the LHS is made up for n green Pauli webs and m red Pauli webs, and the only Pauli web multiset for the RHS with the same boundary coverage contains the corresponding n green Pauli webs and m red Pauli webs. We capture this uniqueness constraint in the following definition.

Definition 3.14 (Uniquely detector-aware rewrite)**.** *Let D_1, D_2 be equivalent ZX diagrams.*

If for any Pauli web multiset S_1 for D_1 with a local detector basis, there exists a unique Pauli web multiset S_2 for D_2 such that $(D_1, S_1) \mathrel{\hat{=}} (D_2, S_2)$ is a detector-aware rewrite, we say that the rewrite $D_1 = D_2$ is uniquely detector-aware.

Note that this definition is specific to the direction of a rewrite. A ZX rewrite can be uniquely detector-aware in one direction, and not uniquely detector-aware in the other direction. Take for example the following two detector-aware rewrites for the fault-equivalent r_4 rewrite.



(3.36)

These two rewrites demonstrate that for the given LHS Pauli web multiset, we can find multiple different RHS Pauli web multisets that lift this rewrite to a detector-aware rewrite, and it is therefore not uniquely detector-aware. In the other direction, however, r_4 is uniquely detector-aware because the LHS has no internal wires and any given boundary coverage fully determines the Pauli webs.

We have noted before for the r_{elim} example that its diagrams do not permit any local detecting regions, and in the r_4 example too, the LHS does not permit any detecting region.

This is no coincidence and we characterise the relationship between the nonexistence of local detecting regions and unique detector-awareness in the following proposition.

Proposition 3.15 (ZX rewrites without detecting regions are uniquely detector-aware)**.** *If $D_1 = D_2$ is a ZX rewrite and D_2 has no nontrivial, local detecting regions, then $D_1 = D_2$ is uniquely detector-aware.*

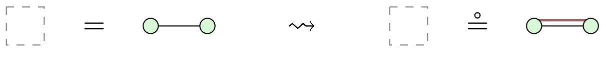
Proof. Let D_1, D_2 be equivalent ZX diagrams such that D_2 has no local detectors and let S_1 be an arbitrary Pauli web multiset for D_1 that has a local detector basis. By Lemma 3.10, there exists some Pauli web multiset S_2 for D_2 such that $(D_1, S_1) \overset{\circ}{=} (D_2, S_2)$ is a detector-aware rewrite. It remains to be shown that S_2 was unique.

Let S_2' be another Pauli web multiset such that $(D_1, S_1) \overset{\circ}{=} (D_2, S_2)$ is a detector-aware rewrite. As S_1 and S_2 have the same boundary coverage, and S_1 and S_2' have the same boundary coverage, S_2 and S_2' must have the same boundary coverage. Therefore, there exists a multiset of Pauli web pairs

$$\begin{gathered} R = \{\{(P_1, P_1'), \ldots, (P_n, P_n')\}\} \\ S_2 = \{\{P_i \mid (P_i, P_i') \in R\}\} \qquad S_2' = \{\{P_i' \mid (P_i, P_i') \in R\}\} \end{gathered} \tag{3.37}$$

such that all Pauli web pairs $(P_i, P_i') \in R$ have the same boundary coverage. But then, $P_i \cdot P_i'$ must be a detecting Pauli web as the boundary coverage cancels out. By our assumption, the detecting Pauli web $P_i \cdot P_i'$ in D_2 must be trivial and therefore we have $P_i = P_i'$ for all $1 \leq i \leq n$. Consequently, $S_2 = S_2'$ must hold and we conclude that $D_1 = D_2$ is uniquely detector-aware. □

So far, we have only seen uniquely detector-aware rewrites whose target diagrams had no nontrivial detecting regions. This correlation naturally raises the question whether the converse to Proposition 3.15 is also true, that is, whether every uniquely detector-aware rewrite has no nontrivial detecting regions in the target diagram. It turns out that there are rewrites that are uniquely detector-aware and still have detecting regions, like the following one:



(3.38)

Here, the right-hand side has a single detector, but for this to be a detector-aware rewrite, there is only one possible choice for both the detector basis and for the boundary covering Pauli webs. This rewrite is not very interesting because it merely creates a scalar diagram. and cannot be used to change the structure of a larger diagram. We characterise such rewrites in the following remark.

Remark 3.16 (Uniquely-detector aware rewrite with local detectors)**.** *Let $D_1 = D_2$ be a uniquely detector-aware rewrite such that D_2 has a nontrivial, detecting Pauli web. First, D_2 can only have a single detector, or the uniqueness property may be violated because we could pick another detector basis for D_2's Pauli web multiset. Second, D_2 cannot have any non-detecting Pauli webs. Otherwise, we would again violate the uniqueness property because we could add the detector to any of these stabilisers to get another Pauli web multiset that lifts $D_1 = D_2$ to a detector-aware rewrite.*

These properties leave us with rewrites that are not really relevant in the context of fault tolerance because if our diagrams had any boundary wires, but no locally stabilising Pauli webs, then all of the boundary edge flips would be locally non-trivial and globally undetectable. But if all faults on the boundaries of D_1 and D_2 are irrelevant, this rewrite is meaningless. In this sense, Proposition 3.15 is an equivalence for all rewrites that could potentially be interesting to us.

With this characterisation, we have gotten a good overview what types of rewrites are uniquely detector-aware and what properties they have. Uniquely detector-aware rewrites are relevant to this work because they preserve matchability:

Proposition 3.17 (Uniquely detector-aware rewrites preserve matchability)**.** *Let $D_1 \hat{=} D_2$ be a uniquely detector-aware and fault-equivalent rewrite. Then for any matchable Pauli web multiset S_1 for D_1 with a local detector basis, the unique detector-aware rewrite $(D_1, S_1) \overset{\circ}{=} (D_2, S_2)$ preserves matchability.*

Proof. Let D_1, D_2, S_1 be as described above and let S_2 be the unique Pauli web multiset for D_2 that lifts $D_1 \hat{=} D_2$ to a detector-aware rewrite. Then $(D_1, S_1) \overset{\circ}{=} (D_2, S_2)$ is already fault-equivalent and detector-aware, and S_1 was assumed to be matchable. It remains to be shown that S_2 is matchable too.

For that, let F be an atomic fault from D_2's edge flip noise model. We distinguish the following two cases.

- **Case 1 (F locally detectable)**: We know from Remark 3.16 that D_2 can have at most one local detector, and that if such a local detector exists, S_2 cannot contain any other Pauli webs. So if F is locally detectable, it must anticommute with the single Pauli web in S_2, so F must be matchable.

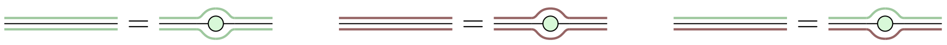
- **Case 2 (F locally undetectable)**: If F is locally undetectable, the fault equivalence of D_1 and D_2 implies that there must exist an equivalent fault on D_1 with at most the same weight. So there must exist a fault F' on D_1 with $\mathrm{wt}(F') \leq \mathrm{wt}(F) \leq 1$, so F' can only be atomic or trivial. If F' is trivial, it doesn't violate any detectors, and if it is nontrivial, then by the matchability of S_1, it can violate at most two detectors. But since F' and F must have an equivalent effect on the diagram, F and F' must violate the same stabilisers of D_1 and D_2. As F didn't violate any internal detectors of D_2 and F' violated at most two stabilising Pauli webs from S_1, F also violates at most two Pauli webs from S_2 and therefore F is matchable.

We chose F to be an arbitrary atomic fault on D_2, so S_2 must be matchable and we conclude that $(D_1, S_1) \mathrel{\overset{\circ}{=}} (D_2, S_2)$ is a matchability-preserving rewrite. □

For rewrites without detectors on the target diagram, we can prove an even stronger result: Fault-equivalent rewrites without local detectors on either side preserve the Tanner graph (up to fault equivalence). While we do not explore this stronger property in this work, we point out that Pesah et al. recently reported working on such a notion of equivalence [38].

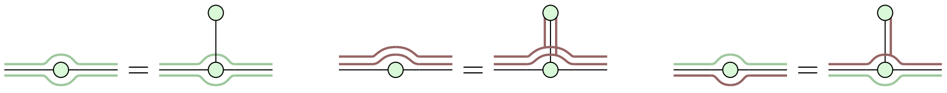
We can apply this proposition to many fault-equivalent rewrites from [14, 13]:

Example 3.18 (r_{elim} preserves matchability in both directions)**.**



(3.39)

Example 3.19 (r_{fuse} preserves matchability in both directions)**.**



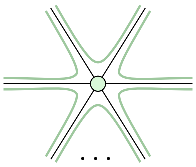
(3.40)

Furthermore, all fault-equivalent rewrites that decompose a single spider into some larger diagram preserve matchability in the reverse direction. This includeds r_4, r_5, r_{even} and r_{odd}. However, we will rarely ever use these rewrites in the reverse direction.

3.3 Decomposing few-legged spiders

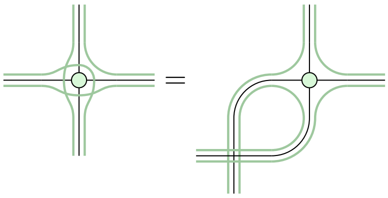
We now turn our attention to the r_4 rewrite which we have already discussed for a bit in the introduction of this chapter. As it turns out, the problem pointed out in the introduction can only occur in certain detector contexts. In this section, we will discuss cases where the problem doesn't occur. We will apply the same methodology to the r_5 and a six-legged, fault-equivalent rewrite.

Definition 3.20 (Blossom-shaped detecting regions)**.** *We say that the Pauli web multiset of an n-legged spider is* blossom-shaped *if it contains n different Pauli webs in the colour of the spider, each of them is covering two adjacent legs.*



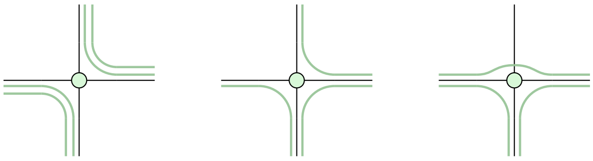
(3.41)

In this definition, the spatial adjacency of the wires covered by Pauli webs is at first a stronger property than we actually need. What we really mean to say is that we can give some order to these wires such that all adjacent legs (w.r.t. this order, also considering the first and last leg adjacent) are "connected" by a Pauli web from the multiset. However, thanks to the OCM rule of the ZX Calculus, these conditions are equivalent. We can demonstrate this with the following spider, covered by a Pauli web multiset that satisfies the ordering property, but doesn't have the blossom shape. Using the OCM rule, we can easily bend the wires to give the Pauli web coverage of the spider the blossom shape.



(3.42)

Crucially, spiders that don't satisfy the ordering property cannot be bent to the blossom shape. This is true for the following examples:

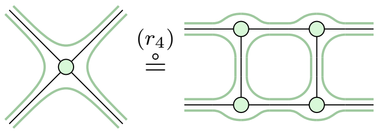


(3.43)

In the first example, some of the Pauli webs are duplicate. The second example has fewer Pauli webs than legs, but it is a sub-multiset of a blossom-shaped Pauli web multiset. The third example also has less Pauli webs than legs but unlike the second one, it is not a sub-multiset of a blossom-shaped detector context.

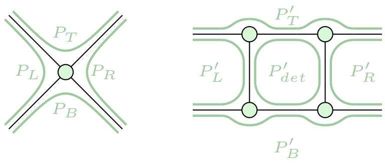
For four-legged spiders with a blossom-shaped Pauli web coverage, we can lift the fault-equivalent r_4 rewrite to a matchability-preserving rewrite as follows.

Lemma 3.21 (r_4 is a matchability-preserving rewrite in the blossom-shaped detector context)**.**



(3.44)

Proof. We already know that r_4 is a fault-equivalent ZX rewrite, so we just need to observe that the given Pauli web multisets satisfy the requirements for detector-aware and matchability-preserving rewrites. We label the Pauli webs as follows to simplify reasoning about them.



(3.45)

The left-hand side does not admit any local detectors and the right-hand side only admits a single local detector P'_{det}, which is contained in the Pauli web multiset once. Therefore the Pauli web multisets of both sides have local detector bases. The following multi-relation between the non-detecting Pauli webs from the multisets shows that the Pauli web multisets indeed have the same boundary coverage:

$$\big\{\{(P_T, P'_T), (P_L, P'_L), (P_B, P'_B), (P_R, P'_R)\}\big\} \tag{3.46}$$

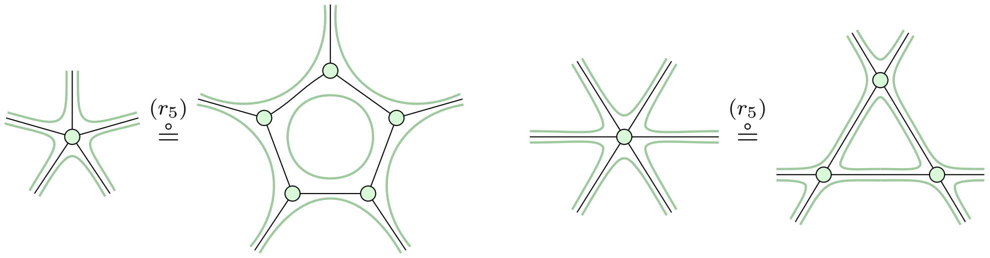
This proves that the above rewrite is a detector-aware rewrite.

It is immediately visible in the diagram that the Pauli web multisets are matchable, as each edge is covered by at most two Pauli webs from these sets. that the given rewrite is matchability-preserving under any edge-flip noise model since each wire in the two diagrams is covered by at most two Pauli webs. We conclude that the given rewrite preserves matchability. □

The key to matchability preservation for this rewrite was the fact that the ring structure of the RHS to nicely complements the blossom-like Pauli web coverage. Each of the boundary Pauli webs ("blossom leaves") will cover at most one of the internal ring wires, and no two boundary Pauli webs will cover the same internal wire. Therefore, the additional wire coverage by the detector in the right-hand side will not destroy the matchability property.

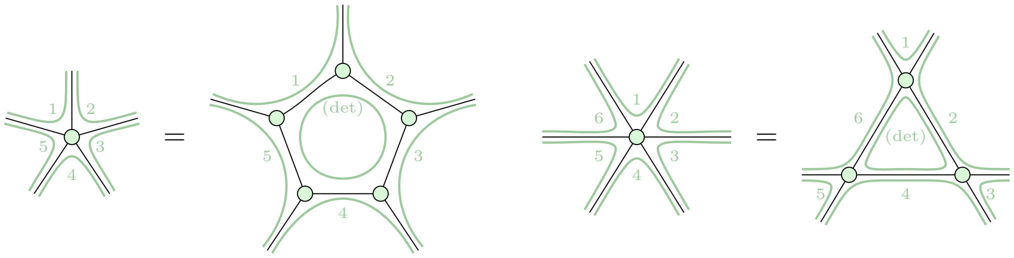
We can use a similar strategy to make r_5 and a novel r_6 rewrite matchable because they too decompose a single spider to such a ring-like structure.

Lemma 3.22 (r_5 and r_6 are matchability-preserving rewrites in the blossom-shaped detector context)**.**


$\overset{(r_5)}{\mathrel{\stackrel{\circ}{=}}}$
$\overset{(r_5)}{\mathrel{\stackrel{\circ}{=}}}$

(3.47)

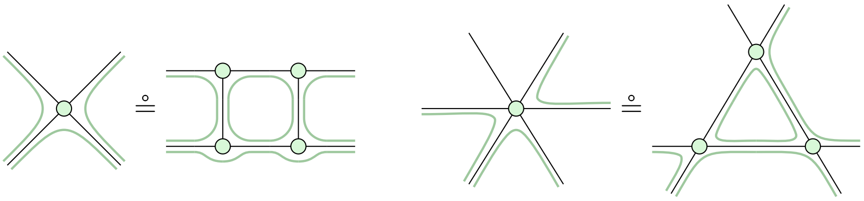
Proof. Both rewrites are already known to be fault-tolerant from [13]. We still need to show the detector-awareness and matchability of the given Pauli web multisets. The following labeling uses numbers to specify the multirelation between the non-detecting Pauli webs of the LHS and RHS of the rewrites:



(3.48)

Furthermore, we have marked the only detecting Pauli webs in these diagrams as "(det)". As the LHS diagrams do not have permit any detecting Pauli webs and the RHS Pauli web multisets contain the only detecting Pauli web, all of these Pauli web multisets have local detector bases. Finally, we can observe that each wire is covered by at most one Pauli web from the respective multisets. We conclude that these rewrites are detector-aware and matchability preserving. □

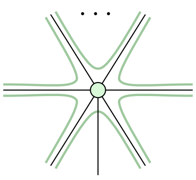
Note that for these (and all other matchability-preserving) rewrites, we can always leave out pairs of non-detecting Pauli webs from the Pauli web multisets of the LHS and RHS. This does not affect the local detector bases, keeps the same-boundary-coverage property intact and does not violate the matchability requirement. So by showing that the rewrites r_4, r_5 and r_6 preserve matchability under these blossom shapes, we have actually proven matchability preservation for a whole family of rewrites such as the following examples:



(3.49)

With these rewrites for blossom detecting contexts and all of their subset rewrites, we can already rewrite four-to-six-legged spiders in many detector contexts. This includes the second non-blossom-shaped detecting context from Eq. (3.43). We will now show that for the third non-example, which arranged the Pauli web multisets in a similar fashion as the blossom shape but left one leg uncovered, we can produce similar matchability-preserving rewrites. Let us first define this case more clearly.

Definition 3.23 (Flower-shaped detector context)**.** *The detector-context of an n-legged spider is* flower-shaped *if there are $n-1$ different Pauli webs, each covering two legs of the spider such that one wire remains uncovered and all but one Pauli web cover adjacent legs.*

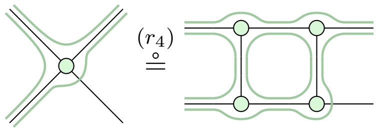


(3.50)

Again, this definition is essentially up to permutations of the legs (see Eq. (3.42)), and we can give similar non-examples with detecting regions occurring multiple times, too few detectors and too few spider legs being covered by the Pauli web multiset (see Eq. (3.43)). For flower-shaped Pauli webs, we can lift the r_4, r_5 and r_6 rewrites to be matchability-preserving, similar to our treatment for blossom-shaped Pauli web multisets.

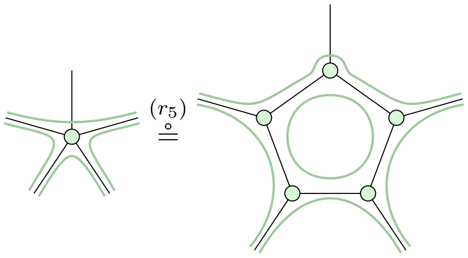
Lemma 3.24 (Matchability-preserving rewrites for four-to-six-legged flowers)**.**

(i)



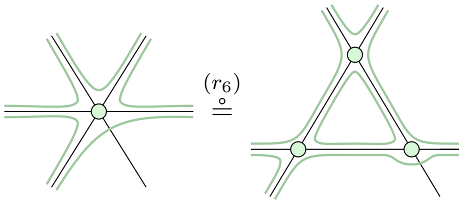
(3.51)

(ii)



(3.52)

(iii)



(3.53)

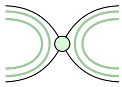
Proof. The proof is analogous to Lemmas 3.21 and 3.22. □

These shapes may seem arbitrary now but we will see later in this chapter that many other Pauli web multisets for single spiders can be reduced to these shapes. This will allow to to claim that under certain conditions, spiders with arbitrary Pauli web multisets can be decomposed with matchability-preserving rewrites.

3.4 Detector partitioning

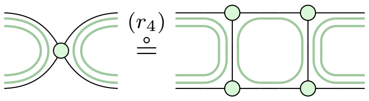
In the previous section, we have seen that we can decorate the r_4, r_5, and r_6 rewrites to be matchability-preserving if the detector coverage of the single spider is a subset of the flower or blossom shape. The remaining cases will be covered in this section and we will show that we can reduce each of these cases to spiders covered in blossom shapes under conservative assumptions regarding the distance of the full diagram. We call this process *detector partitioning* and generalise it to deal with any many-legged spider that is not covered in a subset of the flower or blossom shape.

Consider the following matchable detector coverage of a four-legged spider, which is made up of two "double detectors", that is, Pauli webs that are contained twice in the Pauli web multiset.



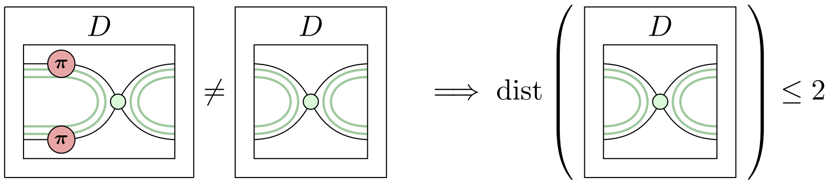
(3.54)

This Pauli web multiset neither has the flower nor the blossom shape discussed in the previous section and indeed, any detector-aware rewrite based on r_4 will not preserve matchability. This is because we need to "route" four non-detector Pauli webs between the bottom and the top spiders, but we only have two vertical wires and the local detector basis of the RHS covers both vertical wires once already.



(3.55)

However, we can also use these double detectors to our advantage: A fault made up of two X edge flips on the wires covered by such a pair of detectors is undetectable because detectors are only violated by odd numbers of anticommuting edge flips. If this weight-two fault is nontrivial in the full diagram, this would imply dist$(D) \leq 2$ for the entire circuit D where the four-legged spider occurred.

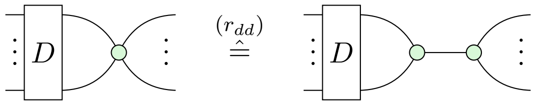


(3.56)

But such a low distance is a rare (and unhelpful) property since many QEC codes use the distance to argue about correctability; and this specific distance does not imply the correctability of any error. It is therefore reasonable to assume dist$(D) > 2$. By contraposition of Eq. (3.56), the $X \otimes X$ fault must therefore be trivial in the context of the full diagram. In the language of stabilisers, this means that $X \otimes X$ at the indicated positions must be stabilised by D.

For contexts with this property, we can construct a new fault-equivalent rewrite:

Lemma 3.25 (Double-detector rewrite r_{dd})**.** *For a ZX diagram D that stabilises $X \otimes X$ on the internal wires, the following rewrite is fault-equivalent.*

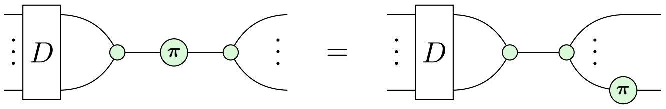


(3.57)

We will refer to this rewrite as the double-detector rewrite r_{dd}.

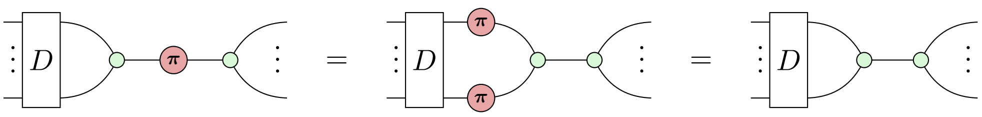
Proof. The ZX diagrams of this rewrite are equivalent through the spider fusion rule. For fault equivalence, we need to show that all undetectable faults on the left-hand side have an equivalent fault with less or equal weight on the right-hand side, and vice versa. For the first direction, we realise that all wires on the left-hand side of the rewrite also exist on the right-hand side. Therefore, all faults have an identical equivalent in this direction.

For the other direction, we consider an arbitrary fault F. If this fault contains a Z flip on the new wire, we can push out the Z flip to observe that it's equivalent to an error on an external wire which also exists in the LHS diagram.



(3.58)

If F contains an X fault, we can push it through the left spider to get the $X \otimes X$ fault that is stabilised by the context D.

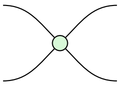
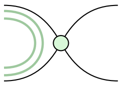


(3.59)

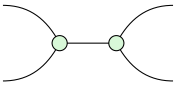
As D absorbs this error, X flip too was trivial. If F contains a Y fault on the new wire, this is equivalent to an X and a Z fault occuring on the internal wire but we have previously argued that the X flip is trivial, so the Y flip is equivalent to the Z flip which we discussed before. All other wires also exist in the LHS diagram and we can therefore apply the identical fault on the LHS. □

This rewrite looks different from the ZX rewrites that we are used to because it leaves the context D unspecified, except for the stabiliser condition. This raises the question of how such a rewrite should be applied. We consider the part of the rewrite that isn't the context D the interesting part of the rewrite – this is where the change to the diagram happens. When considering whether we can apply this rewrite, we try to find the changing part somewhere in a diagram and then search the remaining diagram for a suitable context. We then pick the context D to be large enough that it includes the required stabilising Pauli web[3]. Crucially, finding such a suitable context can be done in polynomial time over the diagram size, so it is efficient enough to be used during compilation [15, 39].

Proposition 3.26 (Double-detector rewrite preserves matchability)**.** *Let* D, D_1 *be ZX diagrams with a matchable detector basis* S_1 *of* $D(D_1)$ *such that* $\mathrm{dist}(D(D_1)) > 2$*,*

$D_1 =$  *and*  $\subseteq S_1[D_1]\,.$ (3.60)

Then there exists a Pauli web multiset S_2 *for* $D(D_2)$ *and*

$D_2 =$  (3.61)

such that $S_1[D] = S_2[D]$ *and* $(D(D_1), S_1) \stackrel{\circ}{=} (D(D_2), S_2)$ *preserves matchability.*

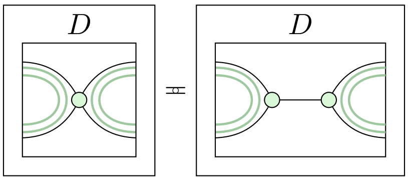
Proof. As we assumed S_1 to be matchable and the indicated wires of D_1 are already covered by two detectors, there cannot be any other detectors covering these wires. Therefore, an $X \otimes X$ fault on these wires is undetectable. As $\mathrm{dist}(D(D_1)) > 2$, this fault must be equivalent to the trivial fault in $D(D_1)$. Then Lemma 3.25 implies that $D(D_1) \hat{=} D(D_2)$ is a fault-equivalent rewrite.

As D_2 does not allow any local detectors, we can use Proposition 3.15 to construct a unique Pauli web multiset S_2' for D_2 such that $(D_1, S_1[D_1]) \stackrel{\circ}{=} (D_2, S_2')$ is a detector-aware rewrite. We can apply this detector-aware rewrite to $(D(D_1), S_1)$ following Theorem 3.8 to get a Pauli web multiset S_2 for $D(D_2)$ such that $S_2[D] = S_1[D]$, $S_2[D_2] = S_2'$ and $(D(D_1), S_1) \stackrel{\circ}{=} (D(D_2), S_2)$ is a detector-aware rewrite.

[3] The performance of automated rewriting may benefit from choosing a minimal context to avoid wasting resources. For the theory, we can also just pick the entire diagram, as long as it has the desired property.

We already know that $D(D_1)\hat{=}D(D_2)$ is fault-equivalent and S_1 is matchable, so the last thing we need to prove is S_2's matchability. $S_2[D] = S_1[D]$ is matchable by assumption. D includes the boundary wires of D_2, so we just need to check the internal wire of D_2. But as we can push out X flips on the internal wire to become $X \otimes X$ faults, which are known to be trivial, the X flip is trivial too and cannot violate any detectors. Therefore S_2 is matchable and we conclude that $(D(D_1), S_1) \mathring{=} (D(D_2), S_2)$ is a matchability-preserving rewrite. □

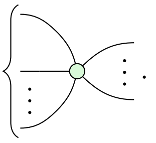
This result allows us to decompose four-legged spiders for Pauli web multisets that are not in the flower or blossom shape from Section 3.3. For the example diagram in Eq. (3.54) that we could not rewrite matchability-preservingly with the r_4 rewrite, we can now apply this alternative rewrite, assuming it occured in some diagram with a higher distance than two.



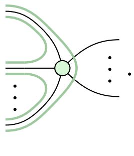
(3.62)

We can generalise this partitioning result to larger groups of Pauli webs as follows:

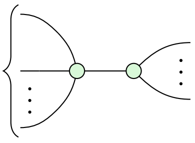
Proposition 3.27 (Generalised double-detector rewrite)**.** *Let D, D_1 be ZX diagrams with a matchable detector basis S_1 of $D(D_1)$ such that* $\mathrm{dist}(D(D_1)) > n$*,*

$D_1 = n$  . (3.63)

Furthermore, let $E_l \subseteq E(D_1)$ be n left edges of D_1 and let the following condition hold:

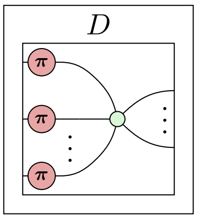
$\{\{P \in S_1[D_1] \mid \exists e \in E_l : P(e) = Z\}\} \subseteq$  . (3.64)

Then there exists a Pauli web multiset S_2 for $D(D_2)$ and

$D_2 = n$  (3.65) *such that* $S_1[D] = S_2[D]$ *and* $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_2), S_2)$ *preserves matchability.*

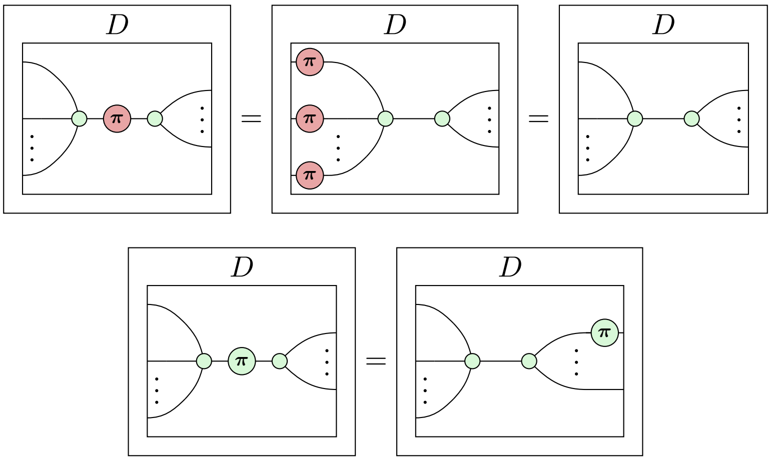
Proof. Let D, D_1, S_1 be as assumed in the claim.

We first prove that $D(D_1)$ and $D(D_2)$ are fault-equivalent in analogy to Lemma 3.25 and Proposition 3.26. For that, we notice that Eq. (3.64) ensures that each Pauli web covering the left boundary wires of D_1 in green covers exactly two of these edges in green. Therefore, the following weight-n fault is undetectable in $D(D_1)$:



(3.66)

Since $\mathrm{dist}(D(D_1)) > n$, this fault must be trivial. We can use this to prove fault equivalence between $D(D_1)$ and $D(D_2)$ by pushing out all undetectable, internal edge flips of D_1 to edges of D without increasing their weight:



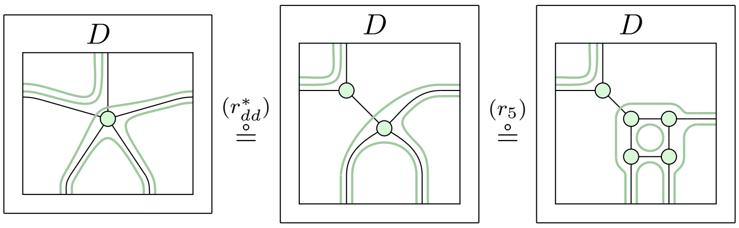
(3.67)

Second, we lift this fault-equivalent rewrite to be a matchability-preserving rewrite in analogy to Proposition 3.26. To do so, we realise that D_2 permits no local detectors, so there must exist a unique Pauli web multiset T for D_2 such that $(D_1, S_1[D_1]) \mathrel{\hat{=}} (D_2, T)$ is a detector-aware rewrite (Proposition 3.15). We apply this detector-aware rewrite to $(D(D_1), S_1)$ using Corollary 3.9 to obtain a detector basis S_2 for $D(D_2)$ such that $(D(D_1), S_1) \mathrel{\hat{=}} (D(D_2), S_2)$ is a detector-aware rewrite and $S_1[D] = S_2[D]$, $S_2[D_2] = T$ hold. $S_2[D]$ is matchable because $S_1[D]$ was matchable. The only wire in $D(D_2)$ that was not already in D is the internal wire of D_2. Any green Pauli web covering this internal wire must also cover a left and a right boundary wire of D_2 nontrivially. But Eq. (3.64) ensures that S_2 contains no such Pauli web.

We conclude that $(D(D_1), S_1) \mathrel{\overset{\circ}{=}} (D(D_2), S_2)$ is a matchability-preserving rewrite. □

Using this generalisation, we can now rewrite four-to-six-legged spiders under arbitrary matchable Pauli web coverage, assuming a global distance of two or three. This is because we found matchability-preserving decompositions for flower- and blossom-shaped Pauli web coverages in Section 3.3. Any Pauli web multiset that is in neither of these shapes can be partitioned and we can use Proposition 3.27 to "pull out" partitions under the respective distance requirement. The partitioning rewrite ensures that the partition that was pulled out now covers another spider with less legs in the flower shape that was introduced in the previous section. The other spider in the decomposition may still not be in the blossom or flower shape but it has one partition less than the original spider. Repeated applications of the partitioning rule allow us ensure that all spiders are covered in subsets of the blossom or flower shapes.

For example, we can use a partitioning and the r_4 rewrite to matchability-preservingly decompose the following five-legged spider with a partitioned Pauli web multiset into spiders with at most three legs. Here, the application of the partitioning rewrite requires the diagram to have a distance of at least two.



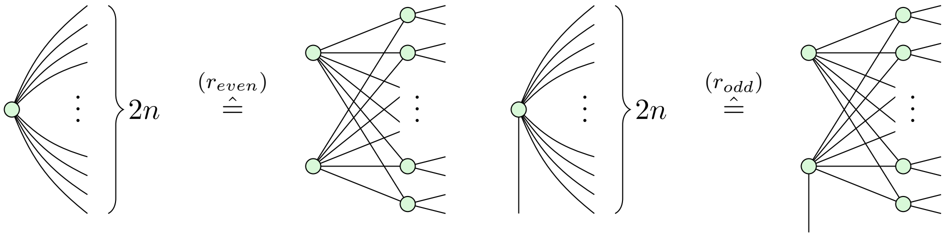
(3.68)

3.5 Decomposing many-legged spiders

In Section 3.3, we have seen how we can lift the fault-equivalent r_4, r_5, and r_6 rewrites to matchability-preserving rewrites for flower- and blossom-shaped Pauli web multisets. We have seen in Section 3.4 that we can use the "detector partitioning" trick to decompose four-to-six-legged spiders with any other matchable Pauli web coverage under reasonable assumptions. In this section, we will generalise these results to spiders with arbitrarily many legs. We will first demonstrate that Rodatz et al.'s r_{even} and r_{odd} rewrites for decomposing many-legged spiders do not generally permit any matchable Pauli web coverage. As a substitute, we present generalisations of the r_4, r_5, and r_6 rewrites, which are fault-equivalent and matchability-preserving for such Pauli web multisets.

3.5.1 r_{even} and r_{odd} do not generally preserve matchability

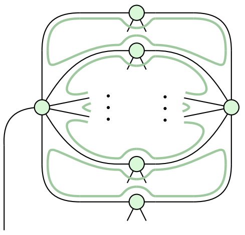
In [14], Rodatz et al. introduced the following fault-equivalent rewrites for decomposing many-legged spiders.


2n
$\overset{(r_{even})}{\hat{=}}$
2n
$\overset{(r_{odd})}{\hat{=}}$

(3.69)

Let us try to lift r_{odd} into a matchability-preserving rewrite that covers the LHS in blossom-shaped Pauli webs. The definition of detector-aware rewrites requires the Pauli web multiset for the RHS to have a local detector basis. The following Pauli web multiset is one such local detector basis.

$S_{2,det} =$

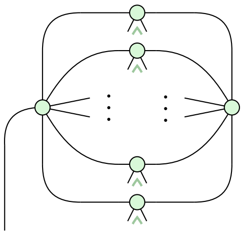


(3.70)

Note that here, we have slightly deformed the RHS of the r_{odd} rewrite to simplify the visualisation. Specifically, the middle spiders in this diagram and their boundary wires correspond to the right spiders in Eq. (3.69), and the left and right spiders in this diagram correspond to the left spiders in Eq. (3.69).

This local detector basis is minimal with respect to the number of edge coverings: Every non-trivial detecting region of this diagram covers at least four wires, and as the sample basis demonstrates, we need $n-1$ basis elements. $S_{2,det}$ reaches this minimum with $n-1$ detectors, each covering 4 edges.

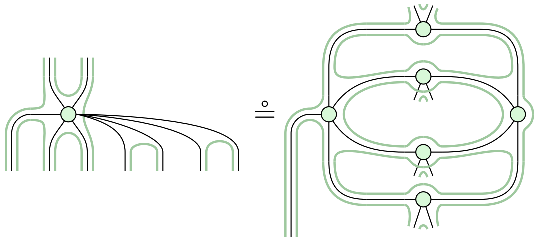
We still need to provide Pauli webs with the same boundary coverage as the blossom-shaped Pauli webs. For this, we observe that n of these can be easily implemented with the edge pairs on the n four-legged spiders. This is advantagous because they cover none of the internal wires and can therefore not be prohibitive for matchability.

$S_{2,0} =$  (3.71)

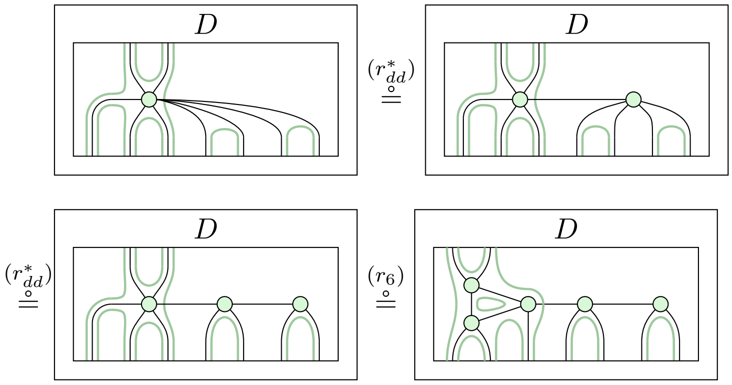
However, we are still left with $n+1$ boundary-covering Pauli webs which cannot allow for such a beneficial placement. As all of the spiders in this diagrams are the same colour as the Pauli webs, we need to "route" the Pauli webs through the diagram. Two of these routes can cover only a single internal wire (namely, these are the Pauli webs involving the leftmost spider). All remaining $n-1$ Pauli webs, however, need to cover at least two internal wires. Combining these wire coverages leaves us with the following total, which exceeds the maximum number of edge coverings allowed on the $2n$ internal wires.

$$\underbrace{4\cdot(n-1)}_{S_{2,det}}+\underbrace{0\cdot n}_{S_{2,0}}+\underbrace{1\cdot 2}_{S_{2,1}}+\underbrace{2\cdot(n-1)}_{S_{2,2}} = 6n-4 > 2\cdot 2n \qquad \text{for } n>2 \tag{3.72}$$

The above inequality forbids the existence of any matchability-preserving rewrite that uses r_{odd} as an underlying fault-equivalent ZX rewrite and has a blossom-shaped Pauli web multiset for the LHS. Note that this does not imply the nonexistence of matchability-preserving rewrites for subsets of the blossom shape. In fact, the following is a perfectly valid matchability-preserving rewrite.

 (3.73)

However, we could have simply used detector partitioning to succinctly unfuse these spiders and apply the r_6 rewrite for the flower shape in the end. For this specific rewrite sequence, the distance of the diagram would need to be higher than four.



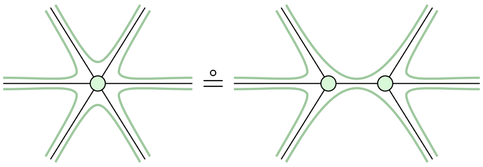
(3.74)

The problem for lifting the r_{even} and r_{odd} rewrites to matchability preservation is that there are more edge coverings required for the internal edges than are available by the matchability criterion. So if any possible detector-aware rewrite with r_{even}/r_{odd} must create unmatchable wires, one might ask whether it is possible to recover matchability using Delfosse et al.'s splitting method which we discussed in Section 2.6. We explore this question in further detail in Appendix A, where we find that splitting can be correctly applied for the r_4 rewrite, but not generally for the r_{even} rewrite.

3.5.2 Decomposing many-legged spiders through wire replication

In Section 3.3, we presented how the fault-equivalent r_4, r_5 and r_6 rewrites can be lifted to matchability-preserving rewrites. We take inspiration from the structure of these rewrites to derive $4n$, $5n$ and $6n$-legged fault-equivalent rewrites in this section. These rewrites permit matchability preservation with a similar structure to that of the r_4, r_5 and r_6 rewrite.

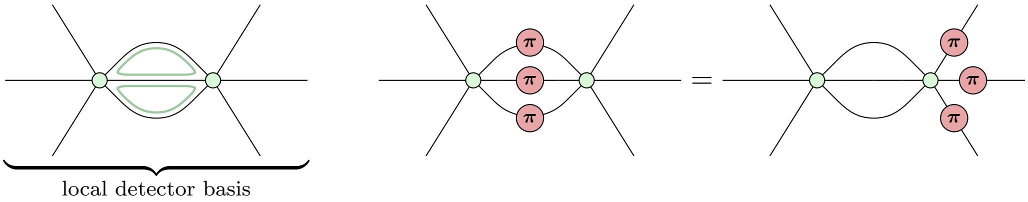
In Section 3.4, we examined a decomposition of a single, many-legged spider into two spiders that are only connected by a single wire, namely r_{dd}. This ZX rewrite was not generally fault-equivalent because a single edge flip on the internal wire of the decomposition would propagate to become multiple edge flips on the boundary wires. In our previous examination, we considered this decomposition for cases where we could separate the Pauli webs into groups. We note that this ZX rewrite can also be lifted to a detector-aware rewrite for blossom-shaped Pauli web multisets:



(3.75)

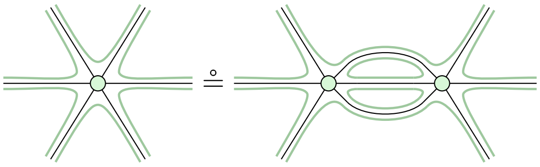
The resulting Pauli web multiset is also matchable, but since this rewrite is not generally fault-equivalent, we do not consider it a matchability-preserving rewrite.

We can mitigate this problem by replicating the internal wire which introduces pairwise detectors between the internal wires. If we use three (or more) internal wires, then the decomposition of this six-legged spider becomes fault-equivalent: A fault on the internal wires can only be undetectable if it occurs on all three internal wires, and we can push out these triples of edge flips through either spider.



(3.76)

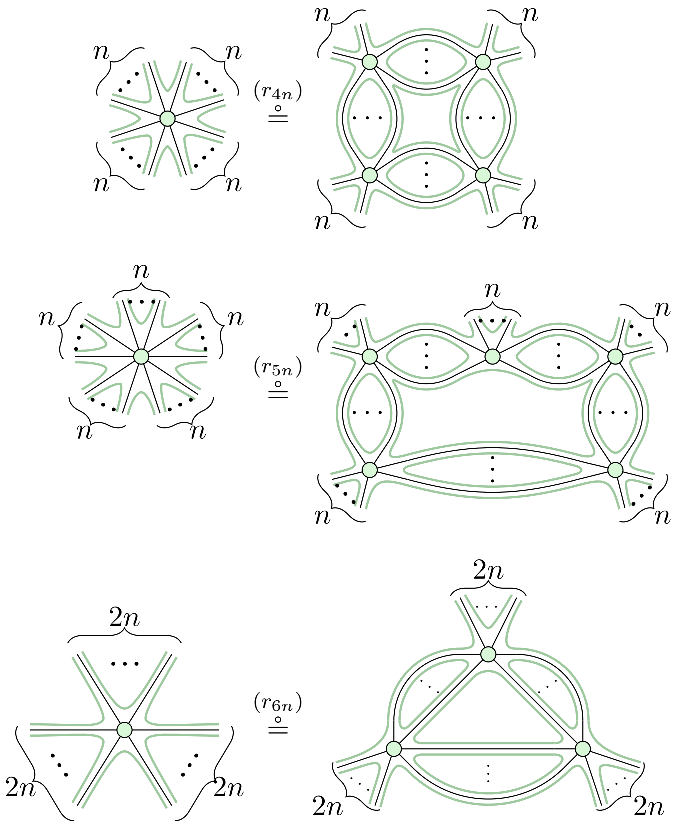
Furthermore, we can always push Z flips through the green spider onto the boundary wires and Y flips can be treated as compositions of Z and X flips. As we were able to push out all internal faults to the boundary edges without increasing their weight, the rewrite is fault-equivalent [13, Prop. A.4]. Therefore, the following is a matchability-preserving rewrite:



(3.77)

This matchability-preserving rewrite does not help us with our goal of decomposing spiders because the two decomposed spiders have the same number of legs and the same detector coverage as the original spider. However, we can apply this trick of replicating wires to the r_4, r_5 and r_6 rewrites. We prove this in the following proposition.

Proposition 3.28 (r_{4n}, r_{5n} and r_{6n} preserve matchability)**.** *The following three rewrites* *preserve matchability:*


n
n
n
n
$\overset{(r_{4n})}{\overset{\circ}{=}}$
n
n
n
n
n
n
n
n
n
$\overset{(r_{5n})}{\overset{\circ}{=}}$
n
n
n
n
n
2n
2n
2n
$\overset{(r_{6n})}{\overset{\circ}{=}}$
2n
2n
2n

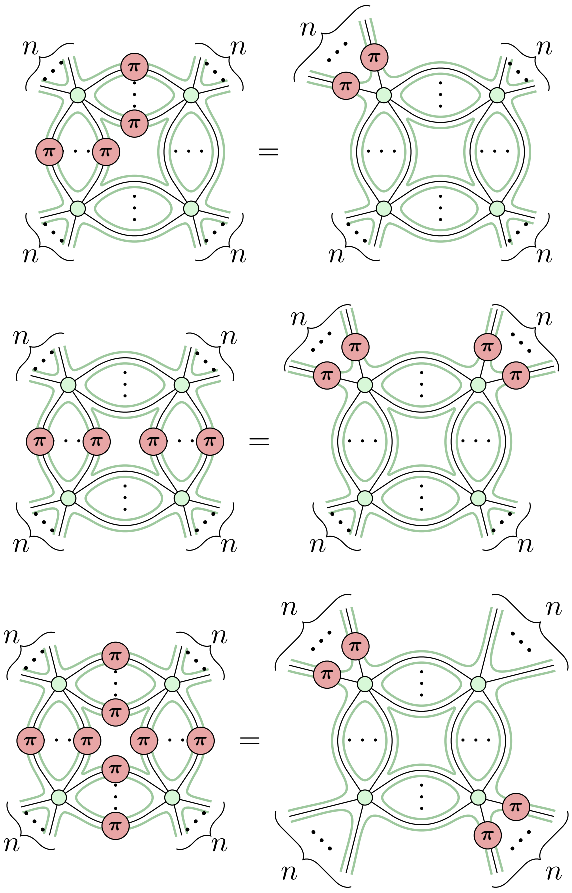
(3.78)

Here, the dots on the internal wires refer to bundles of n *wires with local detectors in between them.*

Proof. We prove this result explicitly for r_{4n}. Let us first prove fault equivalence by pushing out undetectable faults [13, Prop. A.4]. Let F be an undetectable fault on the internal wires of the RHS.

For F to be undetectable all of the replicated wires must either be X-flipped on either no wire or all wires, similar to our reasoning for Eq. (3.76). F is therefore a composition of F_T, F_R, F_B and F_L where F_T contains all n X flips of the replicated top wire, and the others contain all the flips of the right, bottom, or left wire, accordingly.

We can then argue for r_{4n}'s fault equivalence similar to the original proof in [13], considering the aforementioned composed faults instead of individual edge flips.



(3.79)

Here, in the first equation, we halve the fault weight from $2n$ to n, in the second equation we maintain the weight of $2n$ and in the third equation we halve the weight from $4n$ to $2n$. By symmetry of the diagram, this covers all undetectable, internal X faults. Furthermore, all spiders are green and we can therefore freely move all Z edge flips to the boundary without increasing their weight. We decompose Y edge flips to Z and X edge flips and move the Z flips along with the X flips to maintain the fault's weight.

The proofs for r_{5n} and r_{6n} are analogous to the proofs for r_5 and r_6 in the same way that our proof for r_{4n} was analogous to r_4's proof from [13]. □

Using these rewrites, we can decompose arbitrary spiders whose leg counts are multiples of 4, 5 or 6. For any other number of legs, we can use the r_{fuse} rule in reverse to increase the leg count such that it is divisible by 4, 5 or 6. We have proven that r_{fuse} preserves matchability in Section 3.2.

Chapter 4

Examples

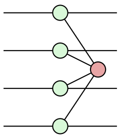
In this chapter, we demonstrate how matchability-preserving rewrites can be used to build fault-tolerant implementations for surface code plaquette measurements that are still efficiently decodable through the MWPM decoder. We first consider a fault-tolerant implementation of the Z^4 stabiliser measurement that we can derive straight-forwardly from matchability-preserving rewrites. Furthermore, we present an improved variation of Grans-Samuelsson et al.'s hook-resilient plaquette measurements for Majorana hardware [36].

4.1 Surface code plaquette measurements

In this section, we demonstrate how to synthesise a fault-tolerant and matchable implementation of surface code plaquette measurements. More specifically, we derive such an implementation for the Z^4 stabiliser measurement, surrounded by all possible boundaries in the rotated surface code. This will also yield an implementation for X^4 measurements, which are just duals of Z^4 measurements.

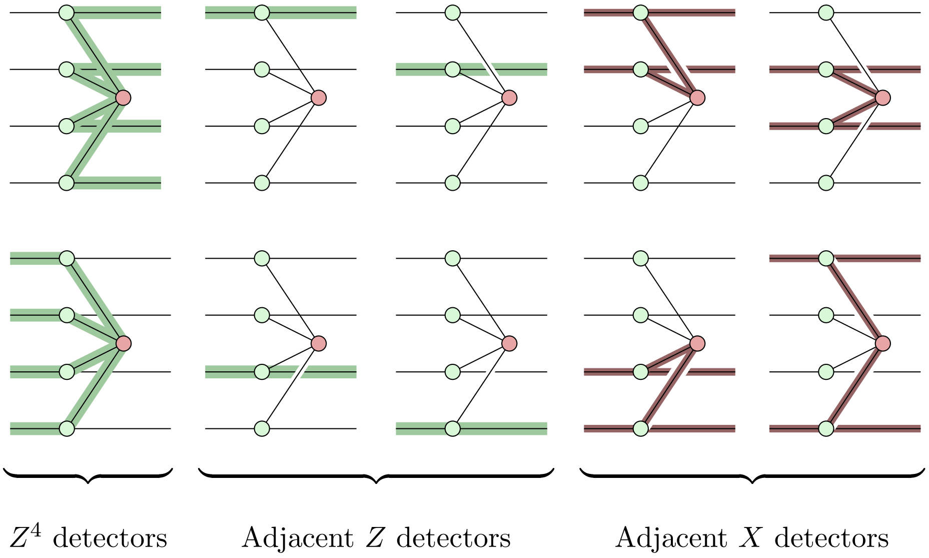
In [14, 13], Rodatz et al. reasoned that a fault-equivalent ZX representation of the Z^4

measurement is as follows[1]:



(4.1)

In the surface code, the four target qubits of a plaquette measurement can be covered by plaquette or boundary stabilisers as follows: For each (non-diagonally-) adjacent pair of qubits, there can be at most one X stabiliser covering this pair of qubits and for each qubit, there can be at most one other Z stabiliser covering this qubit. If we represent all of the corresponding detecting regions as a Pauli web multiset covering this measurement, we observe that they also form a matchable detector set under the edge flip noise model if we ignore Y flips as in Proposition 2.34:

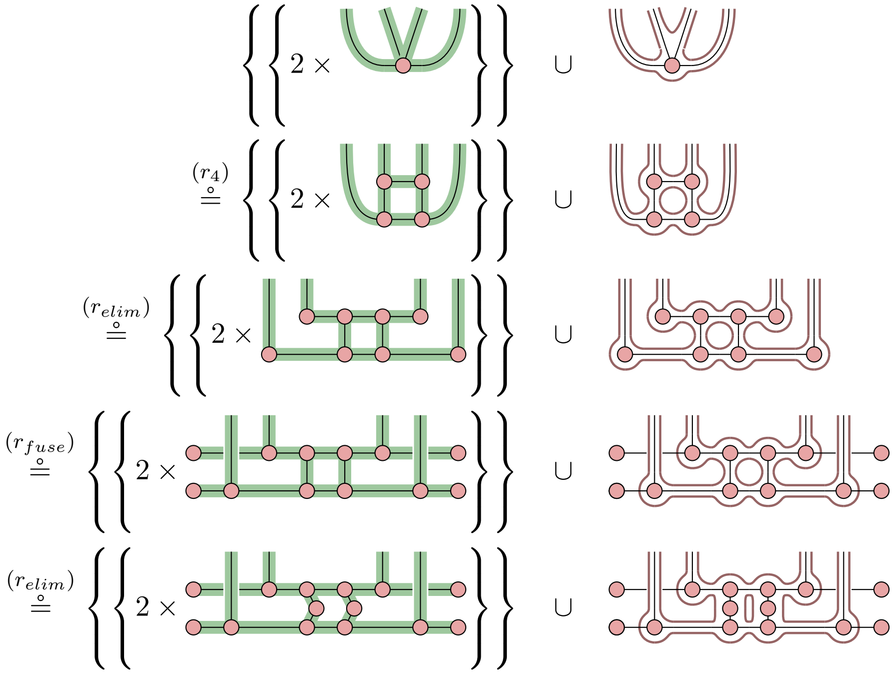


(4.2)

To turn the ZX diagram with a four-qubit measurement into one that only measures single qubits and pairs of qubits, we succinctly apply our matchability-preserving rewrite rules from Chapter 3 to high-weight spiders until all spiders in the diagram have at most three

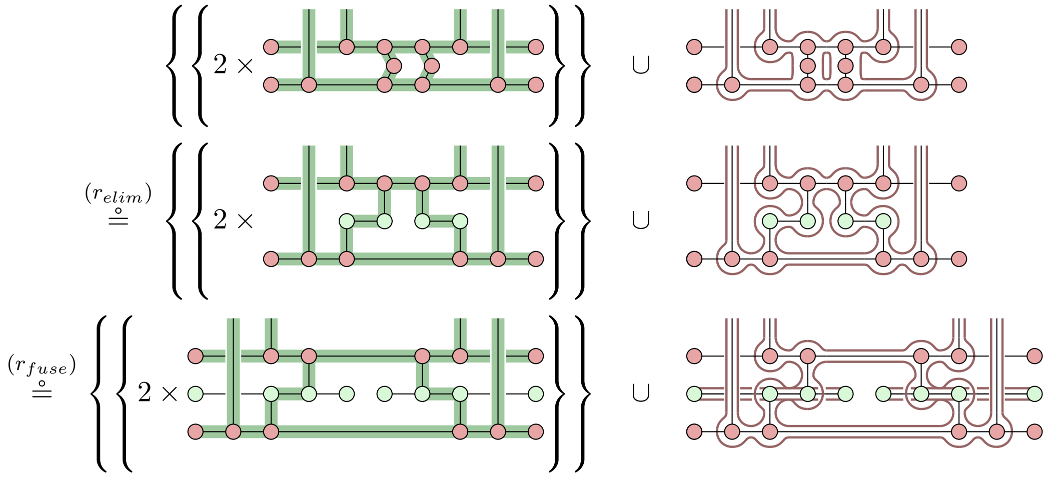
[1] This measurement is post-selected onto the deterministic outcome that will be measured on the codespace in the absence of noise, following [14]. Measurement-preserving synthesis of a Pauli measurement implementation without post-selection is also possible but it is more involved. A fault-equivalent decomposition is presented in [13, Sec. 7.2] but further fault-equivalent rewrites need to be lifted to matchability preservation to show matchability preservation.

legs:



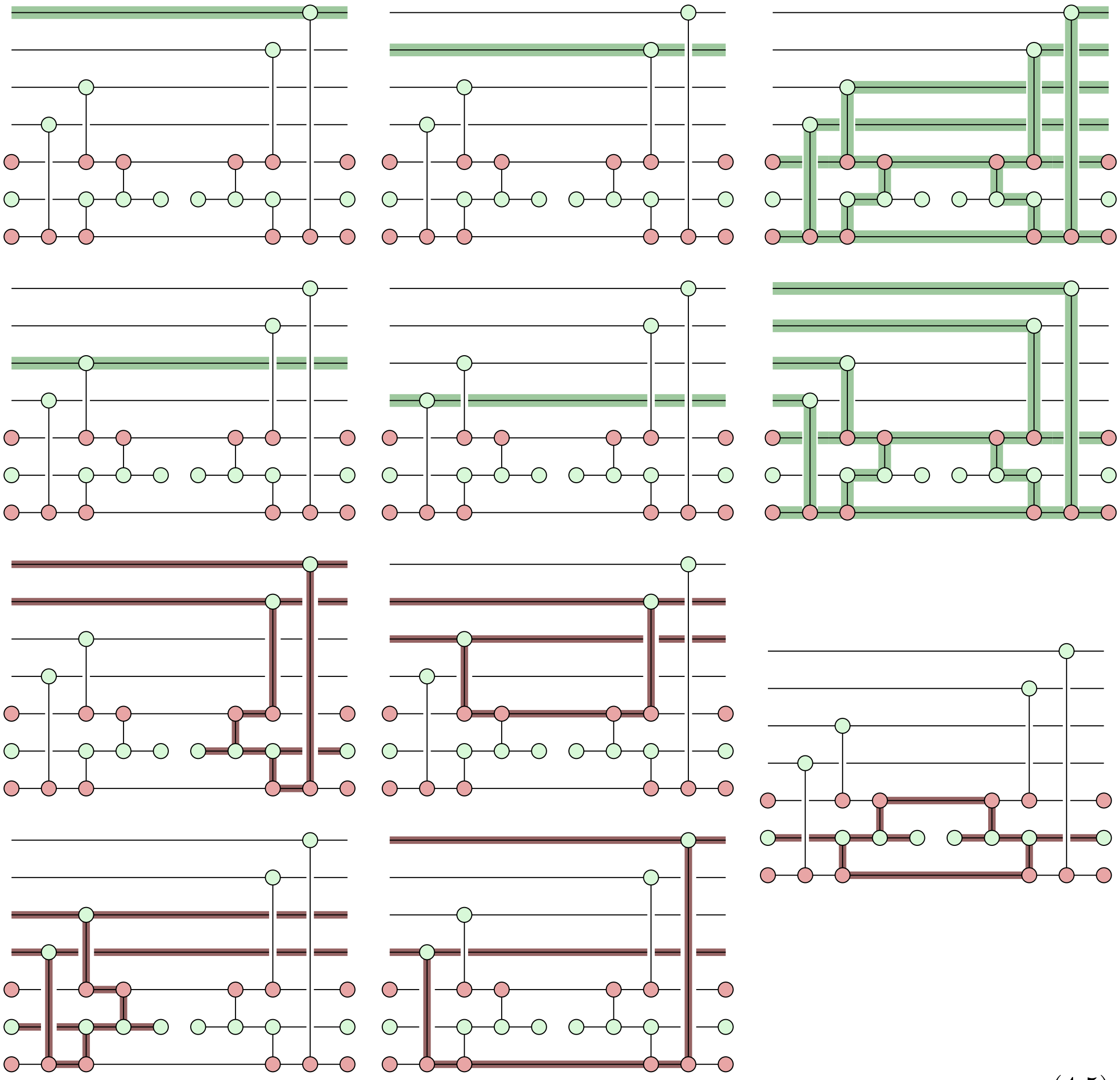
(4.3)

If our gate set supports XX measurements, we are already done. However, this this is not a common gate to support (for example, IBM's quantum computers do not support it [40]), so we can apply further steps to turn these two-qubit-measurements into single-qubit measurements.



(4.4)

Note that in the last rewrite of the X detectors multiset, our notation has gotten a bit convoluted on the three-legged, green spiders. This is because the drawn Pauli webs need to cover all three legs of these green spiders. At this point, our Pauli web multisets have outgrown our notation, and we resort to standard Pauli web drawings to show the resulting Pauli web multiset after applying our rewrite chain to the measurement diagram with our original detector basis.



(4.5)

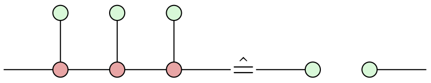
4.2 Improved plaquette measurements

In their paper *"Improved Pairwise Measurement-Based Surface Code"*, Grans-Samuelsson et al. present an improved, fault-tolerant implementation for surface code measurements [36]. Their implementation is designed to work particularly well on Majorana hardware, but its design has favourable properties on any hardware aligning physical qubits on a square grid. For instance, their hook-error-resilient implementation [36, Sec. 3] for any plaquette and boundary measurement uses a circuit of depth at most seven gates. These measurement implementations interlock with measurements of adjacent plaquettes and boundaries so that all stabilisers can effectively be measured in parallel within seven time steps[2].

We study this specific example in this section because the hook-resilient measurement implementation of this paper produced an error-detector model that was not matchable anymore. Even though Grans-Samuelsson et al. successfully recovered matchability through Delfosse et al.'s splitting methodology [11], this example demonstrates that it is difficult to preserve matchability in ad-hoc circuit rewriting. In the following, we will show that the hook-resilient Z^4 measurement implementation can be recovered with matchability-preserving rewrites, and we will be able to read off a matchable detector basis for this implementation.

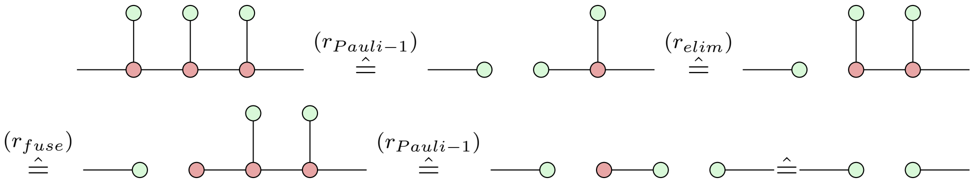
Before we get to the main part of this section, we prove for one more rewrite that it satisfies the matchability preservation criteria. Here, we extend this result to the triple repetition of the same measurement:

Lemma 4.1 (Triple measurement fault equivalence)**.**



(4.6)

Proof. We prove



(4.7)

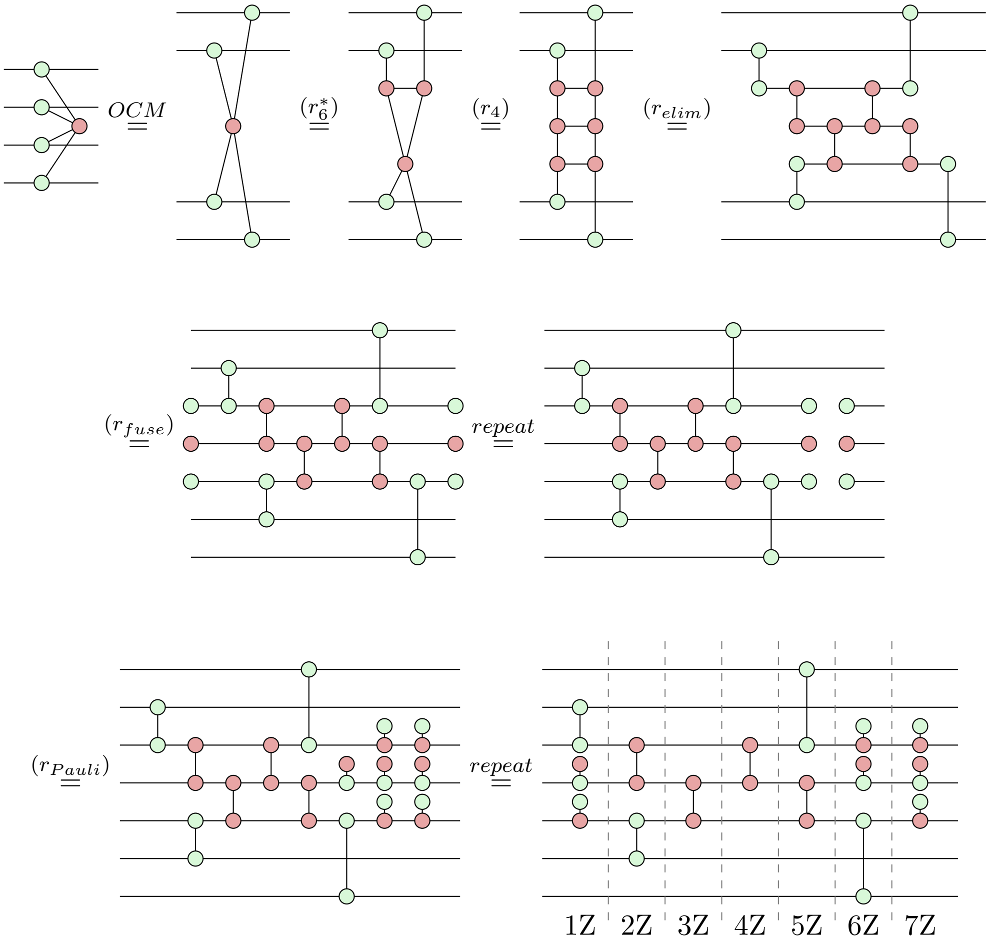
□

While a triple measurement is the largest measurement repetition we need for the purpose of this section, we would like to point out that Lemma 4.1 also holds for an arbitrary number $n \geq 2$ of repetitions of the same measurement. For an even number n of repetitions, we just apply $r_{Pauli-1}$ $n/2$ many times and for an odd number n of repetitions,

[2] Technically, this requires an offset between adjacent plaquette and boundary measurements. But as these measurements will need to be repeated many times in Shor-style syndrome extraction [13, Sec. 7.2.1], the constant-depth ramp-up and ramp-down phases can be amortised within the full syndrome extraction.

we apply the triple measurement rewrite once and then treat the even remaining number of measurements as mentioned before.

This brings us to the main result of this section:



(4.8)

Note that we used one special rewrite which we dubbed *repeat*. With this rewrite, we rolled some of the states and measurements over the periodic boundary of this measurement. If this circuit is sandwiched between multiple instances of itself, it is immediately clear that this is a valid transformation. If the shown circuit is the first or last in a sequence of measurements, then it is preceded/followed by a ramp-up/ramp-down phase, which also contains the respective measurements. For further details on this, we refer to Grans-Samuelsson et al.'s paper [36, Sec. 3].

From the rewritten diagram, we can now read off a matchable basis of detectors. This allows us to avoid having to split the noise model of the circuit.

Chapter 5

Discussion

In this work, we introduced *matchability-preserving rewrites*, a stricter notion of ZX rewrites that preserves the noise model and rewrites the detector basis along with the diagram in such a way that the detector basis remains matchable. We have motivated, defined and explored the properties of this novel type of rewrite rigorously in Chapter 3 and applied them to derive matchable, fault-equivalent surface code plaquette measurement implementations in Chapter 4. In this final chapter, we discuss the limitations and potentials of matchability-preserving rewrites.

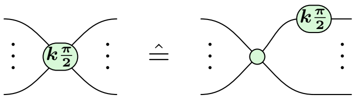
5.1 Circuit and detector limitations

The applicability of matchability-preserving rewrites is currently limited to certain types of circuits and codes.

First, our methodology is limited to the Clifford fragment of the ZX calculus, which is known to be classically simulable [41]. This limitation comes directly from our Pauli web use, as Pauli webs themselves are only defined for the Clifford fragment [15]. Unfortunately, it is unlikely that a Pauli-web-esque structure can be generalised beyond the Clifford fragment. This is because Pauli webs correspond to the way in which we can propagate Pauli flips through the diagram, as explained in Section 2.4. Pushing a Pauli error through a non-Clifford spider of a different colour can, however, result in a non-Pauli error. Any generalisation of Pauli webs would need to take this into account.

Many areas of quantum computing face the problem of generalising approaches beyond the Clifford fragment. For example, Duncan et al. derive an asymptotically optimal circuit extraction procedure for Clifford ZX diagrams in [32]. To extract circuits beyond the Clifford fragment, they suggest cutting the diagram into Clifford and non-Clifford slices to perform circuit extraction on them separately and to combine the resulting circuit parts. Similarly, some approaches to FTQC externalise non-Clifford states and gates to be prepared as so-called *magic states* and then combined into the quantum state of the program using Clifford gates [42]. Future work could explore to what extent Pauli webs can be used to capture redundancies in such states, and if detector-aware or matchability-preserving rewrites can be adapted to such scenarios.

Second, we have focused on the decomposition of phaseless (or $k\pi$-phased) spiders, since most fault-equivalent rewrites from Rodatz et al.'s original presentation of fault equivalence were focused on phaseless spiders [13, 14] and as this was sufficient for decomposing stabiliser measurements. Fault equivalence is, in principle, defined on the entire Clifford fragment and [13]'s appendix features the following fault-equivalent rewrite for unfusing phases from spiders:



(5.1)

Detector-aware and matchability-preserving rewrites are by definition flexible enough to extend to Clifford rewrites. However, matchability is in practice often only achieved under the assumption that Y faults are splittable, i.e. we can treat X and Z faults separately. As the full Clifford fragment allows for Hadamard gates, we can no longer make this assumption since Hadamard gates map between X and Z spiders.

Finally, the most significant limitation of matchability-preserving rewrites is that they require matchability of the detector basis to begin with. Although the surface code and its many variants are prominent examples within these limitations, this requirement prohibits the application to other, more efficient codes [7]. Furthermore, even for matchable codes, we might run into scenarios that require detector partitioning but do not satisfy the respective requirements, such as a sufficient distance or stabilising context. While we did not find examples of actual codes where the matchability-preserving decomposition would run into such a problem, we cannot guarantee at this time that finding such problematic codes is impossible.

5.2 Noise model limitations

For the purpose of this work, we have limited our scope to adversarial noise models. Under adversarial noise models, we analysed QEC codes and ZX diagrams with respect to their distance, that is the ability to detect and correct all faults whose weight is below a fixed number. Analysing QEC codes with respect to their distance is very common in the literature [27] because it provides an analytical tool to assess error correction capabilities.

Although the physical processes that cause noise rarely admit such adversarial modelling, adversarial noise is an approximation of the stochastic nature of noise. If we consider atomic faults to be stochastically independent, then events where many atomic faults happen at the same time (i.e. high-weight faults) are increasingly unlikely. Thus, for a fixed stochastic noise model and some target probability p, we could compute a minimal distance such that the likelihood of an undetectable (or similarly, uncorrectable) fault occurring is below p.

This line of reasoning is appropriate for a fixed noise model, but in the context of fault-equivalent rewrites, the noise model is not fixed. In fact, many fault-equivalent rewrites will be applied in an order that introduces new possible sources noise by increasing the diagram size. As we apply fault-equivalent rewrites, the distance remains constant, but the probability of an uncorrectable error might change.

In this thesis, we considered matchability-preserving rewrites as a specialisation of fault-equivalent rewrites, so the same critique also applies to the methods we presented. In a parallel work, Maximilian Rüsch generalised the notion of fault equivalence to allow for weighted faults which could ultimately resolve the aforementioned issue [29]. Therefore, future work on matchability preservation should generalise matchability-preserving rewrites to work with stochastic, fault-equivalent rewrites. Alternatively, the trade-off between preserving matchability and introducing new sources of noise can be examined with empirical means, for example through simulation [12]. Such an empirical analysis could also be beneficial for asserting the effectiveness of splitting for decompositions of large spiders using the r_{even} and r_{odd} rewrites, as discussed in Appendix A.

5.3 Future work

In this section, we explore some avenues for future work related to matchability preservation. We first present some direct continuations of this project and then discuss a few further ideas in the general direction of efficient decodability preservation.

In Section 3.1, we generalised the diagrammatic Pauli web notation to visualise a multiset of Pauli webs compactly in the same diagram. This notation helped us visualise isolated rewrites in Chapter 3, but even for the surface code plaquette measurement implementation in Chapter 4, it quickly became convoluted, and therefore, less helpful. Rewriting larger diagrams and their detector bases would therefore greatly benefit from computer tooling like ZXLive [43].

Similarly, adding support for detector rewriting and matchability-preserving rewrites to the underlying PyZX library [44] could enable the automated compilation of matchable ZX diagrams to fault-equivalent and matchable implementations [1].

Finally, future work should consider notions of efficient decodability beyond matchability. The most obvious such extension is "Union-Find decoding" [45], which has similar requirements as MWPM decoding. Namely, this is the *matchability* (a.k.a. *graph-likeness*) property which ensures that the error-detector model can be reformulated as a *matching graph* (a.k.a. *decoding graph*). As Union-Find decoding solves the minimum-weight decoding problem only approximately, future work could consider how the changes introduced by matchability-preserving rewrites affect the Union-Find decoder's performance.

More generally, future work could consider *decoding reductions*, that is, decoders that reduce the decoding problem of the rewritten diagram to the efficiently solvable decoding problem on the original diagram. We have briefly discussed similar reductions between unmatchable noise models and split noise models in Appendix A, and we have pointed out how detector-free, fault-equivalent rewrites keep the error-detector model intact in Section 3.2. Such reductions avoid the need to preserve some concrete, structural property of the error-detector model and instead rely directly on the fact that there is an efficient decoder, which makes them more generally applicable.

[1] First steps towards fault equivalence support in PyZX have recently been made in [29].

5.4 Conclusion

Designing Quantum Error Correction codes for arbitrary distances is a complex task on its own and it has been the subject of diverse research over the past three decades. Building and compiling implementations for these codes for specific gate sets is another non-trivial but crucial step towards fault-tolerant quantum computation. In this thesis, we expanded on Rodatz et al.'s previous work on fault-equivalent rewrites [13] and put focus on preserving matchability, which guarantees efficient minimum-weight decoding of, for example, surface codes. More specifically, we presented *matchability-preserving rewrites* as an extension of fault-equivalent rewrites that track the changes ZX rewrites impose on the detector basis of the diagram and ensure that such rewrites cannot introduce unmatchable atomic faults. Our methodology can be used to compile decoder-efficient and fault-tolerant implementations of surface code stabiliser measurements, and it is extendable to other matchable stabiliser codes.

Matchability-preserving rewrites are currently limited to the Clifford fragment due to their dependence on Pauli webs and fault equivalence. This reveals a natural direction for future work in the generalisation of these concepts beyond the Clifford fragment, but this relies on the generalisation of Pauli webs and fault-equivalent rewrites. Furthermore, it will be interesting to consider the efficiency of matchability-preserving rewrites under stochastic noise models either explicitly through some stochastic variant of fault equivalence or through empirical analysis. Finally, we note that while matchability-preserving rewrites are still somewhat easy and intuitive to track on small examples, the analysis of larger circuits will benefit from automated tooling.

As the name suggests, the study of matchability preservation is limited to settings where we have a matchable error-detector model to begin with. Efficient decoding may also be studied in a more general setting where we assume some efficient decoder exists and use ZX rewrites in such a way that the decoding problem for the resulting diagram can be efficiently reduced to the efficient decoder of the original diagram and noise model. From this perspective, we hope that this work serves as a starting point for the study of efficient decoding preservation.

Appendix A

Incomplete splitting formalisation

We have explored in Section 2.5 that the decoding problem for arbitrary error-detector models is a hard one, and that matchable error-detector models form a subclass of this problem that can be solved efficiently. In Section 2.6, we briefly discussed *splitting*, a method introduced by Delfosse et al. [11] for turning unmatchable error-detector models into matchable ones. This is achieved by removing *non-primitive* (including unmatchable) faults from the fault set and distributing the removed fault's probability to a set of atomic faults that compose to the removed fault. For adversarial noise models, splitting boils down to removing composeable, atomic faults from the fault set.

In this section, we explore some properties of the splitting method. We first formalise the relationship between faults of the split decoding problem with the faults of the original decoding problem, with special attention given to the fault weights and distances of these error-detector models. We use this relationship to compare solutions between decoding problems on these models. With this language, we can express when decoding problems on unmatchable error-detector models can be reduced to solutions of the corresponding split models. Finally, we apply this formal method to examples from the main text of this thesis to see that splitting can sometimes guarantee correct decoding and that it fails in other cases.

A.1 Solutions and corrections

In this section, we expand our formalisation of the decoding problem. We categorise faults as *solutions*, *optimal solutions* and *correct solutions* and define what it means for a fault to be correctable.

For decoding in stochastic noise models, we would want to find (some fault from) the fault equivalence[1] class with the highest probability that produces this syndrome. For adversarial noise models, this problem translates to finding the lowest-weight fault equivalence class that triggers this syndrome.

In non-degenerate error-detector models, that is error-detector models where each syndrome is associated with at most one fault, this problem is equivalent to finding the lowest-weight fault triggering the syndrome. For general error-detector models, however, finding the optimal solution is a $\#\mathcal{P}$-complete problem and the single fault with the highest probability or lowest weight is often used as an approximation [46].

This approximation is used in many efficient decoders, including MWPM decoders [9] and we will therefore analyse in this chapter how the solution to this relaxed problem changes though an application of the splitting method. We begin by establishing a vocabulary that we will use throughout this chapter.

Definition A.1 (Solutions, optimal solutions, correctability). *Let T be a Tanner graph for a noise model $\mathcal{F}$ and a detector basis $\mathcal{D}$. Let $F \in \langle\mathcal{F}\rangle$ be a fault and $s = syn_T(F)$ be its syndrome.*

- *We call $\bar{F} \in \langle\mathcal{F}\rangle$ a* solution for s under $\mathcal{F}$ *if $syn_T(\bar{F}) = s$.*
- *We call a solution $\bar{F}$ for s* optimal *if its weight is minimal among the solutions for s under $\mathcal{F}$.*
- *We call F* correctable *if all optimal solutions $\bar{F}$ of s are equivalent to F.*

Generally, decoders should realise a *solution function syn* : $\mathrm{Im}(syn_T) \to \langle\mathcal{F}\rangle$ where each syndrome s is mapped to a solution for s under the respective noise model. Applying

[1] In Section 2.3.1, we defined two faults $F_1, F_2 \in \langle\mathcal{F}\rangle$ to be equivalent if $D^{F_1} = D^{F_2}$ holds. This condition is too loose for our consideration here since for every detectable fault $F \in \langle\mathcal{F}\rangle$, $D^F = 0$ holds and therefore, all detectable faults would be considered equivalent. A stricter notion of equivalence that can tell some detectable faults apart is developed in [29] in parallel to this work. For this appendix chapter, we hope that this stronger notion of is clarified through the examples we give.

a solution to a circuit ensures that the fault and solution cancel out the syndrome measurements. An *optimal solution function* additionally requires all produced solutions to be optimal, that is, to have minimal weight. Minimum weight decoders therefore realise optimal solution functions.

Solutions, and even optimal solutions are often not unique for several reasons: First, there can be equivalent faults in a noise model but for a successful correction, we do not need to distinguish between these. Second, there can be nonequivalent faults with the same syndrome. These faults are indistinguisable to the decoder and they give rise to decoding errors.

We stated an example for correctable faults in Proposition 2.22: All faults whose weight is below half the distance of an error-detector model had a unique syndrome and they were therefore correctable.

A.2 Splitting reductions

In the previous section, we have introduced the terminology to talk about solutions and corrections. We now explore what it means to reduce different error-detector models onto another.

The goal of a reduction is to solve an instance of one problem by compiling it into one or many instances of another problem that we know how to solve, and reconstructing a solution to the original problem from the solutions to the constructed instances. One such reduction can be found in [10], where Nicolas Delfosse reduced instances of the colour code decoding problem to several instances of the surface code decoding problem. In this section, we focus on decoder reductions in the context of splitting.

For the remainder of this section, let $\mathcal{F}$ be a noise model and let $F \in \mathcal{F}$ be an atomic fault such that $F \equiv F_1 \cdots F_n$ for atomic faults $F_1, \ldots, F_n \in \mathcal{F} \setminus \{F\}$. Then $\mathcal{F}' := \mathcal{F} \setminus \{F\}$ is a splitting of F in $\mathcal{F}$.

Let us first assert some basic facts about the relationship between $\mathcal{F}$ and $\mathcal{F}'$. We already know $\mathcal{F}' \subseteq \mathcal{F}$ but as we removed only an atomic fault that was equivalently composeable from other atomic faults that remain in the noise model, the fault group remains unchanged, up to equivalence.

Lemma A.2 (Splitting preserves the fault group)**.** *For the original and the split noise model, we have*

$$\langle\mathcal{F}\rangle \equiv \langle\mathcal{F}'\rangle \tag{A.1}$$

Here, the equivalence between sets means that each element of one set has an equivalent element in the other set and vice versa.

Proof. We assume $\mathcal{F}, \mathcal{F}', F \equiv F_1 \cdots F_n$ as specified in the preceeding paragraph and prove this equation for both directions.

"$\subseteq$" Let $G_1 \cdots G_m \in \langle\mathcal{F}\rangle$ with atomic faults $G_1, \ldots, G_m \in \mathcal{F}$ be an arbitrary fault. For $i \in \{1, \ldots, m\}$, we define

$$G'_i = \begin{cases} G_i & G_i \not\equiv F \\ F_1 \cdots F_n & G_i \equiv F \end{cases} \tag{A.2}$$

where $G_i \equiv G'_i \in \langle\mathcal{F}'\rangle$ holds although G_i is always an atomic fault and G'_i may be composed of several faults in $\mathcal{F}'$. Therefore, $G_1 \cdots G_m \equiv G'_1 \cdots G'_m \in \langle\mathcal{F}'\rangle$.

"$\supseteq$" This direction is an immediate consequence of $\mathcal{F} \supseteq \mathcal{F}'$.

□

This observation already allows us to relate the solutions to each other. For the remainder of the text, let $\mathcal{D}$ be a detector basis, let T be the Tanner graph for $\mathcal{F}$ and $\mathcal{D}$ and let T' be the Tanner graph for $\mathcal{F}'$ and $\mathcal{D}$.

Lemma A.3 (Splitting preserves solutions)**.** *Every solution $\bar{F} \in \mathcal{F}'$ for some syndrome $s \in \mathrm{Im}(syn_{T'})$ under $\mathcal{F}'$ is also a solution for s under $\mathcal{F}$.*

Proof. Any solution $\bar{F} \in \langle\mathcal{F}'\rangle$ is also a fault under $\mathcal{F}$, that is, $\bar{F} \in \langle\mathcal{F}\rangle$. Furthermode, as T and T' are based on the same detector basis $\mathcal{D}$, we must have

$$syn_T(\bar{F}) = syn_{T'}(\bar{F}) = s \tag{A.3}$$

and therefore, $\bar{F}$ is a solution for s under $\mathcal{F}$. □

The preservation of solutions is a nice property, but unfortunately, this property does not generally hold for optimal solutions. While the fault groups and solutions for the original and the split noise models are equal up to equivalence, the minimum-weight composition of faults may change due to the removal of an atomic fault. We make this more precise in the following two lemmas.

Lemma A.4 (Splitting cannot decrease fault weights)**.** *For any fault $F \in \langle \mathcal{F} \rangle \equiv \langle \mathcal{F}' \rangle$, the following weight monotonicity holds:*

$$\mathrm{wt}_{\mathcal{F}}(F) \leq \mathrm{wt}_{\mathcal{F}'}(F) \tag{A.4}$$

Proof. Let $F \in \langle \mathcal{F} \rangle \equiv \langle \mathcal{F}' \rangle$ be a fault. There exists a representation $F = F_1 \cdots F_n$ where $F_i \in \mathcal{F}'$ are atomic faults in the split noise model and $n = \mathrm{wt}_{\mathcal{F}'}(F)$ is the weight of the fault in the split noise model. But since $\mathcal{F}' \subset \mathcal{F}$, $F_i \in \mathcal{F}$ also holds and therefore $wt_{\mathcal{F}}(F) \leq n = wt_{\mathcal{F}'}(F)$. □

This monotonicity may at first be counterintuitive because we are removing a fault from $\mathcal{F}$ to get $\mathcal{F}'$. The right intuition about this is that atomic faults which are equivalent to composed faults, can be used as a “shortcut” to build an equivalent fault with less weight. Consequently, removing a shortcut can increase the weight required to compose some specific fault.

The potential increase in fault weights has in important consequence regarding the distances of the original and the split error-detector models.

Corollary A.5 (Splitting cannot decrease the distance)**.** *For the distances of the original and the split error-detector models, the following monotonicity holds:*

$$\mathrm{dist}(T) \leq \mathrm{dist}(T') \tag{A.5}$$

Proof. Let $(\mathcal{F}, T)$ be the original noise model and let $(\mathcal{F}', T')$ be the split noise model. There must exist an undectectable fault $F \in \langle \mathcal{F}' \rangle$ with $wt_{\mathcal{F}'}(F) = \mathrm{dist}(T')$ in the split noise model. Using the previous lemma, we can conclude

$$\mathrm{dist}(T) \leq wt_{\mathcal{F}}(F) \leq wt_{\mathcal{F}'}(F) = \mathrm{dist}(T)\,. \tag{A.6}$$

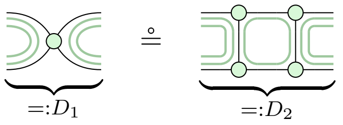
□

A.3 Examples

In this section, we analyse two examples from Chapter 3 where fault-equivalent rewrites produced an unmatchable detector basis from a matchable one. The first example is an application of the r_4 rewrite to a spider with double detecting regions and it demonstrates a successful application of splitting. The second example is the r_{even} rewrite on a spider with a blossom-shaped Pauli web multiset and we demonstrate that splitting can compromise the distance in this case.

Successful application of splitting for r_4

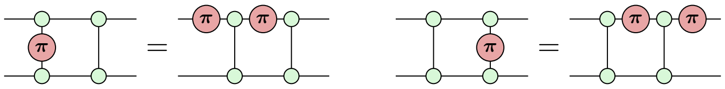
In Chapter 3, we often came across four-legged spiders with matchable Pauli web multisets that would produce an unmatchable Pauli web multiset through the application of the fault-equivalent rewrite r_4. One of these examples is the following rewrite:



(A.7)

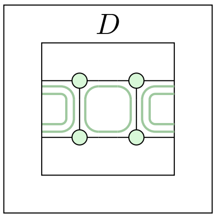
In this example, the X flips on the vertical wires of the rewritten diagram are covered with three green Pauli webs from the multiset, making it unmatchable. This naturally brings up the question whether we can recover matchability through the application of splitting.

Clearly, the X flip on the left wire can be decomposed into two atomic X faults on the top-left and top-middle wire and the X flip on the right wire can be decomposed accordingly:



(A.8)

We use this to specify a split noise model $\mathcal{F}'$ where the decomposed X flips do not exist. To check whether we can successfully reduce decoding problems for the original noise model to the split noise model, we need to check whether their optimal solutions coincide. For that, we need to consider the full ZX diagram that needs to be decoded, so let $D(D_2)$ be an arbitrary ZX diagram with D_2 as a subdiagram, and let S be a detector basis such that $S[D_2]$ is as in Eq. (A.8).



(A.9)

Note that this diagram does not have any boundary wires because those would always allow for undetectable atomic faults [13, Sec. 2.2]. Instead, we assume that this diagram already includes all necessary state preparation and measurements.

Now let $F \in \langle \mathcal{F} \rangle$ be an arbitrary on $D(D_2)$ that is correctable by distance and let $\bar{F} \in \langle \mathcal{F}' \rangle$ be an optimal solution for $syn_T(F)$ in $\mathcal{F}'$.

Case 1: If F did not involve either of the split faults, then $\mathrm{wt}_{\mathcal{F}'}(F) = \mathrm{wt}_{\mathcal{F}}(F)$ and as fault weights are nondecreasing (Lemma A.4), F is an optimal solution in $\mathcal{F}'$ and unique up to equivalence.

Case 2: If F did not violate the square detecting region, then by the fault-equivalence of the r_4 rewrite, we can push out all edge flips from the internal wires of D_2 to the boundary wires of D_2 to get an equivalent fault with the same or less weight. But then by Case 1, this equivalent fault is also correctable in the split noise model.

Case 3: If F violated the square detecting region, then it must have involved an odd number of X edge flips on the internal wires of D_2. Without loss of generality, let the number of X edge flips on the internal wires of D_2 be one because otherwise, we could have pushed out a pair of X flips to the boundary edges of D_2 without increasing their weight (see [13, Prop. 6.14]).

If the one X flip is neither of the split faults, we are already done by Case 1. Otherwise, assume that the left wire was X-flipped (for the right wire, the argument is symmetric).

We push this X flip through the top left spider to get an equivalent fault F' whose weight has either remained the same or increased by one in $\mathcal{F}'$. If the weight remained the same, then F' has the same weight in $\mathcal{F}$ and $\mathcal{F}'$. As $\mathrm{wt}_{\mathcal{F}}(F') = \mathrm{wt}_{\mathcal{F}'}(F') = \mathrm{wt}_{\mathcal{F}}(F)$ and F was correctable by distance under $\mathcal{F}$, F' too must have been correctable by distance under $\mathcal{F}$. As F' has the same weight in $\mathcal{F}'$ and splitting does not decrease the distance,

F' must too be correctable by distance under $\mathcal{F}'$.

Otherwise, if $\mathrm{wt}_{\mathcal{F}'}(F') = \mathrm{wt}_{\mathcal{F}}(F) + 1$, then we can separate F' into a part F'' on D and the X flip on the top wire such that $\mathrm{wt}_{\mathcal{F}'}(F'') = \mathrm{wt}_{\mathcal{F}}(F)$. Therefore F'' is correctable by distance in both noise models. Any solution in $\mathcal{F}'$ to the syndrome of F must involve an X flip on either the top or the bottom internal wire and a solution to $syn(F'')$.

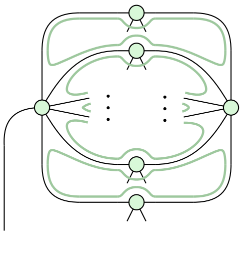
We assumed that F was correctable by distance and as F was nontrivial, $\mathrm{dist}_{\mathcal{F}}(D(D_2)) > 2$ must hold. But then, the X flip on the top and bottom internal wires must be equivalent (cf. Section 3.4). Therefore, F' must be correctable in $\mathcal{F}'$ and as $F \equiv F'$, the decoding reduction from $\mathcal{F}$ to $\mathcal{F}'$ is successful.

We emphasise that the key to success in this proof was that both splittings decomposed a fault with some weight n onto the composition of two faults where one has weight less or equal to n and the other one triggered a detecting region that uniquely identified it. This unique identification allowed us to compensate for the weight increase in the split noise model.

Unsuccessful application of splitting for r_{odd}

In Section 3.5, we have seen that the fault-equivalent rewrites r_{even} and r_{odd} cannot preserve matchability for blossom-shaped Pauli web multisets. We substantiated this claim with the fact that any detector-aware rewrites would need to cover the internal wires of these rewrites with more Pauli webs than allowed by the matchability definition. Here, we demonstrate an attempt to apply splitting to this problem to recover the matchability of the rewritten diagram and Pauli web multiset.

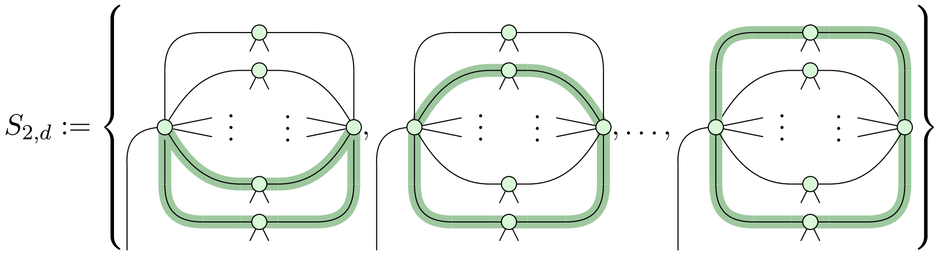
Let us first design the diagram that we want to split. Recall that in Section 3.5, we constructed the following matchable local detector basis for the RHS of r_{odd}:



(A.10)

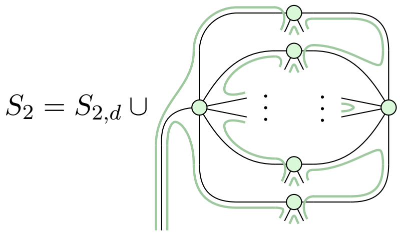
This detector basis already covers all but the upper and lower internal wires maximally, i.e. with two green Pauli webs. Every boundary edge of this diagram needs to be connected with two other boundary edges through a Pauli web. We could route all of these Pauli webs through the left spider but then, at least $n-2$ wires would be covered with more than two Pauli webs.

We can avoid creating many unmatchable wires by routing as many Pauli webs as possible through one specific wire. For example, the following local detector basis contains $n-1$ Pauli webs that each cover the bottom wire:



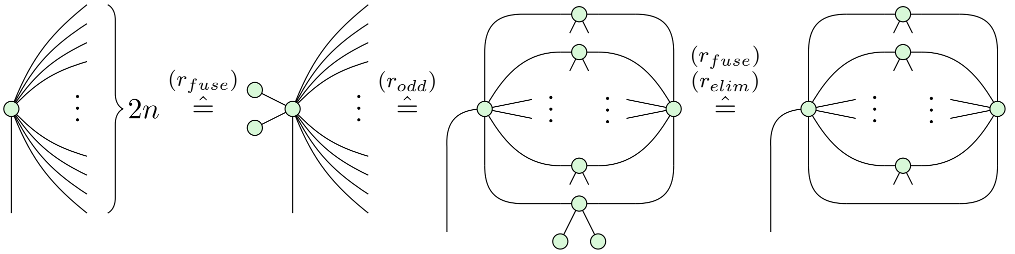
(A.11)

This alternative basis choice is already unmatchable but all but the bottom two internal wires are now only covered with a single Pauli web. This leaves enough room to route all of the boundary-covering Pauli webs through the diagram without making any further wires unmatchable:



(A.12)

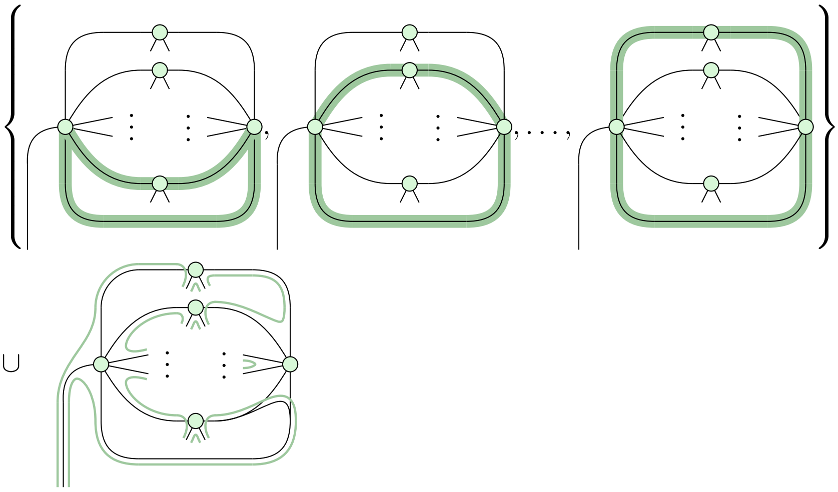
Furthermore, we can unfuse two green spiders from the LHS of the rewrite to create a variant of the r_{odd} rewrite where there is a wire between the left and the right spider in the RHS:



(A.13)

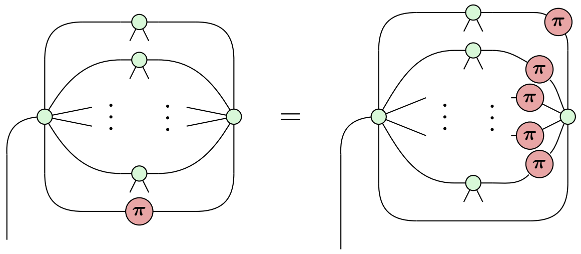
For this rewrite, we can now use the same tricks as in Eqs. (A.11) and (A.12) to create a Pauli web multiset that is matchable on all wires except one:

$S'_2 :=$



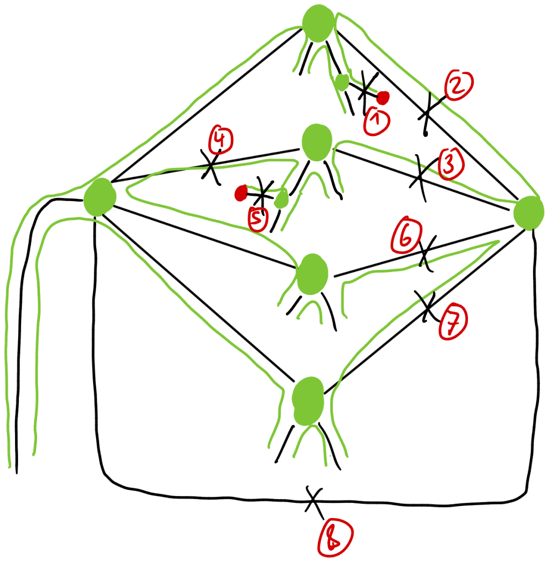
(A.14)

Finally, we can split the X flip on the unmatchable wire by pushing it through the right spider:



(A.15)

However, even though we minimised the number of splittings and onto split onto atomic faults on the internal wires of the RHS, this splitting may corrupt the decoder (in the following diagram, the local detectors are omitted for legibility):



(A.16)

In the above picture, we indicated the X flips $F_1, \ldots, F_8$. If in the original noise model, the fault $F := F_8 F_5 F_1$ with weight 3 occurs, then the correct solution for the split decoder would be $\bar{F} = F_2 F_3 F_6 F_7 F_1 F_5$ with weight 6. However, $\bar{F}' = F_1 F_4 F_2 F_6 F_7$ triggers the same syndrome with weigh 5. Therefore, the optimal decodings of the original and the split decoder do not coincide and splitting is not provably correct in this case.

References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. 9th ed. Cambridge: Cambridge Univ. Press, 2007. ISBN: 978-0-521-63503-5. URL: https://www.cambridge.org/9780521635035.

[2] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[3] Daniel Gottesman. *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation*. 2009. URL: https://arxiv.org/abs/0904.2557.

[4] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint, 2024.

[5] Peter-Jan H. S. Derks et al. *Designing fault-tolerant circuits using detector error models*. 2024. URL: https://arxiv.org/abs/2407.13826.

[6] Min-Hsiu Hsieh and François Le Gall. "NP-hardness of decoding quantum error-correction codes". In: *Physical Review A* 83.5 (May 2011). ISSN: 1094-1622. DOI: 10.1103/physreva.83.052331. URL: http://dx.doi.org/10.1103/PhysRevA.83.052331.

[7] Austin Fowler. "Towards sufficiently fast quantum error correction". Conference Talk. Conference Talk. Sept. 13, 2017. URL: https://www.youtube.com/watch?v=mLnKhSix71A.

[8] Oscar Higgott and Craig Gidney. "Sparse Blossom: correcting a million errors per core second with minimum-weight matching". In: *Quantum* 9 (Jan. 2025), p. 1600. ISSN: 2521-327X. DOI: 10.22331/q-2025-01-20-1600. URL: https://doi.org/10.22331/q-2025-01-20-1600.

[9] Oscar Higgott. *PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching.* 2021. URL: https://arxiv.org/abs/2105.13082.

[10] Nicolas Delfosse. "Decoding color codes by projection onto surface codes". In: *Physical Review A* 89.1 (Jan. 2014). ISSN: 1094-1622. DOI: 10.1103/physreva.89.012317. URL: http://dx.doi.org/10.1103/PhysRevA.89.012317.

[11] Nicolas Delfosse et al. *Splitting decoders for correcting hypergraph faults.* 2023. URL: https://arxiv.org/abs/2309.15354.

[12] Craig Gidney. "Stim: a fast stabilizer circuit simulator". In: *Quantum* 5 (July 2021), p. 497. ISSN: 2521-327X. DOI: 10.22331/q-2021-07-06-497. URL: https://doi.org/10.22331/q-2021-07-06-497.

[13] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Fault Tolerance by Construction.* 2025. URL: https://arxiv.org/abs/2506.17181.

[14] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Floquetifying stabiliser codes with distance-preserving rewrites.* 2024. URL: https://arxiv.org/abs/2410.17240.

[15] Hector Bombin et al. "Unifying flavors of fault tolerance with the ZX calculus". In: *Quantum* 8 (June 2024), p. 1379. ISSN: 2521-327X. DOI: 10.22331/q-2024-06-18-1379. URL: https://doi.org/10.22331/q-2024-06-18-1379.

[16] John van de Wetering. *ZX-calculus for the working quantum computer scientist.* 2020. URL: https://arxiv.org/abs/2012.13966.

[17] Bob Coecke and Ross Duncan. *A graphical calculus for quantum observables.* Oxford, 2007. URL: https://www.cs.ox.ac.uk/people/bob.coecke/GreenRed.pdf (visited on 09/06/2025).

[18] Bob Coecke and Ross Duncan. "Interacting Quantum Observables". In: *Automata, Languages and Programming.* Ed. by Luca Aceto et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 298–310. ISBN: 978-3-540-70583-3.

[19] Aleks Kissinger and John van de Wetering. "Reducing the number of non-Clifford gates in quantum circuits". In: *Phys. Rev. A* 102.2 (Aug. 2020), p. 022406. DOI:

10.1103/PhysRevA.102.022406. URL: https://link.aps.org/doi/10.1103/PhysRevA.102.022406.

[20] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "Completeness of the ZX-Calculus". In: *Logical Methods in Computer Science* Volume 16, Issue 2 (June 2020). ISSN: 1860-5974. DOI: 10.23638/lmcs-16(2:11)2020. URL: http://dx.doi.org/10.23638/LMCS-16(2:11)2020.

[21] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[22] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* 1996. URL: https://arxiv.org/abs/quant-ph/9605043.

[23] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558. DOI: 10.1098/rspa.1992.0167. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1992.0167.

[24] Jay Gambetta. *The hardware and software for the era of quantum utility is here.* IBM Research Blog. Dec. 4, 2023. URL: https://www.ibm.com/quantum/blog/quantum-roadmap-2033 (visited on 08/21/2025).

[25] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (Apr. 2021), p. 433. ISSN: 2521-327X. DOI: 10.22331/q-2021-04-15-433. URL: http://dx.doi.org/10.22331/q-2021-04-15-433.

[26] IBM Quantum. "2025 Development and Innovation Roadmap". 2025. URL: https://www.ibm.com/downloads/documents/us-en/131cf87ab63319bf (visited on 09/09/2025).

[27] Daniel Gottesman. *Surviving as a Quantum Computer in a Classical World.* May 7, 2024. URL: https://www.cs.umd.edu/class/spring2024/cmsc858G/QECCbook-2024-ch1-15.pdf (visited on 07/29/2025).

[28] A.Yu. Kitaev. “Fault-tolerant quantum computation by anyons”. In: *Annals of Physics* 303.1 (Jan. 2003), pp. 2–30. ISSN: 0003-4916. DOI: 10.1016/s0003-4916(02)00018-0. URL: http://dx.doi.org/10.1016/S0003-4916(02)00018-0.

[29] Maximilian Rüsch. “Completeness for Fault Equivalence of Clifford ZX Diagrams”. Master’s thesis. Oxford, UK: University of Oxford, Sept. 3, 2025.

[30] Lance Fortnow. “Fifty years of P vs. NP and the possibility of the impossible”. In: *Commun. ACM* 65.1 (Dec. 2021), pp. 76–85. ISSN: 0001-0782. DOI: 10.1145/3460351. URL: https://doi.org/10.1145/3460351.

[31] Eric Dennis et al. “Topological quantum memory”. In: *Journal of Mathematical Physics* 43.9 (Sept. 2002), pp. 4452–4505. ISSN: 1089-7658. DOI: 10.1063/1.1499754. URL: http://dx.doi.org/10.1063/1.1499754.

[32] Ross Duncan et al. “Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus”. In: *Quantum* 4 (June 2020), p. 279. ISSN: 2521-327X. DOI: 10.22331/q-2020-06-04-279. URL: http://dx.doi.org/10.22331/q-2020-06-04-279.

[33] Jack Edmonds. “Paths, Trees, and Flowers”. In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467. DOI: 10.4153/CJM-1965-045-4.

[34] Yue Wu and Lin Zhong. *Fusion Blossom: Fast MWPM Decoders for QEC.* 2023. URL: https://arxiv.org/abs/2305.08307.

[35] Yue Wu, Namitha Liyanage, and Lin Zhong. “Micro Blossom: Accelerated Minimum-Weight Perfect Matching Decoding for Quantum Error Correction”. In: *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2.* ASPLOS ’25. ACM, Mar. 2025, pp. 639–654. DOI: 10.1145/3676641.3716005. URL: http://dx.doi.org/10.1145/3676641.3716005.

[36] Linnea Grans-Samuelsson et al. "Improved Pairwise Measurement-Based Surface Code". In: *Quantum* 8 (Aug. 2024), p. 1429. ISSN: 2521-327X. DOI: 10.22331/q-2024-08-02-1429. URL: http://dx.doi.org/10.22331/q-2024-08-02-1429.

[37] K. P. Girish and Sunil Jacob John. "Relations and functions in multiset context". In: *Information Sciences* 179.6 (2009), pp. 758–768. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2008.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0020025508004751.

[38] Arthur Pesah. "Fault-tolerant transformations of spacetime codes". Conference Talk. Conference Talk. 2025. URL: https://www.youtube.com/watch?v=FQrL5icKoyI (visited on 07/30/2025).

[39] Coen Borghans. "ZX-Calculus and Quantum Stabilizer Theory". Master's thesis. Nijmegen: Radboud University, 2019. 80 pp. URL: https://www.cs.ox.ac.uk/people/aleks.kissinger/papers/borghans-thesis.pdf (visited on 08/31/2025).

[40] *IBM Quantum Plaform Documentation - QPU Information*. 2025. URL: https://quantum.cloud.ibm.com/docs/en/guides/qpu-information (visited on 09/10/2025).

[41] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. 1998. URL: https://arxiv.org/abs/quant-ph/9807006.

[42] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. 1997. URL: https://arxiv.org/abs/quant-ph/9705052.

[43] Razin Shaikh. "ZXLive - An Interactive GUI for the ZX Calculus". Nov. 20, 2023. URL: https://www.youtube.com/watch?v=J--c2q-KOc8 (visited on 08/16/2025).

[44] Aleks Kissinger and John van de Wetering. "PyZX: Large Scale Automated Diagrammatic Reasoning". In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 2020), pp. 229–241. ISSN: 2075-2180. DOI: 10.4204/eptcs.318.14. URL: http://dx.doi.org/10.4204/EPTCS.318.14.

[45] Nicolas Delfosse and Naomi H. Nickerson. "Almost-linear time decoding algorithm for topological codes". In: *Quantum* 5 (Dec. 2021), p. 595. ISSN: 2521-327X. DOI:

10.22331/q-2021-12-02-595. URL: http://dx.doi.org/10.22331/q-2021-12-02-595.

[46] Pavithran Iyer and David Poulin. *Hardness of decoding quantum stabilizer codes.* 2013. URL: https://arxiv.org/abs/1310.3235.

Bibliography

[1] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information.* 9th ed. Cambridge: Cambridge Univ. Press, 2007. ISBN: 978-0-521-63503-5. URL: https://www.cambridge.org/9780521635035.

[3] Ross Duncan et al. "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus". In: *Quantum* 4 (June 2020), p. 279. ISSN: 2521-327X. DOI: 10.22331/q-2020-06-04-279. URL: http://dx.doi.org/10.22331/q-2020-06-04-279.

[4] Aleks Kissinger and John van de Wetering. "Reducing the number of non-Clifford gates in quantum circuits". In: *Phys. Rev. A* 102.2 (Aug. 2020), p. 022406. DOI: 10.1103/PhysRevA.102.022406. URL: https://link.aps.org/doi/10.1103/PhysRevA.102.022406.

[5] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Floquetifying stabiliser codes with distance-preserving rewrites.* 2024. URL: https://arxiv.org/abs/2410.17240.

[6] Peter-Jan H. S. Derks et al. *Designing fault-tolerant circuits using detector error models.* 2024. URL: https://arxiv.org/abs/2407.13826.

[7] Nicolas Delfosse et al. *Splitting decoders for correcting hypergraph faults.* 2023. URL: https://arxiv.org/abs/2309.15354.

[8] Linnea Grans-Samuelsson et al. "Improved Pairwise Measurement-Based Surface Code". In: *Quantum* 8 (Aug. 2024), p. 1429. ISSN: 2521-327X. DOI: 10.22331/q-2024-08-02-1429. URL: http://dx.doi.org/10.22331/q-2024-08-02-1429.

[9] Michael E. Beverland, Shilin Huang, and Vadym Kliuchnikov. *Fault tolerance of stabilizer channels.* Feb. 25, 2024. DOI: 10.48550/arXiv.2401.12017. arXiv: 2401.12017[quant-ph]. URL: http://arxiv.org/abs/2401.12017 (visited on 04/01/2025).

[10] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation.* Preprint, 2024.

[11] Hector Bombin et al. "Unifying flavors of fault tolerance with the ZX calculus". In: *Quantum* 8 (June 2024), p. 1379. ISSN: 2521-327X. DOI: 10.22331/q-2024-06-18-1379. URL: https://doi.org/10.22331/q-2024-06-18-1379.

[12] John van de Wetering. *ZX-calculus for the working quantum computer scientist.* 2020. URL: https://arxiv.org/abs/2012.13966.

[13] Oscar Higgott and Craig Gidney. "Sparse Blossom: correcting a million errors per core second with minimum-weight matching". In: *Quantum* 9 (Jan. 2025), p. 1600. ISSN: 2521-327X. DOI: 10.22331/q-2025-01-20-1600. URL: https://doi.org/10.22331/q-2025-01-20-1600.

[14] Craig Gidney. "Stim: a fast stabilizer circuit simulator". In: *Quantum* 5 (July 2021), p. 497. ISSN: 2521-327X. DOI: 10.22331/q-2021-07-06-497. URL: https://doi.org/10.22331/q-2021-07-06-497.

[15] Adam Paetznick et al. "Performance of Planar Floquet Codes with Majorana-Based Qubits". In: *PRX Quantum* 4.1 (Jan. 2023), p. 010310. DOI: 10.1103/PRXQuantum.4.010310. URL: https://link.aps.org/doi/10.1103/PRXQuantum.4.010310.

[16] Christopher A. Pattison et al. *Improved quantum error correction using soft information.* 2021. URL: https://arxiv.org/abs/2107.13589.

[17] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction.* 1997. URL: https://arxiv.org/abs/quant-ph/9705052.

[18] Daniel Gottesman. *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation.* 2009. URL: https://arxiv.org/abs/0904.2557.

[19] Nicolas Delfosse. "Decoding color codes by projection onto surface codes". In: *Physical Review A* 89.1 (Jan. 2014). ISSN: 1094-1622. DOI: 10.1103/physreva.89.012317. URL: http://dx.doi.org/10.1103/PhysRevA.89.012317.

[20] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Fault Tolerance by Construction.* 2025. URL: https://arxiv.org/abs/2506.17181.

[21] Rui Chao and Ben W. Reichardt. "Flag Fault-Tolerant Error Correction for any Stabilizer Code". In: *PRX Quantum* 1.1 (Sept. 2020). ISSN: 2691-3399. DOI: 10.1103/prxquantum.1.010302. URL: http://dx.doi.org/10.1103/PRXQuantum.1.010302.

[22] Daniel Litinski. *Blocklet concatenation: Low-overhead fault-tolerant protocols for fusion-based quantum computation.* 2025. URL: https://arxiv.org/abs/2506.13619.

[23] Quantinuum. *Making fault-tolerance a reality: Introducing our QEC decoder toolkit.* Nov. 14, 2024. URL: https://www.quantinuum.com/blog/making-fault-tolerance-a-reality-introducing-our-qec-decoder-toolkit (visited on 06/30/2025).

[24] K. P. Girish and Sunil Jacob John. "Relations and functions in multiset context". In: *Information Sciences* 179.6 (2009), pp. 758–768. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2008.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0020025508004751.

[25] Bob Coecke and Ross Duncan. "Interacting Quantum Observables". In: *Automata, Languages and Programming.* Ed. by Luca Aceto et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 298–310. ISBN: 978-3-540-70583-3.

[26] Daniel Gottesman. *Surviving as a Quantum Computer in a Classical World.* May 7, 2024. URL: https://www.cs.umd.edu/class/spring2024/cmsc858G/QECCbook-2024-ch1-15.pdf (visited on 07/29/2025).

[27] Arthur Pesah. "Fault-tolerant transformations of spacetime codes". Conference Talk. Conference Talk. 2025. URL: https://www.youtube.com/watch?v=FQrL5icKoyI (visited on 07/30/2025).

[28] Alex Townsend-Teague. “Pauli Webs”. Kyoto, Apr. 14, 2025. URL: https://www.youtube.com/watch?v=-mpc-7OzUfU&list=PLudqKsS7UmpSLLQIiFH8KsVOpNtm-5lY5&index=26 (visited on 07/30/2025).

[29] Aleks Kissinger. *Phase-free ZX diagrams are CSS codes (...or how to graphically grok the surface code).* 2022. URL: https://arxiv.org/abs/2204.14038.

[30] Nicolas Delfosse and Adam Paetznick. *Spacetime codes of Clifford circuits.* 2023. URL: https://arxiv.org/abs/2304.05943.

[31] Daniel Gottesman. *Opportunities and Challenges in Fault-Tolerant Quantum Computation.* 2022. URL: https://arxiv.org/abs/2210.15844.

[32] Matt McEwen, Dave Bacon, and Craig Gidney. “Relaxing Hardware Requirements for Surface Code Circuits using Time-dynamics”. In: *Quantum* 7 (Nov. 2023), p. 1172. ISSN: 2521-327X. DOI: 10.22331/q-2023-11-07-1172. URL: http://dx.doi.org/10.22331/q-2023-11-07-1172.

[33] Eric Dennis et al. “Topological quantum memory”. In: *Journal of Mathematical Physics* 43.9 (Sept. 2002), pp. 4452–4505. ISSN: 1089-7658. DOI: 10.1063/1.1499754. URL: http://dx.doi.org/10.1063/1.1499754.

[34] Aleks Kissinger and John van de Wetering. “PyZX: Large Scale Automated Diagrammatic Reasoning”. In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 2020), pp. 229–241. ISSN: 2075-2180. DOI: 10.4204/eptcs.318.14. URL: http://dx.doi.org/10.4204/EPTCS.318.14.

[35] Aleks Kissinger and Vladimir Zamdzhiev. “Quantomatic: A Proof Assistant for Diagrammatic Reasoning”. In: *Automated Deduction - CADE-25.* Springer International Publishing, 2015, pp. 326–336. ISBN: 978-3-319-21401-6. DOI: 10.1007/978-3-319-21401-6_22. URL: http://dx.doi.org/10.1007/978-3-319-21401-6_22.

[36] Razin Shaikh. “ZXLive - An Interactive GUI for the ZX Calculus”. Nov. 20, 2023. URL: https://www.youtube.com/watch?v=J--c2q-KOc8 (visited on 08/16/2025).

[37] Min-Hsiu Hsieh and François Le Gall. “NP-hardness of decoding quantum error-correction codes”. In: *Physical Review A* 83.5 (May 2011). ISSN: 1094-1622. DOI:

10.1103/physreva.83.052331. URL: http://dx.doi.org/10.1103/PhysRevA.83.052331.

[38] Oscar Higgott. *PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching.* 2021. URL: https://arxiv.org/abs/2105.13082.

[39] Yue Wu and Lin Zhong. *Fusion Blossom: Fast MWPM Decoders for QEC.* 2023. URL: https://arxiv.org/abs/2305.08307.

[40] Lance Fortnow. "Fifty years of P vs. NP and the possibility of the impossible". In: *Commun. ACM* 65.1 (Dec. 2021), pp. 76–85. ISSN: 0001-0782. DOI: 10.1145/3460351. URL: https://doi.org/10.1145/3460351.

[41] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[42] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* 1996. URL: https://arxiv.org/abs/quant-ph/9605043.

[43] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558. DOI: 10.1098/rspa.1992.0167. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1992.0167.

[44] Jay Gambetta. *The hardware and software for the era of quantum utility is here.* IBM Research Blog. Dec. 4, 2023. URL: https://www.ibm.com/quantum/blog/quantum-roadmap-2033 (visited on 08/21/2025).

[45] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (Apr. 2021), p. 433. ISSN: 2521-327X. DOI: 10.22331/q-2021-04-15-433. URL: http://dx.doi.org/10.22331/q-2021-04-15-433.

[46] A.Yu. Kitaev. “Fault-tolerant quantum computation by anyons”. In: *Annals of Physics* 303.1 (Jan. 2003), pp. 2–30. ISSN: 0003-4916. DOI: 10.1016/s0003-4916(02)00018-0. URL: http://dx.doi.org/10.1016/S0003-4916(02)00018-0.

[47] P.W. Shor. “Fault-tolerant quantum computation”. In: *Proceedings of 37th Conference on Foundations of Computer Science*. 1996, pp. 56–65. DOI: 10.1109/SFCS.1996.548464.

[48] Yue Wu, Namitha Liyanage, and Lin Zhong. “Micro Blossom: Accelerated Minimum-Weight Perfect Matching Decoding for Quantum Error Correction”. In: *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS ’25. ACM, Mar. 2025, pp. 639–654. DOI: 10.1145/3676641.3716005. URL: http://dx.doi.org/10.1145/3676641.3716005.

[49] Pavithran Iyer and David Poulin. *Hardness of decoding quantum stabilizer codes*. 2013. URL: https://arxiv.org/abs/1310.3235.

[50] Coen Borghans. “ZX-Calculus and Quantum Stabilizer Theory”. Master’s thesis. Nijmegen: Radboud University, 2019. 80 pp. URL: https://www.cs.ox.ac.uk/people/aleks.kissinger/papers/borghans-thesis.pdf (visited on 08/31/2025).

[51] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. 1998. URL: https://arxiv.org/abs/quant-ph/9807006.

[52] Maximilian Rüsch. “Completeness for Fault Equivalence of Clifford ZX Diagrams”. Master’s thesis. Oxford, UK: University of Oxford, Sept. 3, 2025.

[53] Austin Fowler. “Towards sufficiently fast quantum error correction”. Conference Talk. Conference Talk. Sept. 13, 2017. URL: https://www.youtube.com/watch?v=mLnKhSix71A.

[54] Nicolas Delfosse and Naomi H. Nickerson. “Almost-linear time decoding algorithm for topological codes”. In: *Quantum* 5 (Dec. 2021), p. 595. ISSN: 2521-327X. DOI: 10.22331/q-2021-12-02-595. URL: http://dx.doi.org/10.22331/q-2021-12-02-595.

[55] Niel de Beaudrap, Aleks Kissinger, and John van de Wetering. "Circuit Extraction for ZX-Diagrams Can Be #P-Hard". In: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. DOI: 10.4230/LIPICS.ICALP.2022.119. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2022.119.

[56] Bob Coecke and Ross Duncan. *A graphical calculus for quantum observables.* Oxford, 2007. URL: https://www.cs.ox.ac.uk/people/bob.coecke/GreenRed.pdf (visited on 09/06/2025).

[57] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "Completeness of the ZX-Calculus". In: *Logical Methods in Computer Science* Volume 16, Issue 2 (June 2020). ISSN: 1860-5974. DOI: 10.23638/lmcs-16(2:11)2020. URL: http://dx.doi.org/10.23638/LMCS-16(2:11)2020.

[58] IBM Quantum. "2025 Development and Innovation Roadmap". 2025. URL: https://www.ibm.com/downloads/documents/us-en/131cf87ab63319bf (visited on 09/09/2025).

[59] Jack Edmonds. "Paths, Trees, and Flowers". In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467. DOI: 10.4153/CJM-1965-045-4.

[60] *IBM Quantum Plaform Documentation - QPU Information.* 2025. URL: https://quantum.cloud.ibm.com/docs/en/guides/qpu-information (visited on 09/10/2025).