Completeness for Fault Equivalence of Clifford ZX Diagrams

Maximilian Rüsch

University of Oxford

Submitted in partial completion of the

Masters of Science in Advanced Computer Science

Trinity 2025

For truth.

Abstract

Recently, the novel notion of fault equivalence was introduced to reason about the relationship of noise models on different diagrams. Two diagrams are considered fault-equivalent if every undetectable fault on one diagram has a direct correspondence on the other, both in terms of fault semantics and likelihood. Together with the ZX calculus, a diagrammatic language that is used to reason about quantum computations, fault-equivalent rewrites enable optimisation of a computation without changing its behaviour under noise.

We solidify the framework of fault equivalence by adapting the ZX calculus and providing an axiomatisation that is provably sound and complete for showing fault equivalence on Clifford ZX diagrams. We present useful intermediate notions and tools that extend the formal framework of fault equivalence and reasoning about noise on ZX diagrams in general, and show how they find applications beyond fault equivalence. Furthermore, we provide an implementation that is directly derived from the completeness result as an automated method for showing fault equivalence between diagrams. Through this work, we facilitate fully automatic verification of fault-tolerant implementations of quantum computation specifications under a flexible and formally rigorous notion of noise.

Acknowledgements

I take this moment to thank both of my supervisors Benjamin Rodatz and Aleks Kissinger. Your impact reaches from immensely useful discussions and feedback during all stages of this work to enabling further opportunities for the development of myself, this project, and beyond. I remain extraordinarily grateful to you for undertaking this journey with me.

Additionally, I want to thank Boldizsár Poór and Maximilian Schweikart for supporting this work through insightful supplementary discussions that enabled great improvements to the quality of this work. I want to thank Șerban Cercelescu for providing considerations and insights and perhaps inadvertently providing a particularly fruitful idea for proving Proposition 2.33.

I want to thank all those that claimed temporary residence on the fifth floor of the Computer Science department for providing a warm and welcoming environment that greatly facilitated the writing process.

Finally, I thank Becca for your ongoing support throughout all the time spent on this work, for all special moments gifted to encourage me, and for the bravery required to join me in my journey.

Contents

1 Introduction

Scaling quantum computation involves, among many other components, optimising quantum programs and suppressing noise. The former is a part of the field of quantum compilation, while the latter is concerned with in the field of fault-tolerant quantum computing (FTQC). Independently, both fields are making progress: Entire frameworks, libraries and calculi are built to both systematically and heuristically simplify quantum programs while preserving their semantics [KvdW20; vdWet20], while FTQC research offers a versatile range of tools to reason about, detect, and correct faults resulting from noise [Got09; Got97]. This work aims to extend the range of formal tools available and improve the formalisation of existing notions.

An emerging framework for circuit synthesis and optimisation is the ZX calculus [CD11]. In the ZX calculus, quantum programs are thought of as diagrams obtained from a small set of simple generators. These diagrams are then rewritten using a small number of rules which provide alternative diagrams with the same semantics. The calculus inherently comes equipped with a set of rewrites that is both sound and complete for this semantic equivalence.

A key feature of the calculus is that ZX diagrams neither possess nor require a notion of time: A single ZX diagram can have multiple interpretations depending on the direction of time and positioning qubit world lines. This allows the ZX calculus to compress the space of quantum programs considerably, mapping multiple equivalent circuits onto the same ZX diagram which may then be considered as a simple undirected graph. Through ZX rewrites, diagrams may pass through multiple intermediate stages with different interpretations, where for some it might not be directly apparent how to implement them physically. The problem of obtaining a circuit that fits a given interpretation of a ZX diagram is known in the community as 'circuit extraction' [KvdW24], which was shown to be computationally hard in the general case [BKW22].

The ZX calculus has also been applied to reason about noise and thus joined the FTQC toolbox [Hua+23b; TFK23]. In particular, restricting generators to obtain the well known Clifford fragment of the ZX calculus enables reasoning about faults in easily classically simulable ZX diagrams [AG04]. Furthermore, when modeling faults through stabiliser theory, the Clifford fragment suffices to reason about faults in its own diagrams, since it is proven to be complete for stabiliser quantum mechanics [Bac14].

Even though arbitrary rewrites for ZX diagrams can be obtained from a handful of rules, some of these rules change a program's behaviour under noise. When constructing a fault-tolerant computation, inattentive application of rewrites might lead to a decrease in fault tolerance. Interestingly, not all rewrites derived from 'bad' rules are affected: Some rewrites may fully preserve fault tolerance, others preserve it up to a certain degree still sufficient for the purposes of the user. These rewrites are classified as 'fault-equivalent' [RPK24; RPK25]. Two diagrams are fault-equivalent if all undetectable faults in one diagram have a correspondence in the other diagram, i.e. when rewriting one diagram into the other no new and potentially detrimental faults are introduced. Fault equivalence is compositional, so newly discovered fault-equivalent rewrites may open up a wide range of previously unknown fault-tolerant implementations of quantum programs.

Checking whether a certain rewrite is fault-equivalent is either tedious or requires additional knowledge about the rewrite or the context in which it occurs to perform succinctly [RPK25]. For one, fault equivalence as introduced later on consists of finding semantically equivalent faults for the diagram before and after the rewrite, constrained by a discrete measure of likelihood, commonly known as the 'weight' of a fault [Got97]. Determining the semantics of a singular fault may involve calculating the full linear map for general ZX diagrams, which quickly becomes intractable for both manual and algorithmic processing. Moreover, the number of faults that need to be checked for a given rewrite usually grows exponentially with the size of the diagram. In fact, deciding whether a general ZX rewrite is fault-equivalent is NP-hard [RPK25]. Only for particularly small rewrites, deductive proofs of fault equivalence stay meaningful without using additional knowledge about the structure of the rewrite. Structural knowledge can shrink these proofs, but since as this makes the proof dependent on the details of the rewrite, even minimal variations of rewrites may require an entirely different proof (see the proofs in [RPK25, Prop. 6.15, 6.16] for an example). Although naïve automated procedures for this decision problem may be directly derived from its definition, they scale exponentially in the number of internal edges of the diagram, rendering them unusable for larger diagrams. Thus, practically useful and scalable methods and connected definitions still remain to be discovered and implemented.

This work provides a novel extension to the ZX calculus to facilitate modeling faults drawn from some noise model directly inside diagrams. Through this extension, reasoning about weighted faults in the Clifford fragment of the ZX calculus becomes easier and more natural. In particular, the extension enables specifying a finite set of fault-equivalent rewrites that are sound and complete for fault equivalence. That is, the calculus does not allow deriving fault equivalences that do not actually hold, and it allows deriving every relationship that does hold. Thus, proofs of fault equivalence between two diagrams can be provided purely in this extended ZX calculus. As the proof of completeness is constructive, an algorithmic procedure for checking fault equivalence can be directly derived. The new algorithm opens venues to many subsequent optimisations by restricting exponential scaling and making it independent of the number of edges internal to the ZX diagram, offering a clear improvement versus the naïve procedure.

As part of the derivation, the noise model is encoded into the diagram, whereafter they are incrementally brought into a novel normal form that separates the diagram's (fault-free) semantics and changes to these semantics that might occur due to faults from the noise model. This form and its precursors are highly versatile: Besides allowing to prove completeness, they may be used to recover e.g. the notion of a diagrams distance [Got97; RPK25]. In particular, it enables us to decouple the analysis of such properties entirely from the fact that the program under analysis is given by a ZX diagram, even if the physical interpretation is not entirely clear. One property of high interest is the *logical error rate* of a circuit, and the mainstream tools developed to efficiently empirically obtain it have not yet been lifted to the ZX calculus [Gid21b; Tuc20; Hua+23a]. Our decoupling largely prepares such a novel lifting. In particular, once a description of a diagrams noise is obtained and brought into normal form, the now diagram-independent result may be fed directly into the simulators. Although the connection to the logical error rate is not formalised in this work, we provide additional indications for completing the proposed lifting.

Finally, an implementation of the extension to the ZX calculus is provided as a ready-to-use Python package [Rüs25] for specific kinds of noise models. The aforementioned algorithm is optimised in ways that are not straightforward in the native ZX calculus to make it suitable for analysing larger ZX diagrams. In particular, the normal form is significantly easier and faster to obtain when using more powerful computation methods outside the ZX calculus.

This work starts out by introducing the necessary background to reasoning about ZX diagrams as well as faults on ZX diagrams. The remainder is organised in two parts.

In the first part, ranging from Sections 3 to 6, the complete rule set is motivated and provided, along with formulating the completeness result itself. We will construct the intermediate notions, tools and algorithms required to prove this result, which will be the final act of this part.

The second part, consisting of Sections 7 and 8, describes the implementation of the Python package and highlights some optimisations made, mainly for obtaining the novel normal forms and subsequent fault equivalence checks. Further, it describes examples of additional properties of noise derivable from the normal form and outlines their potential implementation. Finally, the completeness result may potentially be lifted to more general notions of noise or even more general notions of diagrams than those we consider. The challenges and considerations attached to such a lifting are outlined, including a discussion of ZX calculi for systems with more than two levels (known as 'qudits') and continuous measures of likelihood for faults.

1.1 Related Work

Our work builds on the existing notion of fault equivalence, first derived in [RPK24] and further refined and formalised in [RPK25]. While this existing work is mainly concerned with deriving a systematic end-to-end compilation procedure for fault-tolerant implementations of circuits using fault-equivalent rewrites, we further facilitate the procedure by fully characterising the sufficient and necessary conditions under which a rewrite is fault-equivalent terms that are easier to verify. Additionally, we extend their notion of fault equivalence with a non-symmetric variant, and show that this variant fulfills most of the properties that fault equivalence has.

During the development of our main result, i.e. finding a provably complete calculus for fault equivalence, we encounter the problem of determining the effect of a fault on certain locations including the outputs of the ZX diagram. For Clifford circuits, this can be implemented with propagation of Pauli faults through the circuit, which is a fundamental idea for the development of the stabiliser formalism and fault-tolerant computation in general [Got97]. Traditionally, fault tolerance is achieved by limiting the spread of faults throughout a circuit (which is determined via the aforementioned propagation) and dealing with the remaining faults via simple error correcting codes. However, in particular in [Got22], it was argued that achieving proper fault-tolerant computation requires a shift in focus: From keeping fault propagation under control towards identifying where a specific fault occurred. This knowledge is subsequently sufficient to correct the fault, regardless of how it spread through the circuit.

The formalism attached to this new paradigm is provided in [DP23]. However, we derive a new formalism of 'signatures' and 'effects' that implicitly generalises this paradigm beyond circuits and facilitates reasoning about noise on diagrams without a notion of time. To achieve this, we leverage and adapt the ZX calculus and implement a diagrams noise model directly in the diagram itself, using a construction known as *fault gadgets*, introduced in [RPK25]. Fault gadgets were initially introduced as a static tracker of potential faults; however, we adapt their handling to make them dynamic and able to move throughout a diagram.

Furthermore, we leverage the notion of Pauli webs [Bom+24], which were introduced as a graphical notation on ZX diagrams to visualise stabilisers and detectors. Beyond this purpose, we will use them to guide the movement of fault gadgets through a diagram, in particular for interpolation between two fault gadgets. Using a technique presented in [Bor19], we can obtain independent generators for these webs, which will prove useful both for the theoretic results and the implementation of this work.

Finally, as part of our work on the implementation, we derive additional properties of noise models from our newly introduced framework of signatures and effects. This includes the notion of the logical error rate, which is usually explored empirically using highly efficient stabiliser circuit simulation techniques. While the foundations of such simulations were introduced in [Got97; AG04], we recognise that one of the most widely used simulation packages for this purpose is the `stim` package, presented in [Gid21b]. This package uses a two-step approach, first analysing a circuit to compile a fast sampler from its noise model, and second providing highly optimised functionality to obtain samples that scales well with available parallel compute resources. While we do not fully integrate with the `stim` package, we argue that our approach poses an alternative to the first step, generalising the sampling pipeline from being applicable to just quantum circuits to being applicable to all Clifford ZX diagrams.

2 Preliminaries

This section aims to provide a reasonable foundation for understanding the remainder of this work. However, due to size constraints, we will assume the reader has gone through a basic introduction into quantum computing and is familiar with the Dirac notation, unitary operators, and the correspondence between linear maps and states. A particularly in-depth discussion of such matter may be found in [NC10], a more light-weight introduction in [Pre15].

Sections 2.1 and 2.2 essentially constitute revisions of existing literature to establish a basic foundation. However, in Section 2.3, in particular in Sections 2.4 and 2.5, as well as in Section 2.6, we either refresh existing formalisms or provide additional but novel notions that extend these formalisms. These notions are essential for the remainder of this work, and as such we encourage engaging with these sections in their entirety.

2.1 Stabiliser Formalism

Throughout the FTQC landscape, one of the most useful notions is those of stabiliser circuits and stabiliser states, which are efficiently classically simulable (shown in [Got98, Thm. 1] and later refined in [AG04]). These codes restrict the unitaries that circuits implement to the well-known Clifford group, and are defined through sets of independent operators. This section serves as a brief summary of stabiliser theory that suffices for our purposes; for interested readers we refer to [KvdW24; Got97; Pre15].

At the core of stabiliser theory in the Clifford group are the *Pauli operators*:

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

When viewing a pure state $|\psi\rangle$ as a point on the well-known three-dimensional 'Bloch sphere', these operators correspond directly to π-radian rotations around the X, Y, Z axis respectively. The identity operator I is often included as the fourth Pauli operator, corresponding to no rotation. We have that $X^2 = Y^2 = Z^2 = I$, but there are other easy identities as well, like $-XZ = ZX = iY$.

These operators then compose into a group that generalises to many-qubit states:

Definition 2.1. $\{X, Y, Z\}$ generate the *Pauli group* $\mathcal{P}^1$ under composition. $\mathcal{P}^1$ contains sixteen distinct operators:

$$\mathcal{P}^1 = \{\alpha P \mid \alpha \in \{1, -1, i, -i\}, P \in \{I, X, Y, Z\}\}$$

The n-fold tensor product of $\mathcal{P}^1$ is the n-qubit Pauli group $\mathcal{P}^n$. We can collect all phases through linearity, so $\mathcal{P}^n$ contains 4^{n+1} distinct operators. Two operators P, Q from $\mathcal{P}^n$ always either commute or anticommute, so $PQ = \pm QP$, and they commute iff an even number of their 1-qubit components anticommute.

Ignoring the scalar, which corresponds to factoring $\mathcal{P}^n$ by its centre $\{\pm I, \pm iI\}$, we get the group $\overline{\mathcal{P}^n}$ of the order 4^n. As we can identify Y with the product XZ up to a scalar, we can express an operator $P \in \overline{\mathcal{P}^n}$ as a vector in $\mathbb{B}^{2n}$ [Bor19, Prop. 2.5.4], encoding X and Z operators per component P_i. This enables speaking of linear independence, vector spaces spanned by Pauli operators, and checking commutativity of two operators purely in through computations in their vector form.

We move on to stabilisers:

Definition 2.2. Let $|\psi\rangle$ be a pure state and U a unitary operator. We say that U *stabilises* $|\psi\rangle$, or that U *is a stabiliser of* $|\psi\rangle$ when

$$U|\psi\rangle = |\psi\rangle\,,$$

so $|\psi\rangle$ must be a $+1$-eigenvector of U.

Definition 2.3. An abelian subgroup of $\mathcal{P}$ that does not contain $-I$ is called a *stabiliser group* $\mathcal{S}$.

We identify a stabiliser group $\mathcal{S}$ with a set of independent generators, where independence of an element is the fact that it cannot be expressed through composition of other elements of the set. We note that these generators cannot be any i-phase Pauli $\pm iP \in \mathcal{P}^n$, as then $(\pm iP)^2 = (\pm i)^2 I = -I$. This group $\mathcal{S}$ then identifies an entire quantum state space:

Definition 2.4. The stabiliser group $\mathcal{S}$ spans a *stabiliser space*, containing all vectors that are stabilised by all $S \in \mathcal{S}$.

If $\mathcal{S} \subset \mathcal{P}^n$ is generated by independent generators $S_1, \ldots, S_m$, its stabiliser space has dimension 2^{n-m} [Got98]. This means if we find n independent generators, the group $\mathcal{S}$ generated by them fixes a space containing *exactly one* vector, referred to as the *stabiliser state* of $\mathcal{S}$.

The range of quantum mechanics considered in stabiliser theory is limited to only using these stabiliser states. We thus have to limit the unitaries used to update those states to a group that preserves stabiliser states. This group is usually referred to as the 'Clifford group'. Applying a unitary U to a stabiliser state spanned by generators $S_1, \ldots, S_n$ is the same as applying $US_iU^\dagger$ for all i. So we can identify the n-qubit Clifford group $\mathcal{C}^n$ as those unitaries with

$$\mathcal{C}^n = \{C \mid \forall P \in \mathcal{P}^n : CPC^\dagger \in \mathcal{P}^n\}\,.$$

Applying the Choi-Jamiołkowski isomorphism [NC10], we find that we can identify some n-qubit Clifford unitary exactly with a $2n$-qubit stabiliser state. We will use this correspondence so often that we will stop speaking of Clifford unitaries all together and simply refer to these unitaries as stabiliser states.

2.2 ZX Calculus

The ZX calculus [CD11] is a graphical formalism that can represent any complex $2^m \times 2^n$ matrix for arbitrary m, n. In particular, the full ZX calculus can represent arbitrary quantum operations [KvdW24, Thm. 3.1.4] and it includes all rules required to transform them under semantic preservation. However, we exclusively focus on the Clifford fragment of the ZX calculus, which is universal, sound and complete for stabiliser quantum mechanics [Bac14]. In the remainder of this work, any mention of a ZX diagram implicitly refers to a Clifford ZX diagram. For a comprehensive overview of the full calculus, we refer to [vdWet20], and for a deep dive into the inner workings of the calculus, related topics and other calculi, to [KvdW24].

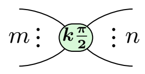
2.2.1 ZX Diagrams

A ZX diagram consists of *spiders* that are equipped with:

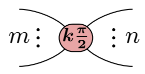
- a type from $\{X, Z\}$ (which is why the calculus is called the 'ZX' calculus),
- a phase $\alpha = k\frac{\pi}{2} \pmod{2\pi}$,
- a number of *input* and *output* legs.

These spiders represent linear maps from $(\mathbb{C}^2)^{\otimes m}$ to $(\mathbb{C}^2)^{\otimes n}$:

Definition 2.5 (Z spider, X spider)**.**

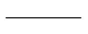
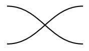
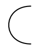
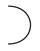
Z-spider: 

$:= |0\rangle^{\otimes n}\langle 0|^{\otimes m} + e^{ik\frac{\pi}{2}}|1\rangle^{\otimes n}\langle 1|^{\otimes m}$

X-spider: 

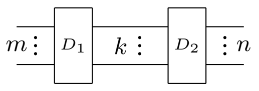
$:= |+\rangle^{\otimes n}\langle +|^{\otimes m} + e^{ik\frac{\pi}{2}}|-\rangle^{\otimes n}\langle -|^{\otimes m}$

Convention allows us to omit the phase parameter of a spider when it is 0. We also include *identities*, *swaps*, *cups*, and *caps* in ZX diagrams:

 $:= \sum_i |i\rangle\langle i|$  $:= \sum_{ij} |ij\rangle\langle ji|$  $:= \sum_i |ii\rangle$  $:= \sum_i \langle ii|$

We can compose spiders to larger networks and express arbitrary linear maps:

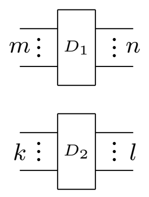
Definition 2.6 (Composition of ZX diagrams)**.** The *sequential composition* of two diagrams $D_1 : (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes k}$ and $D_2 : (\mathbb{C}^2)^{\otimes k} \to (\mathbb{C}^2)^{\otimes n}$ is the diagram $D_2 \circ D_1$, which corresponds graphically to:

$D_2 \circ D_1 \quad \rightsquigarrow$ 

$: (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes n}$

The *parallel composition* of two diagrams $D_1 : (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes n}$ and $D_2 : (\mathbb{C}^2)^{\otimes k} \to (\mathbb{C}^2)^{\otimes l}$ is the diagram $D_1 \otimes D_2$, which corresponds graphically to:

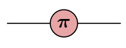
$D_1 \otimes D_2 \quad \rightsquigarrow$



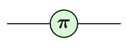
$: (\mathbb{C}^2)^{\otimes(m+k} \to (\mathbb{C}^2)^{\otimes n+l)}$

We will say that two ZX diagrams D_1, D_2 are *semantically equivalent*, written $D_1 = D_2$, if they evaluate to the same linear map up to a global scalar.

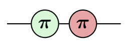
To provide some examples for ZX diagrams, let us revisit the Pauli rotations X, Y, Z from Section 2.1. We can represent the rotations X, Z via spiders, as they directly correspond to rotations about that axis of π radians, and Y as their composition. This is easily checked by computing their linear map:



$$= \quad |+\rangle\langle+| - |-\rangle\langle-| = |1\rangle\langle 0| + |0\rangle\langle 1| \quad = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



$$= \quad |0\rangle\langle 0| - |1\rangle\langle 1| \quad = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

i 

$$= \quad i \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

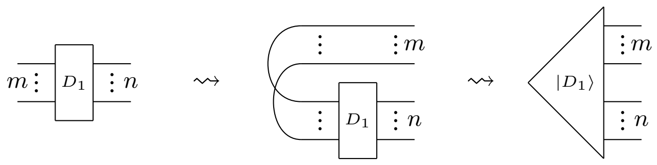
Remark 2.7. In this work, we will generally consider ZX diagrams *up to a global non-zero phase*, so we may omit the scalar i in the definition of the Y rotation above. This matches how we treated stabilisers in Section 2.1 and how we will treat faults in Section 2.3. However, we will encounter two scenarios where the global phase does matter:

- Stabilisers and related constructions (like Pauli webs in Section 2.6) are phase sensitive, i.e. their application requires scalar accuracy. If we disregarded scalars here, the calculus would become unsound for stabiliser theory. In particular, we use this scalar accuracy to our advantage in Proposition 2.29.

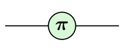
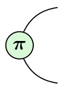
- One can show equality between any two linear maps by showing that they act the same on every element of an orthonormal basis [NC10]. This can also be applied in the ZX calculus. However, the rewrites used to show this equality must be scalar accurate, since one could otherwise introduce relative instead of global phases and become unsound. We will apply this technique in Proposition 6.3.

Note that there may be additional cases where scalar accuracy is required, such as computing probabilities of measurement outcomes [vdWet20], which are not of interest in this work.

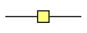
We identify special kinds of Zx diagrams, namely those corresponding to *states* (with no inputs, i.e. column vectors / $|\cdot\rangle$), those corresponding to *measurements* (with no outputs, i.e. row vectors / $\langle\cdot|$), and those corresponding to *scalars* (neither inputs nor outputs). Similar to how we consider Clifford unitaries as stabiliser states through the Choi-Jamiołkowski isomorphism, we can bend around input legs of a diagram into output legs, yielding its corresponding state:



For example, the Z-Pauli map corresponds to the well known (unnormalised) Bell state $|\mathbf{\Phi}\rangle^{-}$:

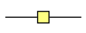
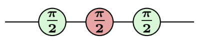
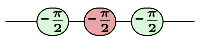
$|0\rangle\langle 0| - |1\rangle\langle 1| \quad =$  $\rightsquigarrow$  $= \quad |00\rangle - |11\rangle$

We further introduce a special notation for the Hadamard gate H, one of the basic Clifford unitaries:



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

This gate may also be expressed diagrammatically in multiple ways using its Euler decomposition into three rotational angles:

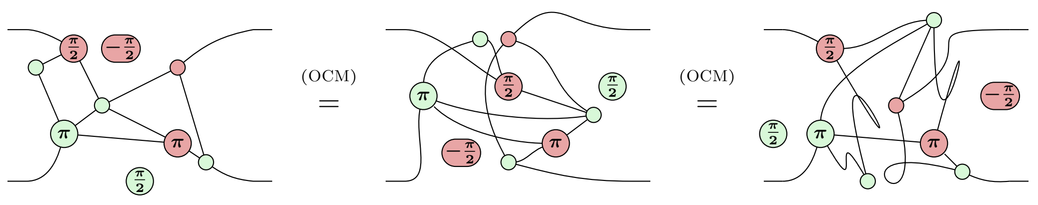
 $=$  $=$  (2.1)

To verify that the two latter diagrams are equal, we could calculate their linear map and check them for equality. However, this is in many cases more cumbersome than required. We can prove every equality that is derivable by using linear algebra in the ZX calculus itself, using a set of rules which we turn to now.

2.2.2 Rewrite Rules

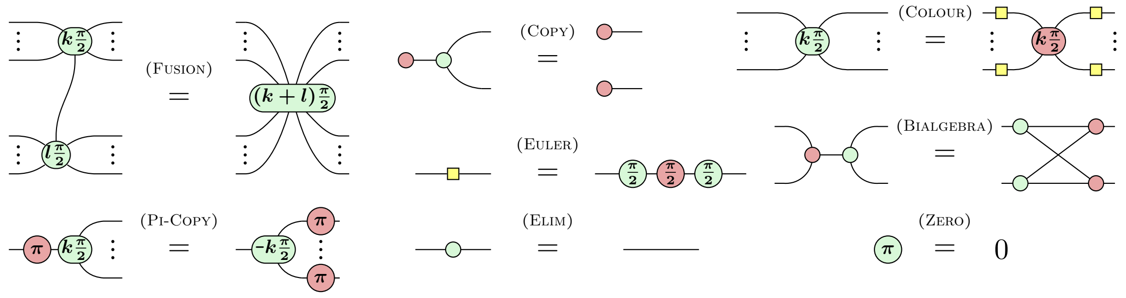
Along with diagrams, the ZX calculus also features a set of axioms called 'rewrite rules', that enable showing equivalences between diagrams. There are different axiomatisations (for example, the axiomatisation in [Bac14] was later refined in [BPW17]), so we use rather *a* ZX calculus than *the* ZX calculus. We settle on a standard set of axioms for ZX diagrams with Clifford phases that is known to be complete for showing semantic equivalence in that fragment [Bac14; BPW17].

Perhaps the most important rule, that we most often assume implicitly, is 'Only Connectivity Matters', commonly abbreviated 'OCM'. This rule states that as long as we keep the connectivity of the spiders internally and the order of inputs and outputs the same, we may move spiders and bend legs arbitrarily without changing the underlying linear map:



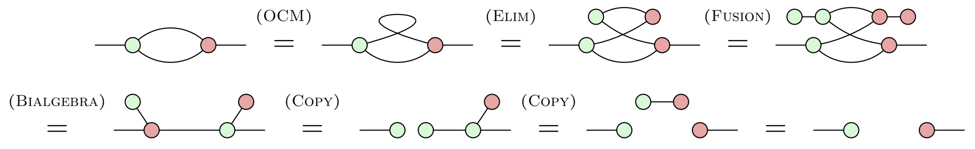
We are thus allowed to view a ZX diagram D as an undirected graph (G, type, α), where type and α assign vertices in G a type in $\{X, Z, B\}$ and an angle respectively. The special type B identifies vertices that are either an input or an output of the diagram. Since we apply the Choi-Jamiołkowski isomorphism and bend all maps into states, we do not distinguish between inputs and outputs and simply refer to them as boundary vertices, and to the edges connected to them as *boundary edges*. All edges that are not boundary edges will be called *internal edges*.

Beyond OCM we present the additional eight rewrite rules that make up the axiomatisation that we use[1]:



All of these rules also hold in the colour inverse, but not all of them are scalar accurate as presented, since we disregard global phases in diagrams most of the time (as outlined in Remark 2.7). The axioms represent the most basic rewrites that transform one diagram into another semantically equivalent diagram. We note that these rewrites are inherently bidirectional, i.e. the transformation they apply has no particular direction in which it does not hold. As the calculus we use is sound and complete, we can write the equivalence $D_1 = D_2$ if and only if there is a proof of this equivalence using the axioms above.

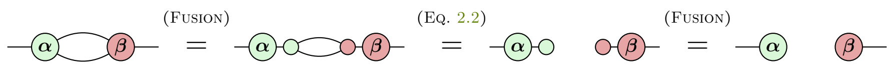
One of the most useful examples is the *Hopf rule* [KvdW24], which says that pairs of legs between red and green spiders can be removed:



(2.2)

[1]We use the 'original' variant of the (Euler) rule referenced in [Bac14] that, with some of the other rules, can be shown to be equally powerful as all other variants in common use [BPW17]. Furthermore, we use 0 as syntactic sugar to denote the diagram representing the all-zero linear map.

In the last step we used the fact that we view diagrams up to a global phase and may thus remove the scalar diagram. Derived rewrite rules, just like axioms, hold in every diagrammatic context, so we can utilise the Hopf rule to derive more complex rules. For example, through application of (Fusion), we can generalise the Hopf rule to spiders with arbitrary phases:



A significant part of this work consists of motivating, introducing, and applying new rewrites. However, we will put restrictions on most of these rewrites: The diagrams that they transform must, at least to some degree, exhibit the same behaviour under noise. We will now move toward formalising noise and fully specifying these additional restrictions.

2.3 Faults

Following a common framework used for stabiliser codes [Got97; DP23; Bac+17; Got22], we only consider faults describable by Pauli operators. As we consider diagrams up to a global phase, we may also take faults up to a global phase and draw them from $\overline{\mathcal{P}^n}$. The notion of noise models used in this work is widely based on previous work from [RPK25], and includes some generalisations for weighted models. Since ZX diagrams generalise quantum circuits, the definitions made here may be reused for circuits.

We remark that while taking faults from $\overline{\mathcal{P}^n}$ like we do is common, it may not be sophisticated enough to model all faults that occur physically. To start, we make the assumption that there is a base set of *independent* Pauli operators that occur physically, and everything else is just a combination of these operators. This assumption is commonly made when analysing faults and correction codes [Got97, Sec. 2.4], so we allow ourselves the simplifications in modeling and reasoning that it enables. Furthermore, considering these Pauli operators without their linear combination removes the possibility to analyse non-Pauli noise channels, e.g. the *amplitude damping* channel[2]. Again, this is a restriction that is common in the field [Gid21a; Got97]. We close the remark by recognising that while handling amplitude decay precisely is difficult, approximations of corrections for this channel can still lead to discovery of good quantum error correction codes [Leu+97].

When modeling faults, one needs to clarify where faults may occur. In a circuit, these *fault locations* are commonly taken as the qubit lines at each timestep, resulting in Pauli faults considered in *spacetime*. For a ZX diagram, the lack of a canonical placement of world lines prevents this. However, the idea generalises to considering all edges of the diagram as possible fault locations:

[2]Analysing this channel is also further hindered by requiring a sense of 'time' that ZX diagrams may lack. In some cases we can get around this which we discuss briefly in Section 2.4.2.

Definition 2.8 (Faults)**.** Let D be a ZX diagram with edges E. A *fault* F on D is an element in $\overline{\mathcal{P}^{|E|}}$, which associates a Pauli rotation with each edge in E. Applying F to D yields a new diagram D^F where all Pauli operators in F are applied on the edges from D associated with them.

We say that a fault acts *trivially* on an edge e if the operator in F associated with e is I, and it acts *non-trivially* on e otherwise.

For faults, we define two different forms of equivalence. One requires that the Pauli decomposition of the faults must be the same, while the other requires that the resulting diagrams are semantically the same:

Definition 2.9 (Congruence, Semantic equivalence)**.** Let D be a ZX diagram with edges E. Two elements $F_1, F_2 \in \overline{\mathcal{P}^{|E|}}$ are *congruent*, written $F_1 \equiv F_2$, if

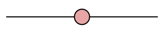
$$F_1 = \bigotimes_i P_{1,i} \quad F_2 = \bigotimes_i P_{2,i} \qquad \text{and} \qquad P_{1,i} = P_{2,i} \text{ for all } i$$

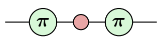
Further, F_1, F_2 are *semantically equivalent*, written $F_1 = F_2$, if $D_1^{F_1} = D_2^{F_2}$.

Based on these equivalences, we define:

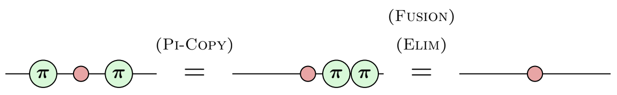
Definition 2.10 (Trivial and detectable faults)**.** Let D be a ZX diagram with edges E. We denote *the* trivial fault by $I \equiv I^{|E|}$. An arbitrary fault F is *trivial* if $D^F = D^I = D$ and *detectable* if $D^F = 0$.

The difference between *the* trivial fault I and some *other* trivial fault F becomes clearer when viewing an example:





On the LHS, the fault instantiated on the diagram consists entirely of identity rotations, i.e. $I_{\mathsf{LHS}} \equiv I \otimes I$, while the RHS features two Z-axis rotations, i.e. $Z \otimes Z$. However, the RHS can be shown to be equal to the LHS via the rules of the ZX calculus:



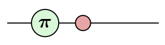
For this work, it will be imperative to distinguish faults that have the same semantic effect on the diagram but that are *not the same fault.* In Section 3.1 and Section 5 we introduce additional notions of equivalence to further refine this.

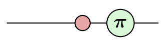
In the example above we have made the assumption that the underlying diagram is susceptible to a $Z \otimes Z$ fault. This assumption might not always hold, e.g. diagrams might be susceptible to different faults depending on how they are physically implemented (if you can implement them directly at all). In other cases, multiple implementations might admit the same faults, but feature different distributions over them [Got97].

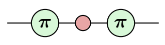
We must thus allow more freedom in our specifications and model not only which faults might occur on a system, but also which likelihood of occurrence it has. For this specification we introduce the notion of a ‘noise model’.

2.4 Noise Models

Let us view a slightly more intricate variation of the last example, where we model that *only* the two faults $Z \otimes I$ and $I \otimes Z$ happen individually. We realise that they can form $Z \otimes Z$ if both happen together:



and


$\rightsquigarrow$


The diagram is subject to a fault that is congruent to $Z \otimes Z$ without us modeling that this fault occurs on its own. This is similar to how two independently occurring bit flips in a classical computer may cancel out, potentiate their effect, or simply coexist without interaction.

In such scenarios we would like to distinguish faults that actually happen physically and faults that may just exist as consequences of multiple other faults occurring together. We call the former *atomic faults*, while the latter non-atomic faults are also referred to as *composite faults*, since they may be expressed as a product of atomic faults. Once we fix a set of atomic faults, all composite faults that may arise are fully characterised via all such products. This is the basis of what we call a *noise model*: A collection of atomic faults. We note that this collection may contain Pauli operators that affect more than one edge, and they do not necessarily have to contain operators for all edges, i.e. not all possible fault locations have to be used.

This notion is further enhanced by allowing a noise model to quantify which atomic faults are *more likely to occur*, which we do for two reasons. First, as outlined above, the assumption that all faults are distributed equally to occur is poor in reality [Got97, Sec. 2.4], so we allow modeling some distinctions. Second, this quantification will become useful in reasoning about the degree to which some noise model is equal to another, in case they are not fully equal.

In the following, we characterise noise models from two different perspectives that make these requirements precise and formal:

1. Referred to as the introduction of *weights* to a noise model, this perspective allows providing an 'unlikelihood' to atomic faults[3]. The weight of a composite fault is directly derived from the weights of the atomic faults of which it is a composite.

2. Referred to as *restrictions* to a noise model, this perspective allows limiting the contents of the set of atomic faults. This provides helpful terminology to refer to different 'kinds' of noise models.

The combination of both perspectives then provides us with a framework to reason about fault equivalence and related notions.

[3]We will take higher weight to correspond to a lower likelihood. This facilitates unification with existing notions of weight, as we will see in Remark 4.7.

2.4.1 Weights

For this perspective, we require a noise model to assign atomic faults a finite integer non-zero *weight*: The higher the weight of the fault, the less likely it is modeled to be. The weight of a composite fault F is then the minimal sum of weights from atomic faults that compose to F, meaning that our initial assignment of weights to atomic faults induces a weight assignment for composite faults. We identify two edge cases for a composite fault F:

1. F is not achievable with the given atomic faults, i.e. there is no combination of atomic faults that yields F, in which case it is assigned composite weight ∞ .
2. F is composed of *no* atomic faults, i.e. $F \equiv I$, in which case it is assigned composite weight 0 .

We thus treat the trivial fault I which models that exactly nothing is happening, as a fault composed of exactly nothing, so it would be inconsistent for our noise models to include I as an atomic fault.

Further, let us clarify why the weights we assign to an atomic fault F should be finite and non-zero:

- The weight ∞ would model that F never occurs, but we can also simply exclude F from the atomic fault set.
- The weight 0 would model that F always occurs[4]. Since we exclude I from the atomic fault set, F must be non-trivial. But a non-trivial fault that always occurs is not really a fault, but instead a fixed part of the diagram and should be implemented as such.

We now formalise:

Definition 2.11 (Weighted noise model)**.** Given a ZX diagram D with edges E, a *weighted noise model* $\mathcal{F}$ is a pair (A, awt) containing

- a set $A \subseteq \overline{\mathcal{P}^{|E|}} - \{I\}$ of *atomic faults*, and
- an *atomic weight function* $\mathrm{awt} : A \to \mathbb{N}_{\neq 0}$.

The set of all faults that can occur is the group $\langle A \rangle \subseteq \overline{\mathcal{P}^{|E|}}$, also denoted as $\langle \mathcal{F} \rangle$. An element $F \in A$ may be alternatively written as $F \in \mathcal{F}$.

Composite faults are created from atomic faults, although there may be more than one combination that leads to the same composite. Thus, we formalise the sets of atomic faults which provide us with a particular composite fault:

[4]Alternatively, one could take the weight 0 to model that a fault F has no 'cost', i.e. it *could* but does not have to be added to every other fault without changing its weight. However, since we use weight to model (un-)likelihood, we use the smallest weight (set as 0) to model the largest likelihood (occurs always).

Definition 2.12 (Composing sets)**.** Let $\mathcal{F}$ be a noise model for some ZX diagram with edges E, and $F \in \overline{\mathcal{P}^{|E|}}$. A set $\widetilde{F} \subseteq \mathcal{F}$ is *composing* for F if

$$\prod_{F_i \in \widetilde{F}} F_i \equiv F\,.$$

Let us now provide weights for all faults that are generated by our noise model:

Definition 2.13. An atomic weight function $\operatorname{awt}(\cdot)$ from a noise model $\mathcal{F}$ induces a weight function $\operatorname{wt} : \overline{\mathcal{P}^{|E|}} \to \mathbb{N}^\infty$ where

$$\operatorname{wt}(F) = \min_{\widetilde{F} \text{ is composing for } F} \sum_{F_i \in \widetilde{F}} \operatorname{awt}(F_i)$$

providing for a fault F the minimum weight with which F is pieced together from atomic faults.

We find that using this definition, we are consistent with the edge cases identified earlier:

- Edge case 1: A (non-trivial) fault F that is impossible to achieve with the given atomic faults has no composing sets. The computation of $\operatorname{wt}(\cdot)$ thus takes an empty minimum, which we take to default to the upper bound in $\mathbb{N}^\infty$, i.e. we get $\operatorname{wt}(F) = \infty$.
- Edge case 2: The trivial fault can be constructed from no faults, i.e. $\emptyset$ is a composing set for I. We take an empty sum to default to 0, so the computation of $\operatorname{wt}(\cdot)$ takes a minimum containing 0, i.e. we get $\operatorname{wt}(I) = 0$.

The concept of weights is not particularly new in the FTQC toolbox. For example, a multi-qubit Pauli operator is established to have a certain weight w, if it contains exactly w non-trivial Pauli components. Building on this, [RPK25, Def. 2.3] defines the weight of a fault F to be the number of 'things' that need to go wrong, i.e. the number of atomic faults required to generate F. When taking the atomic faults in the noise model to be all Pauli operators with exactly one non-trivial component, as in [RPK25, Def. 5.2], these two definitions coincide.

We now see that we can implement such 'noise models' from [RPK25, Def. 2.3] in our framework. Such noise models consist of just a set of atomic faults, so every atomic fault F is implicitly assigned an atomic weight $\operatorname{awt}(F) = 1$. The induced weight of composite faults then already aligns with [RPK25, Def. 2.3]: Take $F_1, \ldots, F_n$ to be the minimal number of atomic faults such that $\prod F_i \equiv F$, then since $\operatorname{awt}(F_i) = 1$ for all F_i, we directly have $\operatorname{wt}(F) = n$. Since the weight differences of atomic faults are what distinguishes our weighted noise models from [RPK25, Def. 2.3], we refer to noise models with $\operatorname{awt}(\cdot) = 1$ as *unweighted noise models.*

While our definition of assigning atomic faults weights independent of their Pauli decomposition may seem like a strict generalisation, we will see in Theorem 4.6 and Remark 4.7

that this is not the case: Everything that we model as an atomic fault with a specific weight can be modeled through the above notions of weight, albeit taking some extra steps.

Weighted noise models as defined here have a key property that we call ‘adversariality’. To determine the weight of a composite fault in Definition 2.13, we take the *minimum* of all possible weight sums that may produce this fault. We thus associate the ‘likelihood’ of a fault to be equal to that of the most likely composition of atomic faults. This enables worst-case reasoning, also referred to as adversarial reasoning.

We conclude by generalising from such weighted noise models with adversariality, distinguishing two possible approaches:

1. A noise model may assign precise *probabilities* to atomic faults, providing a continuous rather than a discrete likelihood specification.

2. A noise model may not obey adversariality.

Most results of this work can be lifted to such more general definitions of Pauli noise models. This is especially true for results that do not require purely diagrammatic reasoning, e.g. when using tools described in Section 7. We will provide a start to these generalisations and highlighting difficulties as well as nuances otherwise ignorable through adversariality in Section 8. A full proof of completeness in Section 6 is only provided for adversarial weighted noise models while a full, precise, and formal generalisation is left for future work.

2.4.2 Restrictions

So far, we were mainly concerned with the weight and thus likelihood of atomic faults. However, we can also restrict the set of atomic faults, yielding different kinds of noise models. Furthermore, although a model $\mathcal{F}$ is already specific to a ZX diagram, there are multiple ways to generate ‘canonical’ noise models given a diagram. Perhaps the easiest is the type of models that considers each edge of the diagram to be independent and thus potentially able to generate a fault on its own:

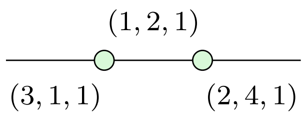
Definition 2.14 (Edge flip noise model)**.** Let D be a ZX diagram with edges E. A noise model $\mathcal{F}$ is an *edge flip noise model* for D if all $F \in \mathcal{F}$ have exactly one non-trivial Pauli operator.

Edge flip noise models have a distinct advantage: Atomic faults can be described by a single operator-edge-weight combination. There are three possible Pauli operators, so we can fully capture the noise model in the diagram by annotating each edge with a 3-tuple of weights associated with the (X, Y, Z) operators.

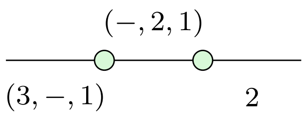
Notation 2.15. We will annotate an edge with a tuple (w_X, w_Y, w_Z) to reference that the edge flip noise model in use for this diagram provides edge flips as atomic faults for this edge, with the weights w_X, w_Y, w_Z respectively. The tuple may contain a special marker '$-$' to mean that the particular edge flip is not part of the noise model. Further, we abbreviate $(-,-,w_Z)$ simply with w_Z, which we will use extensively in Section 4 and beyond when we unify all atomic faults to single-edge Z-flips.

Edges without annotation may represent free variables in the noise model, neither specifying which edge flips are part of the noise model nor which weights they have. During derivations, we will parametrise the annotation to denote arbitrary but fixed weight values.

We illustrate this idea reusing the same diagram with different annotated noise models:



full edge flip noise



sparse edge flip noise

The LHS fully annotates each edge with a weight, while the RHS omits some edge flips from its noise model. Note that the rightmost edge only features a Z-edge flip with a weight of 2.

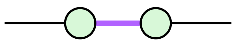
There may also be different restrictions than edge flip models. An example is the *circuit level noise model* as presented in [RPK25, Def. 2.4], which requires identifying parts of the ZX diagram as quantum circuit components. This is not always possible, as some ZX diagrams lack a clear interpretation as a circuit, e.g. when there is no clear flow of time. However, it may still make sense to use this model: If the ZX diagram was just created from a circuit without additional rewrites we still know the component structure. We can then reason about properties of this noise model and compare it to some edge flip noise model, or how these properties change when we start to rewrite the diagram. Note that the circuit level noise model may feature atomic faults containing more than one non-trivial Pauli operator (e.g. for faulty multi-qubit measurements); such a noise model cannot be faithfully represented by an edge flip noise model without additional tools. Finally, using weighted circuit level noise we could create a model approximating 'depolarising noise' [CB18, Sec. 1.2] [Got97], which is a standard definition in existing literature.

In some cases, noise models may specify that some edges of a diagram do not generate any noise. This is particularly simple to express for edge flip models: We simply remove the atomic faults acting on that edge from the atomic fault set, leaving the noise generated on the other edges intact. These noise models are not physically realistic, but they still pose a great mathematical and visual reasoning tool, so we formalise them:

Definition 2.16 (Idealised edge)**.** Let D be a ZX diagram and $\mathcal{F}$ a noise model for D. An edge e of D is an *idealised edge* if there is no $F \in \mathcal{F}$ that acts non-trivially on e.

We will visually mark edges as idealised by using the colour purple:

Example 2.17. Noise models for the diagram

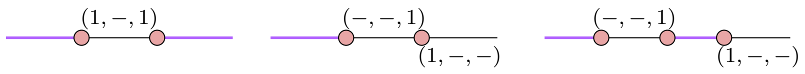


$$\text{may contain some of } \left\{ \begin{matrix} I \\ \otimes \\ I \\ \otimes \\ X \end{matrix}, \begin{matrix} I \\ \otimes \\ I \\ \otimes \\ Z \end{matrix}, \begin{matrix} Z \\ \otimes \\ I \\ \otimes \\ I \end{matrix}, \begin{matrix} Y \\ \otimes \\ I \\ \otimes \\ Z \end{matrix}, \begin{matrix} Z \\ \otimes \\ I \\ \otimes \\ X \end{matrix}, \right\} \text{ but not of } \left\{ \begin{matrix} I \\ \otimes \\ Z \\ \otimes \\ I \end{matrix}, \begin{matrix} I \\ \otimes \\ X \\ \otimes \\ I \end{matrix}, \begin{matrix} Z \\ \otimes \\ X \\ \otimes \\ Z \end{matrix}, \begin{matrix} Y \\ \otimes \\ Y \\ \otimes \\ I \end{matrix}, \right\}$$

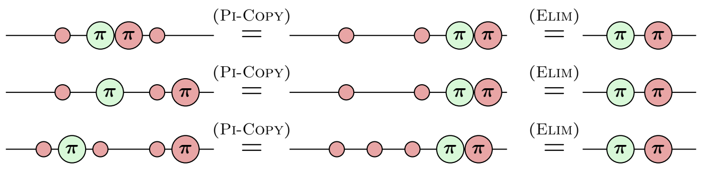
Note that if $\mathcal{F}$ is such that all edges of D are idealised, the only fault possible is the trivial fault I. We call such a diagram *fully idealised*.

2.5 Fault Equivalence

We might encounter two noise models on the same diagram, or even two noise models on different diagrams, that are to some degree equivalent. To illustrate, consider the following three diagrams with annotated edge flip noise:



All three diagrams are semantically equivalent by simple application of (Elim). Additionally, the potential faults produced by these diagrams with their noise models are the same when considering how they might be pushed to the boundary via (Pi-Copy), e.g. all three noise models may produce a Y edge flip:



Just from viewing the annotations such relationships may not be directly visible. The formulation of such equivalences becomes even more involved when considering noise models that leverage weights, i.e. have $\mathrm{awt}(F) \neq 1$ for some atomic F. We thus reserve this section to make these formulations precise under the notion of 'fault equivalence'.

Fault equivalence was first defined in [RPK24] as 'distance-preservation' and later refined in [RPK25], where both definitions are specifically considering unweighted edge flip noise. We lift this concept to arbitrarily weighted noise models as defined in Definition 2.11. To do this, we consider a non-symmetric version of the equivalence:

Definition 2.18 (Fault boundedness, w-fault boundedness)**.** Let D_1, D_2 be ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$. The diagram D_1 under $\mathcal{F}_1$ is *w-fault-bounded* by D_2 under $\mathcal{F}_2$, written $D_1 \underset{w}{\hat{\leq}} D_2$, if and only if for all faults $F_1 \in \langle \mathcal{F}_1 \rangle$ where $\mathrm{wt}(F_1) < w$, we have either:

1. F_1 is detectable, or

2. there exists a fault $F_2 \in \langle \mathcal{F}_2 \rangle$ such that:

$$\mathrm{wt}(F_2) \leq \mathrm{wt}(F_1) \quad \text{and} \quad D_1^{F_1} = D_2^{F_2}$$

A diagram D_1 is *fault-bounded* by D_2, written $D_1 \mathrel{\widehat{\leq}} D_2$, if $D_1 \mathrel{\underset{\infty}{\widehat{\leq}}} D_2$. If the diagrams D_1, D_2 are the same, we also write $\mathcal{F}_1 \mathrel{\underset{w}{\widehat{\leq}}} \mathcal{F}_2$ and $\mathcal{F}_1 \mathrel{\widehat{\leq}} \mathcal{F}_2$.

With this definition, we can express that if we e.g. replace D_2 with D_1 in a ZX rewrite, the bad / undetectable noise exhibited by the quantum program does not get worse. Recall that a lower weight represents a higher likelihood of the fault occurring. Then we can read the definition as: The likelihood of every undetectable fault in D_1 is bounded by the likelihood of its equivalent in D_2, at least for the most common faults, i.e. those with weight below w. Increasing w then naturally increases the strength of such guarantees.

The range of ZX diagrams that *can* be in such a relationship is already constrained: Fault boundedness implies semantic equivalence between the diagrams. Thus, fault boundedness (and later fault equivalence) is a stronger relationship than semantic equivalence. In fact, we can already infer this from 1-fault boundedness:

Proposition 2.19. Let D_1, D_2 be two non-zero ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$ such that $D_1 \mathrel{\underset{1}{\widehat{\leq}}} D_2$. Then $D_1 = D_2$ holds.

Proof. Through 1-fault boundedness, we can consider the trivial fault I_{D_1} as it has $\mathrm{wt}(I_{D_1}) = 0 < 1$. It is undetectable, since $D_1^{I_{D_1}} = D_1 \neq 0$ by assumption. The only fault F_2 that has $\mathrm{wt}(F_2) \leq 0$ on D_2 is $F_2 \equiv I_{D_2}$. So then fault boundedness mandates

$$D_1 = D_1^{I_{D_1}} = D_2^{I_{D_2}} = D_2 \,. \qquad \square$$

In the special case that $\mathcal{F}_1$ does not contain any atomic faults, D_1 is fault-bounded by any semantically equivalent D_2. Although this result is rather simple, we will use it often to justify fault boundedness of rewrites. Thus, we state:

Proposition 2.20. Let D_1 be a ZX diagram with a noise model $\mathcal{F}_1$ such that all edges of D_1 are idealised. Then for any semantically equivalent diagram D_2 under any noise model $\mathcal{F}_2$, $D_1 \mathrel{\widehat{\leq}} D_2$ holds.

Proof. On D_1, the only fault possible under $\mathcal{F}_1$ is the trivial fault I_{D_1}, for which we find the trivial fault I_{D_2}. Since D_1, D_2 are semantically equivalent, $D_1^{I_{D_1}} = D_1 = D_2 = D_2^{I_{D_2}}$ holds. Additionally, $\mathrm{wt}(I_{D_1}) = 0 = \mathrm{wt}(I_{D_2})$ by definition of $\mathrm{wt}(\cdot)$. $\square$

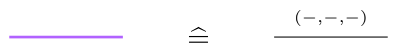
To define fault equivalence, we can now simply require symmetric fault boundedness:

Definition 2.21 (Fault equivalence, w-fault equivalence)**.** Let D_1, D_2 be ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$. The diagram D_1 under $\mathcal{F}_1$ is *w-fault-equivalent* to D_2 under $\mathcal{F}_2$, written $D_1 \underset{w}{\hat{=}} D_2$, if and only if $D_1 \underset{w}{\hat{\leq}} D_2$ and $D_2 \underset{w}{\hat{\leq}} D_1$. We call D_1 and D_2 *fault-equivalent* if $D_1 \hat{\leq} D_2$ and $D_2 \hat{\leq} D_1$. If the diagrams D_1, D_2 are the same, we also write $\mathcal{F}_1 \underset{w}{\hat{=}} \mathcal{F}_2$ and $\mathcal{F}_1 \hat{=} \mathcal{F}_2$.

Similar to regular ZX rewrites, which consist of an asserted equivalence $D_1 = D_2$, fault equivalence gives rise to asserted equivalences of the form $D_1 \hat{=} D_2$ w.r.t. some noise models $\mathcal{F}_1, \mathcal{F}_2$, or similar for other relationships like fault boundedness. This naturally enables thinking about these equivalences as *fault-equivalent* rewrites, similar to their definition in [RPK25].

Note that, unlike in [RPK25], our definition does not fix the noise models $\mathcal{F}_1, \mathcal{F}_2$, so the asserted equivalence has to clarify the noise models in use, if they are not inferrable from context. We will usually do this by annotating the diagrams with the noise model directly, and as we will discover in Section 4.1 this is sufficient to deal with all noise models in this work.

The perhaps simplest example of a fault-equivalent rewrite rule that of a fully idealised edge and an edge annotated such that no fault is happening on it (in the context of an edge flip noise model), which may be seen as defining alternative notation:



On either side there may not be any faults, so the two are clearly fault-equivalent. For additional examples of fault-equivalent rewrites we refer to the entirety of Sections 3 to 6 or [RPK25; RPK24].

Finally, we may now have non-trivial rewrites that keep the underlying diagram the same and only change the noise model. This technique will prove useful, especially when the noise model is visually encoded into the diagram, making the change easier to understand. In fact, we will use such rewrites when defining our proposed set of fault equivalence axioms in Section 3.3.

For a practically useful calculus consisting of fault-equivalent rewrites we need to ensure that these rewrites are compositional (to use them in larger contexts) and transitive (to enable forming logical reasoning chains), just like rewrites built on diagrammatic equivalence are. Indeed, we can show that these properties hold for the most general w-fault boundedness and thus by extension for derived properties like (∞-)fault boundedness and (w-/∞-)fault equivalence. Compositionality and transitivity have already been proven for w-fault equivalence in [RPK25, Prop. 3.11, Prop. 3.12]. We adapt their proof for w-fault boundedness and more general weighted noise models:

Proposition 2.22. Fault boundedness is compositional, i.e. for ZX diagrams D_1, D_1', D_2, D_2' with respective noise models $\mathcal{F}_1, \mathcal{F}_1', \mathcal{F}_2, \mathcal{F}_2'$ it holds that

$$\begin{aligned} D_1 \underset{w_1}{\widehat{\leq}} D_2 \quad \text{and} \quad D_1' \underset{w_2}{\widehat{\leq}} D_2' \qquad &\Longrightarrow \qquad D_1' \circ D_1 \underset{\min(w_1,w_2)}{\widehat{\leq}} D_2' \circ D_2\,, \\ D_1 \underset{w_1}{\widehat{\leq}} D_2 \quad \text{and} \quad D_1' \underset{w_2}{\widehat{\leq}} D_2' \qquad &\Longrightarrow \qquad D_1 \otimes D_2 \underset{\min(w_1,w_2)}{\widehat{\leq}} D_1' \otimes D_2'\,, \end{aligned}$$

where the composed diagrams have noise models uniquely composed of $\mathcal{F}_1, \mathcal{F}_1', \mathcal{F}_2, \mathcal{F}_2'$ under atomic fault set union (while padding faults with I for edges in the other diagrams) and addition of atomic weights.

Proof. We prove the claim for sequential composition, the proof for parallel composition is analogous. Now let $\mathcal{F}_1^\circ, \mathcal{F}_2^\circ$ be the noise models obtained for $D_1' \circ D_1$ and $D_2' \circ D_2$ respectively. These noise models are unique, because the original noise models they are respectively sourced from are operating on disjoint sets of edges, and thus in disjoint spaces of $\overline{\mathcal{P}^{|\cdot|}}$.

We start with considering an undetectable fault $F_1^\circ \in \mathcal{F}_1^\circ$ of weight less than $\min(w_1, w_2)$. F_1° is composed of F_1, F_1' drawn from $\mathcal{F}_1, \mathcal{F}_1'$ respectively, and since $\mathcal{F}_1^\circ$ was formed under addition of atomic weights, it must be that

$$\mathrm{wt}(F_1^\circ) = \mathrm{wt}(F_1) + \mathrm{wt}(F_1') < \min(w_1, w_2)\,.$$

Then via the precondition of w_1, w_2-fault boundedness, we can find semantically equivalent F_2, F_2' drawn from $\mathcal{F}_2, \mathcal{F}_2'$ that have lower weight than F_1, F_1'. Their composite $F_2^\circ = F_2 F_2'$ is then a fault on $D_2' \circ D_2$ that can be drawn from $\mathcal{F}_2^\circ$ and it holds that

$$\begin{aligned} (D_2' \circ D_2)^{F_2^\circ} \quad &= \quad {D_2'}^{F_2'} \circ D_2^{F_2} \quad = \quad {D_1'}^{F_1'} \circ D_1^{F_1} \quad = \quad (D_1' \circ D_1)^{F_1^\circ}\,, \\ \mathrm{wt}(F_2^\circ) \quad &= \quad \mathrm{wt}(F_2) + \mathrm{wt}(F_2') \quad \leq \quad \mathrm{wt}(F_1) + \mathrm{wt}(F_1') \quad = \quad \mathrm{wt}(F_1^\circ)\,. \end{aligned}$$

Thus, there is an equivalent F_2° for F_1° with lower or equal weight. $\square$

Furthermore, we have for transitivity that:

Proposition 2.23. w-fault boundedness is transitive, i.e. for ZX diagrams D_1, D_2, D_3 with respective noise models $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ it holds that

$$D_1 \underset{w_1}{\widehat{\leq}} D_2 \quad \text{and} \quad D_2 \underset{w_2}{\widehat{\leq}} D_3 \qquad \Longrightarrow \qquad D_1 \underset{\min(w_1,w_2)}{\widehat{\leq}} D_3\,.$$

Proof. We start with an undetectable fault $F_1 \in \mathcal{F}_1$ with $\mathrm{wt}(F) < \min(w_1, w_2)$. Then via the precondition $D_1 \underset{w_1}{\widehat{\leq}} D_2$, there is some $F_2 \in \mathcal{F}_2$ with $D_1^{F_1} = D_2^{F_2}$ and $\mathrm{wt}(F_2) \leq \mathrm{wt}(F_1)$. Via the precondition $D_2 \underset{w_2}{\widehat{\leq}} D_3$, we can similarly obtain some $F_3 \in \mathcal{F}_3$ with $D_2^{F_2} = D_3^{F_3}$ and $\mathrm{wt}(F_3) \leq \mathrm{wt}(F_2)$.

But then we owe to the transitivity of diagram equivalence that $D_1^{F_1} = D_2^{F_2} = D_3^{F_3}$ and $\mathrm{wt}(F_3) \leq \mathrm{wt}(F_2) \leq \mathrm{wt}(F_1)$, which completes the claim. $\square$

The results for the remaining three relations derived from w-fault boundedness follow directly from setting $w_1 = w_2 = \infty$, or via symmetry, or both:

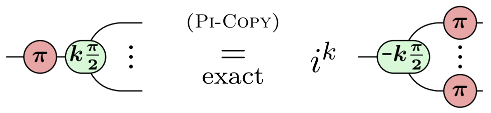
Corollary 2.24. Fault boundedness, w-fault equivalence, and fault equivalence are compositional and transitive.

We will be assuming compositionality and transitivity of fault equivalence throughout this work without explicit mention.

2.6 Pauli Webs

The rewrite rules we have been using thus far are introduced up to a global scalar, along with faults that we handle up to a global scalar as well. We now take another look at Pauli products in stabiliser groups, which as introduced in Section 2.1 still retain a scalar of either 1 or -1. Recall that such a Pauli stabilises a state if and only if the state is a $+1$ eigenvector of the Pauli; as such, keeping track of the scalar of a Pauli is vital.

It turns out that visualising stabilisers in ZX diagrams is a powerful method to understand *how* a diagram can be stabilised, and this will subsequently enable us to speak more precisely about faults in Section 3.1. To find a diagrammatic description of these scalar accurate stabilisers, we require introduction of the scalar-accurate copy rule [BPW17; Bor19]:

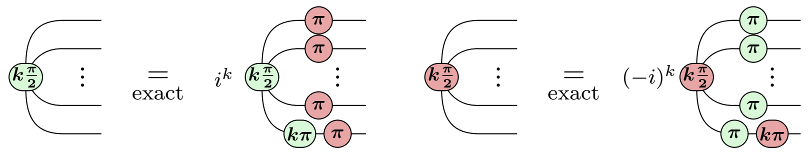


(2.3)

We use the symbol $\underset{\text{exact}}{=}$ to denote an exact and scalar-accurate diagram equality.

Stabilisers for all Clifford spiders can be derived using just this rule, which by definition describes how to introduce π-phases around these spiders without changing the semantics of the diagram. Such an introduction of π-phases is also known as *firing* the spider [Bor19].

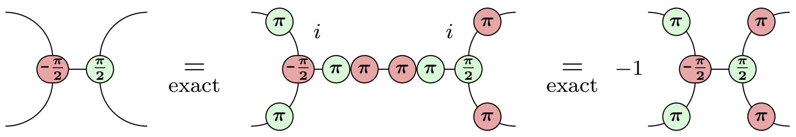
All spiders are stabilised by pairs of π-phases in their own colour, since we can always introduce / eliminate a pair of π-phases with (Elim) and (Fusion). Opposite-colour stabilisers however become more involved:



Note that there is now a colour specific difference in the introduced scalar, which arises from keeping the order of π-phases on the boundary consistent. This difference is only visible for phases of $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$ where k is odd, and vanishes for the remaining phases when k is even.

The need for such consistency is twofold:

- As in Section 2.1, we can describe these stabilisers using the language of Pauli products. In particular, if we keep the order of decomposition of Y consistent, we can easily translate between the Pauli products and the phases that these stabilisers introduce in ZX diagrams. In our case we use $Y = iXZ$, so the phase of the Pauli that describes the stabiliser can be obtained through division by i. We will only work with Pauli products instead of an explicit decomposition, so we only consider phases of $+1, -1$ as i is already removed.
- Multiple stabilisers of single spiders compose to expose an overall stabiliser (with a phase of -1) of a diagram:



Keeping the edges consistent allows the π-phases to cancel out without introducing new scalars through Eq. (2.3). Thus, we can obtain the overall scalar of the stabiliser simply by multiplying all scalars of the local stabilisers of spiders.

To visualise local stabilisers, Pauli webs [Bom+24] were developed as a graphical overlay notation for ZX diagrams. Each Pauli web defines a highlighting of spider legs in green and/or red, visualising how π-phases may be introduced around a spider, i.e. how a spider may be fired. We introduce the *signed* version of Pauli webs, which are stored as Pauli operators and, in addition to the highlighting, track the sign of the overall global phase that is introduced when firing every spider according to the highlighting.

Definition 2.25 (Signed Pauli Web)**.** Let D be a ZX diagram with edges E. An element $P \in \mathcal{P}^{|E|}$ provides a highlighting of edges in D where an edge e is highlighted in red if P_e anticommutes with Z and green if P_e anticommutes with X. Such an operator P is a *Pauli web* if:

- a spider with phase in $\{0, \pi\}$ satisfies:
  - P highlighting an *even* number of its legs in its own colour, **and**
  - P highlighting *all or none* of its legs in the opposite colour
- a spider with phase in $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$ satisfies:
  - P highlighting an *even* number of its legs in its own colour *and none* in the opposite colour, **or**
  - P highlighting an *odd* number of its legs in its own colour *and all* in the opposite colour

The *sign* of a Pauli web P is the sign of the global phase that gets introduced when firing the spiders according to the highlighting.

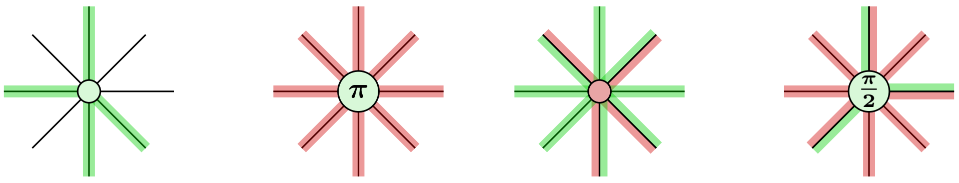
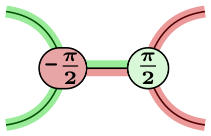
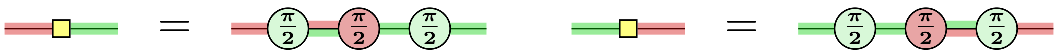


Figure 1: Examples of Pauli webs. The second Pauli web has a sign of -1.

We provide some illustrative examples of single spider webs in Fig. 1. Similar to how local stabilisers compose to larger networks of spiders with a consistency requirement, Pauli webs need to be compatible on a per-edge basis too. That is, highlightings coming from either vertex of an edge must match. For the earlier example we thus get the following web with a sign of -1:



We further provide the two possible webs (up to combinations) for the Hadamard gate, which expectedly inverts the colour between its boundaries:



A natural question is how Pauli webs may be obtained for a given diagram and what properties they have. As Pauli webs are defined through elements of $\mathcal{P}^{|E|}$, we would expect them to form a group under Pauli operator multiplication. This is indeed the case, and through [Bor19, Alg. 3] we find that for a Clifford diagram with n boundary edges, we can find $n+d$ independent generators for the group $\mathcal{W}$ of Pauli webs, for some $d \geq 0$:

$$\mathcal{W} := \langle S_1, \ldots, S_n, R_1, \ldots, R_d \rangle \tag{2.4}$$

Remark 2.26. Viewed end to end, the algorithm from [Bor19, Alg. 3] only specifies the stabilisers for a given diagram. However, an intermediate result of the algorithm is a set of $n+d$ 'firing assignments', which can be uniquely translated to Pauli webs. To be precise, these firing assignments are constructed for an intermediate diagram, known as a *graph like* diagram, which can be deterministically obtained for any Clifford ZX diagram D through introduction of spiders using the rule (Elim). Pauli webs obtained for such graph like diagrams can be easily reversed to Pauli webs for D as (Elim) preserves all stabilisers.

We recall that an n-qubit stabiliser state is uniquely determined by n independent stabiliser generators of the space that only contains that state. For a Clifford diagram with n boundary edges, there analogously are n independent Pauli products from $\mathcal{P}^{|B|}$ that stabilise the diagram. We thus expect to find n independent Pauli webs that highlight some boundary edge.

We then formalise:

Definition 2.27 (Stabilising Pauli web)**.** Let D be a diagram with boundary edges B. A Pauli web P for D is *stabilising* if it highlights at least one boundary edge.

Indeed, the independent generators $S_1, \dots, S_n$ from Eq. (2.4) are stabilising webs, corresponding to a unique independent stabiliser generator each [Bor19, Alg. 3] with the same sign, which is why we use a similar notation.

It remains to be clarified what the generators $R_1, \dots, R_d$ from Eq. (2.4) correspond to. They cannot correspond to a new independent stabiliser as we already have n generators for the stabilisers of the diagram. Furthermore, we can take them to not correspond to any stabiliser, i.e. they do not highlight any boundaries: If a generator R_i would correspond to a stabiliser S, we must be able to generate a web w corresponding to S with the generators $S_1, \dots, S_n$, and wR_i yields a generator that does not highlight any boundaries. Instead, such webs are known as *checks* [Bom+24] or *detecting regions* [RPK25; MBG23]:

Definition 2.28 (Detecting Region)**.** Let D be a diagram with boundary edges B. A *detecting region* is a Pauli web P such that P does not highlight any boundary edges. The group of all detecting regions in a diagram is $\mathcal{R} = \langle R_1, \dots, R_d \rangle$, i.e. a subgroup of $\mathcal{W}$.

We must conclude that the correspondence between stabilising webs and stabiliser generators is not one-to-one, since we can always multiply a stabilising web with a detecting region to receive another stabilising web that by construction highlights the same boundary edges. One may find a 'canonical' stabilising web by only using the web generators $S_1, \dots, S_n$ from Eq. (2.4); however, we will only use stabilising webs to map to stabilisers, not vice versa, thus we are not concerned with finding such a canonical web in this work.

Detecting regions possess a key property: Firing spiders according to them does not change the diagram save for introduction of the sign that the region carries. Since webs must be compatible on a per-leg basis, once we fire each spider locally, there are two π-phase spiders of the same type on the same edge, which cancel out. This culminates in the following:

Proposition 2.29. *Let D be a ZX diagram with a detecting region P such that P has sign -1. Then $D = 0$.*

Proof. Firing the detecting region P produces a new diagram that *is* D, up to a phase of -1. But then $D = -D$, which can only be satisfied by $D = 0$. □

Corollary 2.30. *If D is a non-zero diagram, every detecting region of D has positive sign.*

We are now ready to see why they are called 'detecting' regions: They allow us to detect faults! Since faults (be it single-edge or multi-edge) introduce new π-phase spiders into a diagram, they may affect its Pauli webs. However, the webs highlighting itself remains invariant, and faults may only influence the sign of webs [Bom+24].

Formally, the *sign* of a web P is flipped by a fault $F \in \overline{\mathcal{P}^{|E|}}$ exactly when odd many π-phase spiders are introduced by F on edges highlighted by P in an anticommuting colour (since the web locally forms a -1 stabiliser for each π spider), i.e. when F anticommutes with P:

Definition 2.31. A fault $F \in \overline{\mathcal{P}^{|E|}}$ is said to be *detected* by a detecting region P if F flips the sign of P, i.e. when $FP = -PF$.

Of course, if F is detected by some detecting region, it must be detectable by at least one generator for the detecting region group $\langle R \rangle$.

To recall Definition 2.10, a fault F is called detectable if $D^F = 0$. It would be great if these two definitions of detectability coincide. Indeed, they do, but we have to show a useful intermediate result first:

Proposition 2.32. Given a stabiliser group $\mathcal{S} \subseteq \mathcal{P}^N$ of $n \leq N$ independent generators $S_1, \ldots, S_n$, for each S_i, we can find an operator $T \in \mathcal{P}^N$ that anticommutes only with S_i.

Proof. By [Bor19, Lem. 2.5.10] we may choose an $m < n$ and find an operator T that commutes with $S_1, \ldots, S_m$ and anticommutes with $S_{m+1}, \ldots, S_n$. But we may always choose $m = n - 1$ and rearrange the generators to successively find these T for every S_i. □

We then finally show:

Proposition 2.33. Let D be a non-zero ZX diagram. A fault F is detectable if and only if there is a detecting region P in D that detects F.

Proof. 'if' / $\Leftarrow$: Assume a detecting region P that detects F. Since D is non-zero and due to Corollary 2.30, P must have positive sign in D. As F flips the sign of P, P has negative sign in D^F. Then by Proposition 2.29, $D^F = 0$ holds.

'only if' / $\Rightarrow$: Assume for the contrapositive that there is no detecting region that detects F. Then F can at most flip some of the independent *stabilising* Pauli web generators $S_1, \ldots, S_n$ from Eq. (2.4), which by definition have support on the boundary. Let S_F be the set of those sign-flipped independent stabilising Pauli webs. Further, for a stabilising web $s \in S_F$, let s_B be the corresponding stabiliser generator on the boundary. We define

$$S_B \coloneqq \{s_B \mid s \in S_F\}$$

to be the set of stabilisers corresponding to the stabilising webs that were flipped.

As all webs in S_F are independent and are the only stabilising webs in *Eq.* (2.4), the stabilisers in S_B must be independent too. Then through Proposition 2.32, we can find an operator T_i for each $s_i \in S_B$, such that T_i anticommutes only with s_i.

Then, we can compose a new fault $F' = \prod_{s_i \in S_B} T_i$, which by construction flips exactly those webs that F flips as well. So D^F and $D^{F'}$ must be stabilised by the same stabilisers, and thus the stabiliser spaces of D^F and $D^{F'}$ are the same, i.e. we have $D^F = D^{F'}$. But F' exists purely on the boundary of the diagram, and since *composing* a diagram with single edge flips on its boundary never sends it to zero and $D \neq 0$ by assumption, we have $D^{F'} \neq 0$ and thus $D^F \neq 0$. $\square$

3 Completeness - Part I: Foundations and Setup

The ZX calculus has a comparatively long history of completeness results. That considers completeness for semantic equivalence, i.e. for any two ZX diagrams that are semantically equivalent this relationship can be proven with the rules of the ZX calculus.

The first iteration of the calculus brought forward in [CD11] did not contain the Euler decomposition rule from Eq. (2.1). This version was shown to be incomplete even for the Clifford fragment in [DP09]. Adding the Euler decomposition rule yields the version of the Clifford ZX calculus that we use, which (as the name suggests) is complete for semantic equivalence between Clifford ZX diagrams [Bac14], but incomplete outside this fragment [SZ14]. Completeness for all ZX diagrams requires at least a single additional rule [Vil19], even though larger complete axiomatisations exist [NW17].

Obtaining these completeness results even beyond semantic equivalence of diagrams serves as an important motivation of the ZX community, since *proving* your calculus can prove every relationship you are potentially interested in makes it distinctly more powerful. Thus, it is of major interest to prove whether there is a set of fault-equivalent rewrites that can prove every relationship of fault equivalence for two diagrams, i.e. a set that is complete for fault equivalence. The present Sections 3 to 6 of this work emerge precisely in response to this interest. The completeness result itself requires a considerable number of intermediate results and its proof is split across these sections.

As a broad overview, the proof commences as follows: We introduce a new notion of equivalence for faults, and derive a special function that assigns each new equivalence class a weight, such that the function is provably unique for a diagram D and its noise model $\mathcal{F}$. Next, we restate fault boundedness (and thus fault equivalence) as an element-wise comparison of this function. We proceed to show that we can rewrite D along with $\mathcal{F}$, first into a form that encodes $\mathcal{F}$ into D fault-equivalently, and afterwards extract the aforementioned weight function for D using a small set of fault equivalence axioms that operate on edge flip noise only. This involves separating the now fault-free diagram D and the diagrammatic representation of the weight function, removing all potential noise from edges internal to D. Finally, the fault-free D and the representation of the weight function both have a normal form, yielding an overall normal form and thus completeness.

This first section addresses three key points:

- Providing an introduction to the new notion of equivalence for faults and restating fault boundedness with it.
- Motivating, deriving and stating the collection of axioms that we require for completeness.
- Proving that the axioms are *sound*, i.e. the diagrams they describe and transform obey fault equivalence.

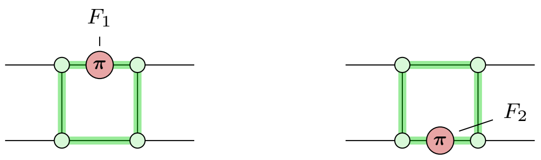
Afterwards in Sections 4 and 5 we will derive the intermediate results required for the final proof, which concludes in Section 6.

3.1 Fault Effects

We recall that a noise model as defined in Definition 2.11 specifies a set of atomic faults A (without the trivial fault) and a function $\text{awt} : A \to \mathbb{N}_{\neq 0}$ that provides atomic weights for the atomic faults. This function in turn induces a weight function $\text{wt} : \overline{\mathcal{P}^{|E|}} \to \mathbb{N}^{\infty}$ that provides weights for possible faults in the diagram. The induced weight for a fault F is then given as the minium sum of atomic weights for atomic faults that compose to exactly F. For faults that cannot occur we use the weight ∞, for the trivial fault we use the weight 0.

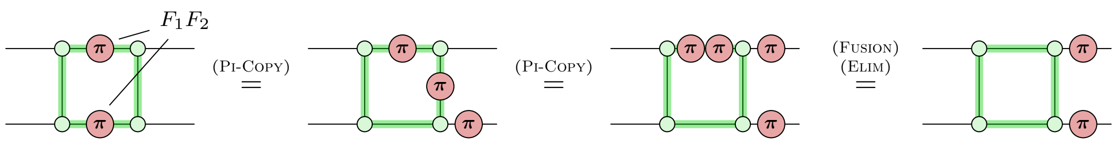
When studying fault equivalence of diagrams using this weight function we may encounter some redundancies: Many of the atomic faults could be semantically equivalent to each other even if they are not the same Pauli operator (i.e. they are not congruent), with even more composite faults being equivalent as a result. Further, we may try to optimise and eliminate these redundancies by removing one of two atomic faults F_1, F_2 if $D^{F_1} = D^{F_2}$. However, this can result in inconsistencies and ultimately becoming unsound:

Example 3.1. Consider the following diagram D with a noise model that allows only the faults F_1, F_2 on D:



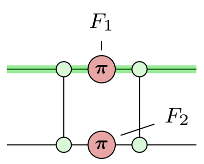
Both faults are detectable F_1, F_2 by the marked detecting region, so $D^{F_1} = 0 = D^{F_2}$, and we would expect that we can eliminate one of the faults as they may seem redundant.

However, this would change the set of *undetectable* faults induced by the noise model. In particular, the composite fault F_1F_2 flips the detecting region twice, returning it to positive sign and making the fault undetectable. We can show through ZX rewrites that the composite fault is semantically equivalent to one that is non-trivial and outside the detecting region:

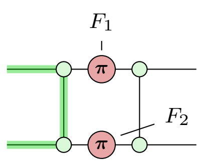


Removing one of F_1, F_2 would remove this undetectable fault, and since the noise model does not allow other atomic faults we cannot recover it. We would thus have changed our noise model unexpectedly!

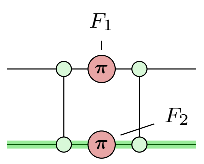
So we want to consider a finer grained equivalence for identifying redundancies in atomic faults. We take one key observation away from Example 3.1: The atomic faults flip different Pauli webs. In addition to the shown detecting region the stabilising webs of the diagram are generated by four independent stabilising webs. We can visualise how the faults interact with them:



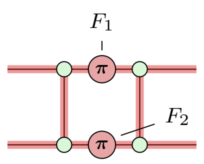
S_1: Flipped only by F_1



S_2: Flipped by no fault



S_3: Flipped only by F_2



S_4: Flipped by no fault

As composition of faults naturally leads to considering all web flips together, some of these flips may cancel out again. In the above case we had one web flip remaining, which is exactly the web that is also flipped by the undetectable composite fault.

As it turns out, considering which Pauli webs a fault F flips is exactly the right notion for our purposes. In contrast to D^F, which is the 'semantic interpretation' of F, we capture this as a faults *effect*:

Definition 3.2 (Fault effect)**.** Let D be a ZX diagram with a Pauli web group $\langle W \rangle$, and let F be a fault from the corresponding noise model $\mathcal{F}$. The *effect* of F is given as a set

$$\text{eff}(F) := \{w \mid w \in W : wF = -Fw\}\,,$$

which describes the web generators that anticommute with F. The set of all effects from $\mathcal{F}$ is given by

$$\text{Eff}(\mathcal{F}) = \{\text{eff}(F) \mid F \in \langle \mathcal{F} \rangle\}\,.$$

Remark 3.3. While the possibility of capturing fault effects through sign flips of Pauli webs was noted in passing in [Bom+24, Sec. 3] and implicitly used for descriptions of detectable faults in [Bom+24; RPK24; RPK25], these usages do not provide a full formal treatment. Since Pauli webs w.r.t. to faults are essential in almost all upcoming parts of this work, we aim to provide this missing formal framework for fault effects.

As we saw in Example 3.1, two or more faults may combine to yield other faults. Whether the composite fault flips a web w purely depends on whether an odd number of the original faults flipped w. Thus, the effect of the composite fault is a reduction of the effects of original faults:

Proposition 3.4. Let D be a ZX diagram with a noise model $\mathcal{F}$. Further, let $F_1, F_2 \in \mathcal{F}$ and let $F_3 \equiv F_1 F_2$. Then the effect of the composite fault F_3 is

$$\text{eff}(F_3) = (\text{eff}(F_1) \cup \text{eff}(F_2)) \setminus (\text{eff}(F_1) \cap \text{eff}(F_2)) =: \text{eff}(F_1) \oplus \text{eff}(F_2)\,.$$

Proof. Directly, since webs that are flipped by either F_1 or F_2 (but not both) directly carry over to F_3, and webs that are flipped by both F_1 and F_2 cancel out. □

Corollary 3.5. $F_1, F_2 \in \mathcal{F}$ have $\text{eff}(F_1) = \text{eff}(F_2)$ if and only if $\text{eff}(F_1 F_2) = \emptyset$.

From Proposition 2.33 we know that a fault is detectable in particular if it flips the sign of a detecting region. Reversing the implication, we can derive that undetectable faults cannot flip any detecting regions:

Definition 3.6 (Undetectable effects)**.** Consider a ZX diagram with the detecting region group $\langle R \rangle$. For any undetectable fault F it holds that

$$\mathrm{eff}(F) \cap R = \emptyset \,,$$

making $\mathrm{eff}(F)$ an *undetectable effect*. The set of all undetectable effects in the noise model $\mathcal{F}$ is

$$\mathrm{Eff}_{\neq 0}(\mathcal{F}) = \{W \mid W \in \mathrm{Eff}(\mathcal{F}) : W \cap R = \emptyset\} \,.$$

Undetectable effects by definition contain only stabilising webs, so they can be described purely by the stabilisers these stabilising webs correspond to.

For a diagram with boundary edges B, we will express the *restriction* of a Pauli web w to the boundary as $w|_B$. As we will need to compare the signs of these stabilisers across multiple potentially different (but semantically equivalent) diagrams, we lift this to effect sets:

Definition 3.7. (Boundary restriction for effects) Consider a ZX diagram with boundary edges B and a noise model $\mathcal{F}$. Then for any effect $\mathrm{eff}(F) \in \mathrm{Eff}(\mathcal{F})$, the *boundary restriction* is

$$\mathrm{eff}(F)|_B \coloneqq \{w|_B \mid w \in \mathrm{eff}(F)\} \,.$$

Finally, we can derive that for undetectable effects, evaluating fault effects is exactly as useful as evaluating fault semantics, which holds across diagrams:

Proposition 3.8. Let $D_1 = D_2$ be non-zero ZX diagrams with respective effect functions $\mathrm{eff}_1, \mathrm{eff}_2$. For any two undetectable faults F_1, F_2 from D_1, D_2 respectively, it holds that

$$\mathrm{eff}_1(F_1)|_B = \mathrm{eff}_2(F_2)|_B \quad \Leftrightarrow \quad D_1^{F_1} = D_2^{F_2} \,.$$

Proof. 'if' / $\Leftarrow$: Assume w.l.o.g. for a contrapositive that there is a stabiliser $s \in \mathrm{eff}(F_1)|_B$ such that $s \notin \mathrm{eff}(F_2)|_B$. F_1 flips the sign of s, so s is no longer stabilising $D_1^{F_1}$, but instead $-s$ is.

However, by assumption, $D_1 = D_2$, so both diagrams must be stabilised by the same set of stabilisers. Since F_2 did not flip s, s is still stabilising $D_2^{F_2}$. If now $-s$ was also stabilising $D_2^{F_2}$, we could produce a stabiliser $-ss = -I$, which similar to Proposition 2.29 implies $D_2^{F_2} = 0$. This contradicts F_2 being undetectable, so $-s$ is not stabilising $D_2^{F_2}$ and the stabiliser spaces of $D_1^{F_1}$ and $D_2^{F_2}$ are different, so $D_1^{F_1} \neq D_2^{F_2}$.

'only if' / $\Rightarrow$: Via a similar argument to $\Leftarrow$, if both faults flip the same webs, the stabiliser spaces of $D_1^{F_1}, D_2^{F_2}$ remain the same, so the claim directly follows.

□

We acknowledge that because we assume that $D_1 = D_2$, D_1 and D_2 must have the same number of boundary edges even though they might be different diagrams. Thus, we simplify and make no distinction between the two technically different but isomorphic sets of boundary edges, and simply denote the restriction to the boundary by $\cdot|_B$.

Now suppose that on a diagram D, there are two detectable faults F_1, F_2 that flip the same webs, then by definition, they have $D^{F_1} = 0 = D^{F_2}$. Together with Proposition 3.8 for undetectable faults in the same diagram (where boundary effect equivalence corresponds to effect equivalence), we can directly conclude:

Corollary 3.9. Let D be a diagram with a noise model $\mathcal{F}$. For any two faults $F_1, F_2 \in \langle\mathcal{F}\rangle$ with $\mathrm{eff}(F_1) = \mathrm{eff}(F_2)$ it holds that $D^{F_1} = D^{F_2}$.

As two congruent faults would trivially flip the same webs, we thus can place fault effects *between* the scrutinous congruence and the broad semantic equivalence, i.e. for all faults F_1, F_2 in a noise model we have

$$F_1 \equiv F_2 \qquad \Longrightarrow \qquad \mathrm{eff}(F_1) = \mathrm{eff}(F_2) \qquad \Longrightarrow \qquad F_1 = F_2\,.$$

Proposition 3.8 has another interesting side effect: When considering faults purely on boundary edges, the trivial faults, i.e. those with $\mathrm{eff}(F) = \emptyset$, are exactly the stabilisers of the diagram:

Proposition 3.10. Let D be a non-zero ZX diagram with boundary edges B and effect function eff. For any fault $F \in \overline{\mathcal{P}^{|B|}}$ on the boundary, it holds that $\mathrm{eff}(F) = \emptyset$ if and only if either F or $-F$ stabilise D.

Proof. 'if' / $\Leftarrow$: Since we treat diagrams up to a global non-zero scalar, we have $D^{\pm S} = D = D^I$ by assumption for a stabiliser S. But then by Proposition 3.8, we have that $\mathrm{eff}(S) = \mathrm{eff}(I) = \emptyset$.

'only if' / $\Rightarrow$: Via the contrapositive, if for some fault F on the boundary neither F nor $-F$ stabilises D, we have $D^{\pm F} \neq D = D^I$. But then via Proposition 3.8, it holds that $\mathrm{eff}(F) \neq \mathrm{eff}(I)$ and thus $\mathrm{eff}(F) \neq \emptyset$.

□

So far, we have covered a new notion of equivalence, finer grained than semantic equivalence, that allows us to capture when atomic faults are redundant. Further, as we eliminate all detectable effects from the noise model, this notion approaches semantic equivalence. We will continue by using this idea of approach to discover hidden structures in the notions of fault boundedness, and through it fault equivalence, allowing us to restate both in terms that are ultimately easier to check diagrammatically.

3.2 Restating Fault Boundedness with Effects

Checking fault boundedness as defined in Definition 2.18 is performed by considering each fault F_1 generated by the first noise model $\mathcal{F}_1$ individually: If F_1 is undetectable, we have to find a semantically equivalent fault F_2 in the second noise model $\mathcal{F}_2$ with equal or lower weight than F_1. This F_2 is similar to a 'witness' that F_1 has some equivalent that is at least as likely as F_1. However, we can take a shortcut here: For two undetectable faults that are semantically equivalent, we only have to obtain such a witnessing F_2 for the fault with lower weight. For the remaining fault with higher weight, the same F_2 can be reused. We could thus collapse all semantically equivalent undetectable composite faults into a single equivalence class and only find a witness for the minimum weight in this class.

As outlined, this simplification is sound, so finding a witness for the minimum weight fault in all classes is sufficient for fault boundedness. We reason adversarially, so this case is also necessary as fault boundedness requires at least finding such a witness for the minimum weight fault. This section is reserved to formalise the simplification and the relationship of sufficiency / necessity to fault boundedness.

As we found in Proposition 3.8, we can use fault effects to study semantic equivalence between undetectable faults, while considering effects instead of semantics provides us with more flexibility and expressiveness for detectable faults. Thus, we define our equivalence classes by effect equivalence and identify them directly with the exact effect of the faults within. We observe that mapping a fault to its equivalence class is already performed by the fault effect function $\mathrm{eff}(\cdot)$.

We now introduce weights for effects, considering them as the aforementioned identifiers for entire equivalence classes. As outlined above, the weight of an effect W is the minimum weight of all faults with effect W:

Definition 3.11. Let D be a ZX diagram with effects $\mathrm{Eff}(\mathcal{F})$. The *effect weight function* $\mathrm{ewt} : \mathrm{Eff}(\mathcal{F}) \to \mathbb{N}$ is given by

$$\mathrm{ewt}(W) = \min_{\substack{F \in \langle \mathcal{F} \rangle \\ \mathrm{eff}(F) = W}} \mathrm{wt}(F)\,.$$

We thus decouple finding all faults that we can consider equivalent without becoming unsound (as we would have with semantic equivalence) from finding the minimum weight for a fault. This decoupling allows restating fault boundedness between two diagrams as an element-wise comparison of their ewt functions. It is exactly this comparison that can be done diagrammatically by enumerating all relevant elements of these functions in both diagrams, making the decoupling a key component in the journey to completeness.

We observe:

Proposition 3.12. Let $D_1 = D_2$ be two ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$, undetectable effects $\mathrm{Eff}^{(1)}_{\neq 0}, \mathrm{Eff}^{(2)}_{\neq 0}$ and corresponding effect weight functions $\mathrm{ewt}_1, \mathrm{ewt}_2$.

Then D_1 under $\mathcal{F}_1$ is w-fault-bounded by D_2 under $\mathcal{F}_2$ if and only if for all $W_1 \in \mathrm{Eff}^{(1)}_{\neq 0}(\mathcal{F}_1)$ with $\mathrm{ewt}_1(W) < w$ there is some $W_2 \in \mathrm{Eff}^{(2)}_{\neq 0}(\mathcal{F}_2)$ such that

$$W_1|_B = W_2|_B \quad \text{and} \quad \mathrm{ewt}_2(W_2) \leq \mathrm{ewt}_1(W_1)\,.$$

Proof. 'if' / $\Leftarrow$: Assume that there is a correspondence for effects as outlined above. Consider an undetectable fault F_1 in D_1 with at most weight $w-1$.

As F_1 is undetectable, $\mathrm{eff}_1(F_1) \in \mathrm{Eff}^{(1)}_{\neq 0}(\mathcal{F}_1)$ by definition, and then by assumption there is a corresponding effect $W_2 \in \mathrm{Eff}^{(2)}_{\neq 0}(\mathcal{F}_2)$ with $\mathrm{eff}_1(F_1)|_B = W_2|_B$ and $\mathrm{ewt}_2(W_2) \leq \mathrm{ewt}_1(\mathrm{eff}_1(F_1))$. So there must exist some F_2 in D_2 such that $\mathrm{eff}_1(F_1)|_B = \mathrm{eff}_2(F_2)|_B$, which by Proposition 3.8 implies $D_1^{F_1} = D_2^{F_2}$.

Further by the above, W_2 also satisfies $\mathrm{ewt}_2(W_2) \leq \mathrm{ewt}_1(\mathrm{eff}_1(F_1))$. Then there must be at least one fault F_2' that is semantically equivalent to F_2, whose weight saturates the value $\mathrm{ewt}_2(W_2) = \mathrm{ewt}_2(\mathrm{eff}_2(F_2))$. So $D_1^{F_1} = D_2^{F_2} = D_2^{F_2'}$, and F_2' has equal or lower weight than F_1. As this was shown for an arbitrary F_1 with up to weight $w-1$, $D_1 \underset{w}{\hat{\leq}} D_2$ must hold.

'only if' / $\Rightarrow$: Assume that $D_1 \underset{w}{\hat{\leq}} D_2$ holds. Then take some $W_1 \in \mathrm{Eff}^{(1)}_{\neq 0}(\mathcal{F}_1)$ with $\mathrm{ewt}_1(W_1) < w$, which is saturated by some fault F_1 in D_1 with $\mathrm{eff}(F_1) = W_1$.

By assumption of $D_1 \underset{w}{\hat{\leq}} D_2$, there exists some F_2 in D_2 with $D_1^{F_1} = D_2^{F_2}$ and $\mathrm{wt}(F_2) < \mathrm{wt}(F_1)$. By Proposition 3.8, the former implies $\mathrm{eff}(F_2)|_B = \mathrm{eff}(F_1)|_B$, and more generally $\mathrm{eff}(F_2) \in \mathrm{Eff}^{(2)}_{\neq 0}(\mathcal{F}_2)$. But as F_1 saturates the weight $\mathrm{ewt}_1(W)$ and by Definition 3.11, $\mathrm{wt}(F_2) < \mathrm{wt}(F_1)$ implies $\mathrm{ewt}_2(W) < \mathrm{ewt}_1(W)$, which completes the claim.

□

Proposition 3.12 exactly captures that when determining if $D_1 \underset{w}{\hat{\leq}} D_2$ holds, we can simply disregard all detectable faults in D_1. Instead, we are able to focus on undetectable faults, and further only have to handle the minimum weight that a fault might produce in a given semantic equivalence class.

Recall that fault equivalence is the special case of symmetric ∞-fault boundedness. Then, we obtain a succinct description of fault equivalence directly from Proposition 3.12:

Corollary 3.13. Let $D_1 = D_2$ be two non-zero ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$, undetectable effects $\mathrm{Eff}^{(1)}_{\neq 0}, \mathrm{Eff}^{(2)}_{\neq 0}$ and corresponding effect weight functions $\mathrm{ewt}_1, \mathrm{ewt}_2$.

Then D_1 under $\mathcal{F}_1$ is fault-equivalent to D_2 under $\mathcal{F}_2$ if and only if:

- There is a bijection f that maps an undetectable effect $W_1 \in \mathrm{Eff}^{(1)}_{\neq 0}(\mathcal{F}_1)$ to an undetectable effect $f(W_1) = W_2 \in \mathrm{Eff}^{(2)}_{\neq 0}(\mathcal{F}_2)$ such that $W_1|_B = W_2|_B$, and vice versa, and

- $\mathrm{ewt}_1(W) = \mathrm{ewt}_2(f(W))$ for all $W \in \mathrm{Eff}^{(1)}_{\neq 0}(\mathcal{F}_1)$.

A similar claim for atomic faults has been used extensively in [RPK25] to show fault equivalence for simple transformations. Thus, we successfully generalised this correspondence to more involved transformations between noise models.

It will become useful to handle the special case where D_1, D_2 are *identical*, so only the noise model changes. In that case, a correspondence between boundary restrictions of effects is equivalent to a correspondence between undetectable effects themselves and we can simplify:

Corollary 3.14. Let D be a non-zero ZX diagram with two noise models $\mathcal{F}_1, \mathcal{F}_2$ and corresponding effect weight functions $\mathrm{ewt}_1, \mathrm{ewt}_2$.

Then $\mathcal{F}_1 \mathrel{\widehat{=}} \mathcal{F}_2$ holds if and only if:

- $\mathrm{Eff}_{\neq 0}(\mathcal{F}_1) = \mathrm{Eff}_{\neq 0}(\mathcal{F}_2)$, and
- $\mathrm{ewt}_1(W) = \mathrm{ewt}_2(W)$ for all $W \in \mathrm{Eff}_{\neq 0}(\mathcal{F}_1)$.

3.3 Axioms

We start developing the axioms by proving some algebraic results majorly facilitated by Corollary 3.13. In particular, these results show us how to do the following fault-equivalently:

- Remove / Introduce redundant faults, under the notion of redundancy introduced in Section 3.1.
- Remove / Introduce composite faults.
- Translate between all faults detectable by some region generator R and all of their pairwise composites, which are undetectable by R.

We will show these in quick successive order:

Proposition 3.15 (Merging redundant atomic faults)**.** Let D be a ZX diagram with a noise model $\mathcal{F}$ and let eff be its corresponding effect function. Then for all $F_1, F_2 \in \mathcal{F}$, if $\mathrm{eff}(F_1) = \mathrm{eff}(F_2)$ and $\mathrm{awt}(F_1) \leq \mathrm{awt}(F_2)$, we can fault-equivalently remove F_2 from $\mathcal{F}$.

Proof. Let $\mathcal{F}'$ be the noise model obtained by removing F_2 from $\mathcal{F}$. Further, let $\mathrm{ewt}, \mathrm{ewt}'$ be the effect weight functions of the respective noise models $\mathcal{F}, \mathcal{F}'$. Since F_1 and F_2 have the same effect, we can directly deduce $\mathrm{Eff}(\mathcal{F}) = \mathrm{Eff}(\mathcal{F}')$, so inherently $\mathrm{Eff}_{\neq 0}(\mathcal{F}) = \mathrm{Eff}_{\neq 0}(\mathcal{F}')$ as well.

Now assume for a contradiction that there exists an effect W with $\mathrm{ewt}(W) \neq \mathrm{ewt}'(W)$, and let F_W be the composite fault such that $\mathrm{eff}(F_W) = W$. Since we only removed an atomic fault, it must be that F_2 was in a composing set $\widetilde{F}$ for F_W. F_1 and F_2 have the same effect by assumption, so there must be another composing set $\widetilde{F}'$ for F_W such that

$$\widetilde{F}' = \{F_1\} \cup (\widetilde{F} - \{F_2\}),$$

i.e. the two composing sets only differ by only F_1 or F_2. The fault F_1 is still present in $\mathcal{F}'$, and since it has at most the same weight as F_2 the composing set $\widetilde{F}'$ must have an equal or lower weight sum than $\widetilde{F}$. Thus, we must still have $\mathrm{ewt}(W) = \mathrm{ewt}'(W)$, yielding a contradiction.

So $\mathrm{Eff}_{\neq 0}(\mathcal{F}) = \mathrm{Eff}_{\neq 0}(\mathcal{F}')$ and $\mathrm{ewt} = \mathrm{ewt}'$ hold which implies fault equivalence by Corollary 3.13. □

Proposition 3.16 (Combining atomic faults)**.** Let D be a ZX diagram with a noise model $\mathcal{F}$ and effects $\mathrm{Eff}(\mathcal{F})$. Then for all $F_1, F_2 \in \mathcal{F}$, we can fault-equivalently introduce their product F_1F_2 as an atomic fault, assigning it the weight

$$\mathrm{awt}(F_1F_2) = \mathrm{awt}(F_1) + \mathrm{awt}(F_2).$$

Proof. Let $\mathcal{F}'$ be the noise model obtained by adding F_1F_2 to $\mathcal{F}$. Further, let ewt and ewt$'$ be obtained from $\mathcal{F}, \mathcal{F}'$ respectively.

Since F_1F_2 was already contained in $\langle\mathcal{F}\rangle$, we already had $\mathrm{eff}(F_1F_2) \in \mathrm{Eff}(\mathcal{F})$. Thus, we have that $\mathrm{Eff}(\mathcal{F}) = \mathrm{Eff}(\mathcal{F}')$.

Under $\mathcal{F}$, $\{F_1, F_2\}$ is a composing set of F_1F_2, which has a weight sum of exactly $\mathrm{awt}(F_1) + \mathrm{awt}(F_2)$. Adding another composing set $\{F_1F_2\}$ with the exact weight sum does not change the induced weight of any faults that are composites of F_1F_2. Then we directly have $\mathrm{ewt} = \mathrm{ewt}'$ and thus fault equivalence by Corollary 3.13. □

Proposition 3.17 (Removing detectable atomic faults)**.** Let D be a ZX diagram with a noise model $\mathcal{F}$ and detecting region group $\langle R\rangle$. For a detecting region generator $r \in R$, let $\bar{F}_r$ be the set of atomic faults detectable by r.

Further, let there be some $F \in \bar{F}_r$ such that

$$\forall F' \in \bar{F}_r \setminus \{F\}: \qquad FF' \in \mathcal{F}' \quad \text{and} \quad \mathrm{awt}'(FF') = \mathrm{awt}(F) + \mathrm{awt}(F')$$

holds. Then we can fault-equivalently obtain a new noise model $\mathcal{F}'$ by removing F from $\mathcal{F}$.

Proof. Assume for a contradiction that $\mathrm{Eff}_{\neq 0}(\mathcal{F}) \neq \mathrm{Eff}_{\neq 0}(\mathcal{F}')$, so there is some $W \in \mathrm{Eff}_{\neq 0}(\mathcal{F})$ with $W \notin \mathrm{Eff}_{\neq 0}(\mathcal{F}')$ since we only removed an atomic fault. There must be some composite fault F_{comp} in $\langle\mathcal{F}\rangle$ such that $\mathrm{eff}(F_{comp}) = W$.

We only removed F, so F must be in a composing set for F_{comp}. Furthermore, since W is an undetectable effect, F_{comp} must be undetectable, i.e. it is not allowed to flip r. That requires that, to yield F_{comp}, F was at least composed with some F' that also flips r. Then $\{F, F'\}$ is part of some composing set for F_{comp}, and so is the product FF'

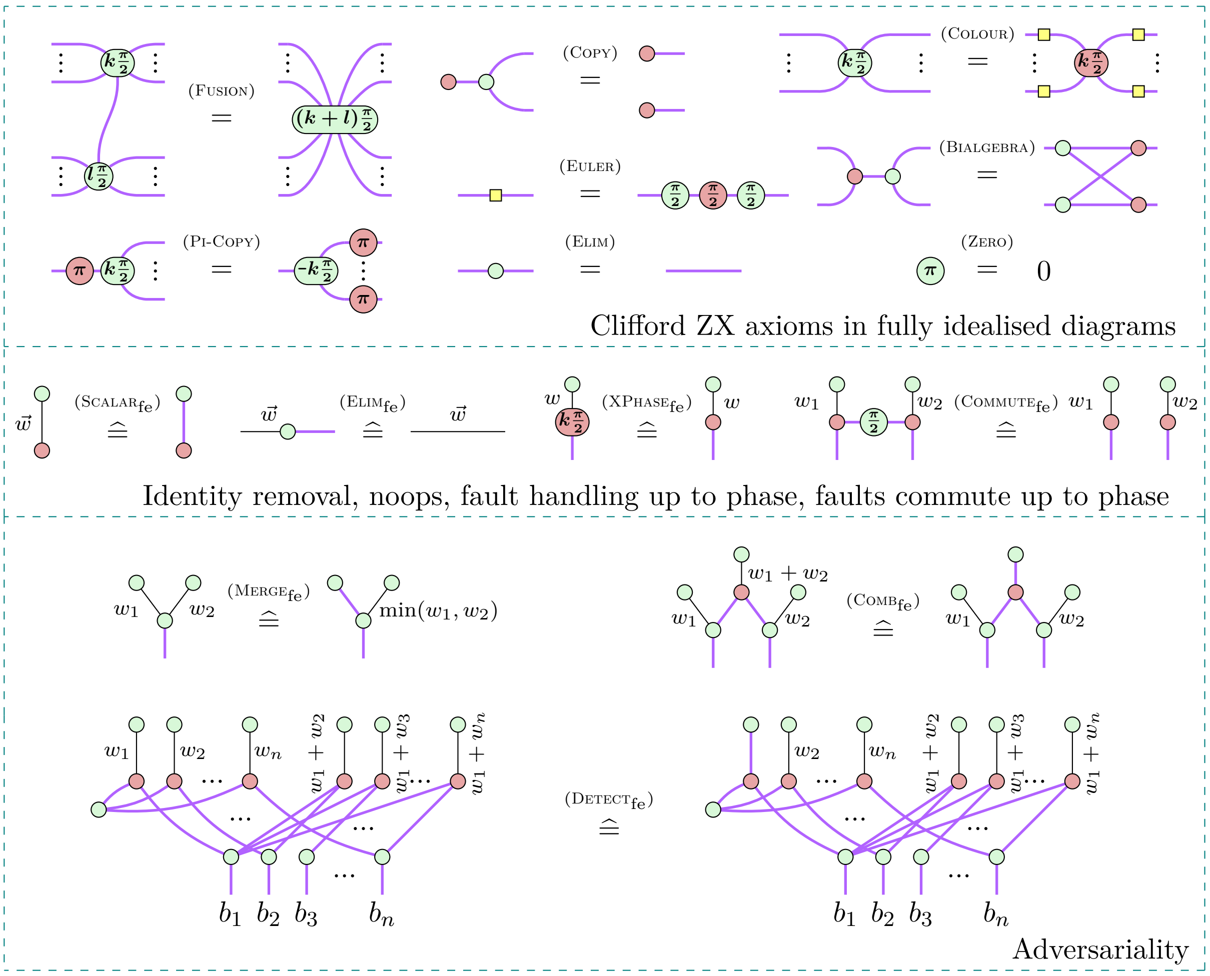


Figure 2: The set of axioms required for a ZX calculus that is complete for fault equivalence, divided into three groups: Fully idealised Clifford axioms, intuitive handling of faults and adversarial reasoning.

which is included in the atomic faults for $\mathcal{F}'$ by assumption. But then F_{comp} is also in $\langle \mathcal{F}' \rangle$, and thus $\text{eff}(F_{comp}) = W \in \text{Eff}_{\neq 0}(\mathcal{F}')$ which completes the contradiction.

Beyond $\text{Eff}_{\neq 0}(\mathcal{F}) = \text{Eff}_{\neq 0}(\mathcal{F}')$, the proof of F_{comp} having equal effect weight in both noise models is identical to the proof in Proposition 3.16. Thus, since F_{comp} was arbitrary, we can directly conclude that both noise models agree on the weights of all undetectable effects, so the rewrite is fault-equivalent by Corollary 3.13. □

Note that this result directly implies that if $|\bar{F}_r| = 1$, we can simply remove the single fault in the set from the noise model. We will see a diagrammatic example of this in Section 6.3. Additionally, we highlight that the fault equivalence in Propositions 3.15 to 3.17 implies symmetry: Every time we add / remove faults, we could just as easily do the inverse.

We are now ready to state the fault equivalence axioms that we will show form a ZX calculus that is complete for fault equivalence of Clifford ZX diagrams. The axioms are presented in Fig. 2. We divide them into three groups, will be particularly useful at different stages of the completeness proof:

1. The regular Clifford ZX axioms on fully idealised diagrams.
2. Axioms that enable intuitive handling of faults and diagrams that have an annotated noise model, which we will generalise to multi-edge scenarios in Section 4.
3. Axioms that enable adversarial reasoning through Propositions 3.16 to 3.17, applicable in Section 6.

Finally, we state the main theorem of this work:

Theorem 3.18 (Completeness for fault equivalence)**.** The ZX calculus given through axiomatisation in Fig. 2 is complete for fault equivalence.

Proof. In Section 6. □

At this point, we must note the following about the axioms from Fig. 2:

- The axioms use the weight annotation introduced in Section 2.4, where we recall to use a single weight w to abbreviate $(-, -, w)$.
- Some of the axioms are direct applications of the algebraic results we just derived, namely ($\textsc{Merge}_{\text{fe}}$) applies Proposition 3.15, ($\textsc{Comb}_{\text{fe}}$) applies Proposition 3.16 and ($\textsc{Detect}_{\text{fe}}$) applies Proposition 3.17.
- While we show these axioms to be complete in Section 6, they differ from the regular ZX calculus axioms in that using them explicitly on a regular basis is cumbersome. Instead, we will use them in Section 4.4 to derive diagrammatic results that are closer to Propositions 3.15 to 3.17 and can be applied just as naturally.
- We will use all of these axioms while proving completeness, although we do not formally prove their necessity. Thus, we can *not* confidently state that this set of axioms is minimal.

3.4 Soundness

If we were able to derive a fault equivalence relationship between two diagrams that does not actually exist between them, i.e. becoming *unsound*, we would have to be careful using our set of rewrites. This would eliminate a major benefit of the ZX calculus and similar graphical calculi: They allow unguided exploration to alternative diagrams while guaranteeing that the underlying interpretation stays the same through soundness. It is thus vital for the ZX calculus itself to guarantee soundness for semantic equivalence, which it does [CD11]. Now it is our turn to argue that the axioms we presented are sound for fault equivalence and thus usable in an explorative sense, so we reserve this section entirely for this argument.

Proposition 3.19. The axioms in Fig. 2 are sound w.r.t. fault equivalence under edge flip noise.

Proof. Clifford Rules For each such rule, the LHS and RHS are fully idealised by definition. Applying Proposition 2.20 both ways yields symmetric fault boundedness, which is fault equivalence.

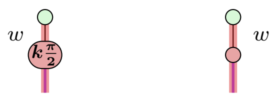
(Scalar$_{\mathsf{fe}}$) The diagram has no Pauli webs, so $\mathrm{Eff}(\mathcal{F}_{\mathsf{LHS}}) = \mathrm{Eff}(\mathcal{F}_{\mathsf{RHS}}) = \{\emptyset\}$. But then for both sides, $\emptyset$ is the effect of the trivial fault, so

$$\mathrm{ewt}_{\mathsf{LHS}}(\emptyset) = \mathrm{ewt}_{\mathsf{RHS}}(\emptyset) = \mathrm{wt}(I_{\mathsf{LHS}}) = \mathrm{wt}(I_{\mathsf{RHS}}) = 0$$

Thus, $\mathrm{ewt}_{\mathsf{LHS}} = \mathrm{ewt}_{\mathsf{RHS}}$, and by Corollary 3.13 the sides are fault-equivalent.

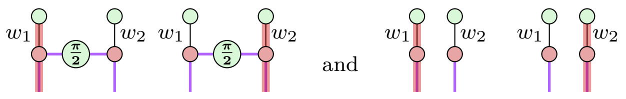
(Elim$_{\mathsf{fe}}$) By (Elim), the rewrite holds semantically, so LHS = RHS is given. The diagram has two possible Pauli webs, one green and one red, and each possible edge flip flips the same webs on both sides. Thus, similar to (Scalar$_{\mathsf{fe}}$), the rewrite is fault-equivalent.

(XPhase$_{\mathsf{fe}}$) By (Pi-Copy), the rewrite holds semantically, so LHS = RHS is given. Further, the diagrams both have exactly one Pauli web:



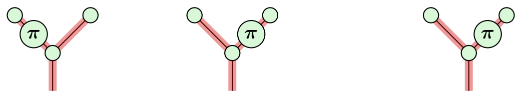
The same single-edge fault flips this web on both sides, so similar to (Scalar$_{\mathsf{fe}}$) both sides are fault-equivalent.

(Commute$_{\mathsf{fe}}$) Without regards to faults, the rewrite holds up to a scalar using the (Bialgebra) rule, so the two diagrams are semantically equivalent. Furthermore, the Pauli webs for the diagrams are generated by two generators each, which correspond to each other w.r.t. their boundary restriction:



Both atomic faults flip exactly one of these webs on either side, so the set of effects of the diagram remains the same, and thus as before the rewrite is fault-equivalent.

(Merge$_{\mathsf{fe}}$) By application of (OCM), w.l.o.g. we can assume $w_2 \geq w_1$. Then we observe that the diagram has only one Pauli web, and on both sides all possible atomic faults flip it, so they all have the same effect:



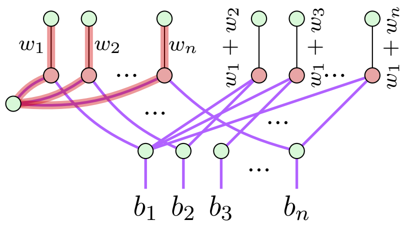
flipped with w_1 flipped with w_2 flipped with $\min(w_1, w_2)$

Then, this axiom is simply an application of Proposition 3.15, where the fault with weight w_1 is removed from the noise model, so the LHS and RHS are fault-equivalent.

(Comb$_{\mathsf{fe}}$) Similar to the previous case, it is easy to see that this is an application of Proposition 3.16.

(Detect$_{\mathsf{fe}}$) We will refer to the Z-flips weighted as $w_1, \ldots, w_n$ with $F_1, \ldots, F_n$, and to the Z-flips with weight $w_1 + w_j$ as F_{1j}.

We note that the diagram contains a detecting region that encapsulates all $F_1, \ldots, F_n$:



Further, we realise that for F_1, all composites F_{1j} formed with other faults F_j are contained in the diagram. Via the connectivity to the boundary edges, F_1F_j and F_{1j} have the same effect. Finally, we see that this is just an instance of Proposition 3.17 and that the rewrite is thus fault-equivalent.

□

3.5 Outlook: Diagrammatic Extraction of Effect Weights

We briefly sketch the contents of the upcoming work up to and including Section 6. In Section 4 we develop a diagrammatic model for arbitrary atomic faults using only unweighted edge flip noise, thus proving that unweighted edge flip noise is universal, and reason how these models change as they are moved throughout the diagram.

We then formulate and formalise a goal in Section 5: We want to diagrammatically separate the description of faults from the original diagram, i.e. from its fault-free semantics. This is directly possible for undetectable faults, as by Proposition 2.32 we can find equivalents for each individual flipped web from a faults effect on the boundary. However, for detectable faults, we need to develop additional tools. We thus systematically provide a few locations inside the diagram that are sufficient for us to find such equivalents for each fault, which we proceed to call 'signatures'. Signatures are a precursor to effects, in the sense that the equivalence relationship between them is strictly stronger than effect equivalence. We then implement and diagrammatically obtain signatures in Section 5.3.

Finally in Section 6, we show that the locations we obtained for detecting regions are special in that we can disconnect them from the original diagram entirely, achieving the desired separation of a diagrams semantics and the diagrammatic description of its noise model. These locations, now only part of the noise model, are subsequently eliminated in a fault-equivalent manner. For all remaining signatures, we identify the effect that they correspond to and find a normalisation method that reduces every signature in a given effect equivalence class to one canonical representative. As we retain the minimal weight for each class, this diagrammatically enumerates the entire function ewt in a normal form, whereafter Corollary 3.13 guarantees fault equivalence and thus completeness. We note that this normal form can have an exponential size and thus take exponentially many rewrite steps to generate; however, it is already known that checking fault equivalence in general is NP-hard [RPK25, Thm. 3.10].

To visualise, extracting the effect weighting function of a diagram D under noise model $\mathcal{F}$ unfolds as:

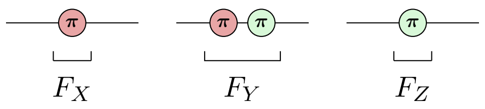
$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{Model}\\\text{atomic faults}\\\text{in diagram}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Obtain}\\\text{signatures}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Separate}\\\text{diagram } D\\\text{and noise } \mathcal{F}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Enumerate}\\\text{all undetectable}\\\text{faults}\end{array}} \arrow[r] & \boxed{\text{Normalise}} \\
\text{Section 4} & \text{Section 5} & \text{Sections 6.1 and 6.2} & \text{Section 6.3} & \text{Section 6.4}
\end{tikzcd}$$

Every modification to the pair $(D, \mathcal{F})$ in this plan is possible (although cumbersome) diagrammatically through the axioms introduced in Section 3.3, which we show constructively.

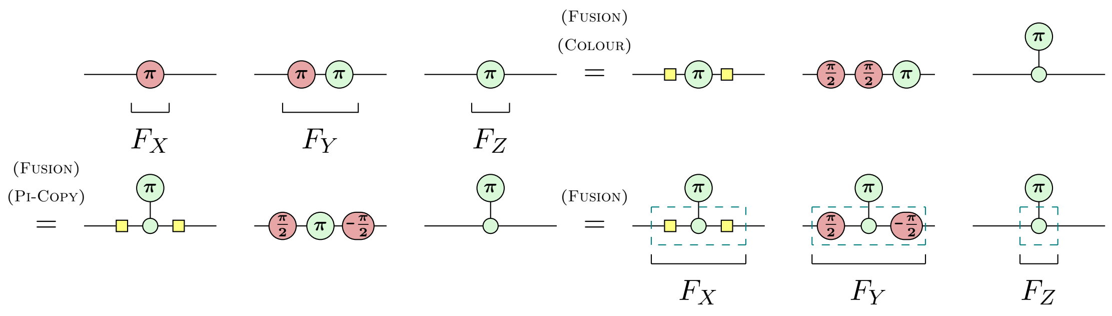
4 Fault Gadgets

We now tackle the proposed universality of edge flip noise through a construction that allows explicitly modeling faults from arbitrary weighted noise models using edge flip noise. This construction is known as *fault gadgets*, introduced in [RPK25, Def. 6.3] as a method to diagrammatically track faults operating on multiple fault locations in circuits. We will be introducing them similarly, and prove universality of unweighted edge flip noise for all weighted multi-edge noise channels in Theorem 4.6, as a generalisation of [RPK25, Prop. 6.4]. However, fault gadgets will not remain static trackers as was their initial purpose: Instead, we will use them as a dynamic reasoning tool for faults in the proof of Theorem 3.18. Beyond their introduction, this section provides notion of 'moving' fault gadgets throughout a diagram, and implements important algebraic results from Section 3 diagrammatically for multi-edge noise.

We start deriving models for the simplest of faults, i.e. the three non-trivial edge flips which are instantiated on a diagram as:

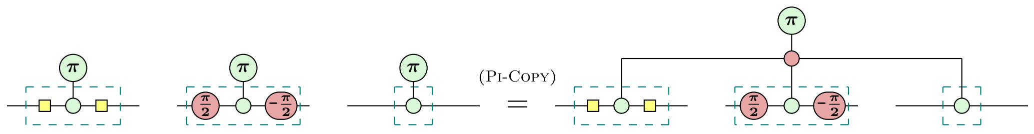


We then apply some transformations to find a common structure:

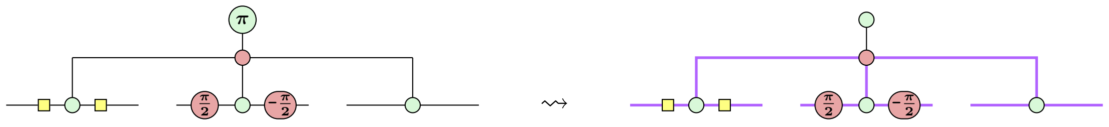


Note how each dashed box in the last diagram is controlled by a green π-phase on top of it. These controlled units are known as *Pauli boxes*, since by construction they exactly provide one of the three Pauli rotations to a diagram edge.

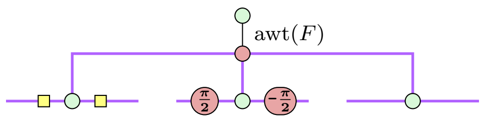
To obtain models for more complex faults, we can control multiple Pauli boxes at once with one π-phase:



All faults that we handled so far *did already happen*. We now move to faults that *may happen*, in a way that only utilises single edge flips, even for multi-edge faults. Crucially, we have to make sure that we only allow the *exact* multi-edge fault that we want to obtain. We do this by idealising all edges where a single edge flip would not have the effect that we want it to have:

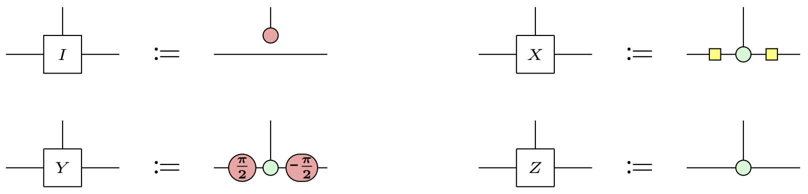


In this new diagram, there is only a single edge that is not idealised where as of now we allow all three edge flips to occur. But there is no use to allowing all three edge flips: X-flips can be removed from the edge via (Pi-Copy), and similarly Y-flips have the same effect as Z-flips, so to determine whether the Pauli boxes should be supplied with a green π-phase or not, it suffices to use Z-flips. Thus, when modeling an atomic fault F with weight $\mathrm{awt}(F)$, we only use that annotation:



So we obtain a construction that can create multi-edge faults with a single Z-flip, while disallowing other unwanted faults. This construction is exactly a *fault gadget*, and we generalised it to be associated with an arbitrary weight.

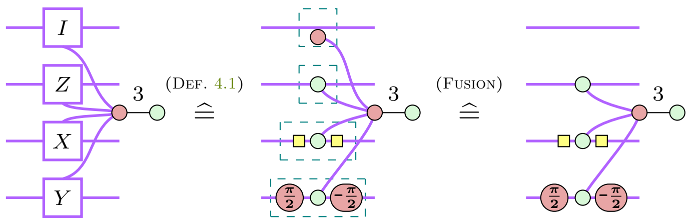
Definition 4.1 (Pauli boxes)**.** The following four diagrams are the *Pauli boxes*, associated with the annotated Pauli rotation:



Similar to the derivation above, we use fault-free Pauli boxes:

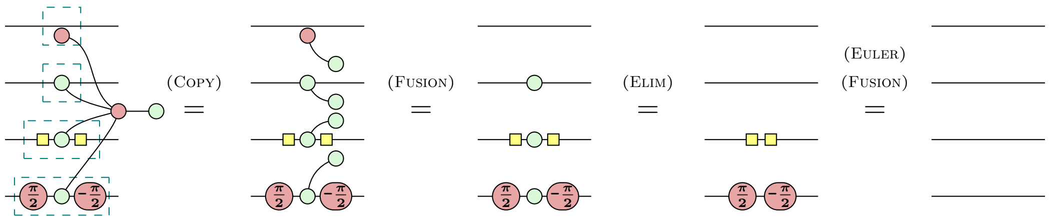
Definition 4.2. Let D be a ZX diagram under a weighted noise model $\mathcal{F} = (A, \mathrm{awt})$, and let $F \in A$. A *fault gadget* for F is constructed by including fault-free Pauli boxes, called *targets*, of the types and on the edges indicated by F. These boxes are connected via idealised edges to a red spider. Further, the red spider is connected via a regular edge to a green spider, called the *spawning edge* of the fault gadget. The spawning edge is annotated with $\mathrm{awt}(F)$.

For example, the following diagram consists of four standalone wires, i.e. it implements a four-qubit identity. Suppose we now consider a noise model for this diagram with an atomic fault $I \otimes Z \otimes X \otimes Y$ that has atomic weight 3. The fault gadget corresponding this fault is:



The identity Pauli box is inherently disconnected, so its one-legged spider can be merged into the red distribution spider of the gadget. We choose not to draw the identity Pauli box at all in the remainder of this work.

Finally, we note that Pauli boxes only implement their corresponding rotation when provided with a green π-phase spider. When the green spider has phase 0, all boxes implement the identity rotation. This remains true when generalising Pauli boxes to multiple edges as fault gadgets:



We can thus directly formalise:

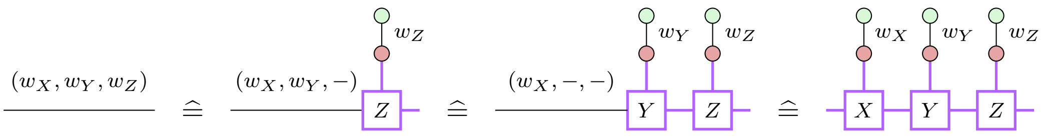
Proposition 4.3. Let D be a ZX diagram and F be a fault for D. If D' is the diagram obtained by adding a fault gadget for F to D, then $D' = D$, i.e. adding fault gadgets does not change the linear map that a diagram evaluates to.

Proof. Via repeated application of (Copy), (Fusion), (Euler) and (Elim) as above. □

Note that from now on, we will not label applications of the (fault-free) Clifford rules anymore, unless required for clarity. Thus, all subsequent derivations assume a basic intuition for reading and understanding these rule applications. Additionally, if we want to apply a fault-free Clifford rule on spiders that have auxiliary faulty legs not included in the rewrite, we can always use (Elim$_{\text{fe}}$) to move the faulty legs further from the spider, then apply the rewrite, and use (Elim$_{\text{fe}}$) to move them back.

4.1 Universality of Unweighted Edge Flip Noise

We now have everything at our disposal to prove that unweighted edge flip noise can be used to model arbitrarily weighted multi-edge noise. Shortly returning to edge flip noise, we can visualise through edge annotations how fault gadgets 'shift' the weight of a fault from the actual diagram edge to the fault gadgets spawning edge. Indeed, we can use this to progressively idealise an edge:



Note that we only allow Z edge flips to occur on the fault gadgets; the actual rotation applied to the edge is determined by the type of the Pauli box that is in use.

Clearly this concept can be applied to multi-edge faults as well, explicitly modeling arbitrary atomic faults on the diagram using only edge flip noise. This brings us a step closer to the claim of universality:

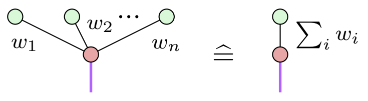
Proposition 4.4. Let D be a ZX diagram and $\mathcal{F}$ be a weighted noise model for D. Then there exists a ZX diagram D' along with a weighted *edge flip* noise model $\mathcal{F}'$ such that $D = D'$ and D under $\mathcal{F}$ is fault-equivalent to D' under $\mathcal{F}'$.

Proof. Let $\mathcal{F} = (A, \mathrm{awt})$. To obtain D' and $\mathcal{F}'$, first idealise all edges within D, then add a fault gadget for each atomic fault $F \in \mathcal{F}$ with a weight annotation equal to $\mathrm{awt}(F)$ on the spawning edge.

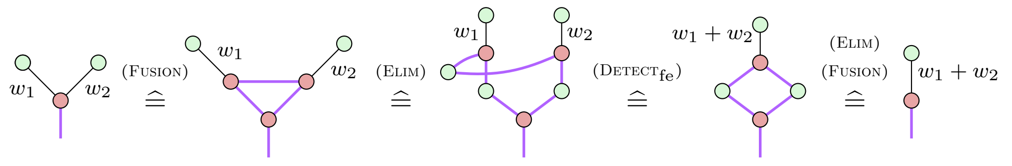
Fault gadgets do not influence the linear map by Proposition 4.3, so we directly have $D = D'$. Additionally, every fault gadget F has exactly one atomic fault Z_F on its spawning edge that activates the Pauli boxes, and there are no additional non-idealised edges in the diagram. Similarly, each atomic fault Z_F in D' directly corresponds to an atomic fault F in D, yielding $D \mathrel{\widehat{=}} D'$. □

We now have a fully idealised diagram save for the spawning edges of the fault gadgets, which are annotated with the atomic weight of the atomic fault they represent. However, we realise that this annotation can be viewed as syntactic sugar, using a simple addition rule that holds in fault gadget heads:

Lemma 4.5 (Head addition)**.** For $n \geq 1$, it holds that

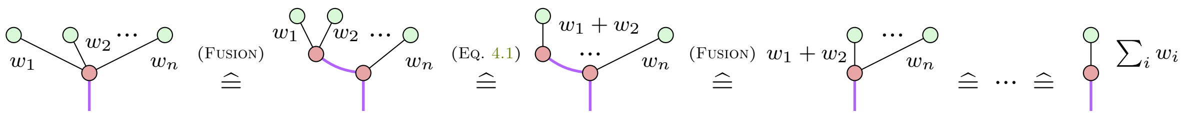


Proof. The case $n = 1$ is trivial. We show the claim for the case $n = 2$ first:



(4.1)

The general case $n > 2$ is then shown via repeated application of the case $n = 2$:



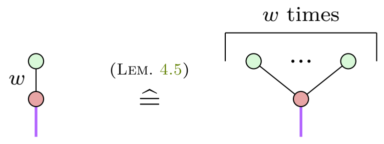
□

We can now prove full universality of edge flip noise without using weights:

Theorem 4.6 (Universality of unweighted edge flip noise)**.** Let D be a ZX diagram and $\mathcal{F}$ be a weighted noise model for D. Then there exists a ZX diagram D' along with an *unweighted edge flip* noise model $\mathcal{F}'$ such that

1. $D = D'$ and
2. D under $\mathcal{F}$ is fault-equivalent to D' under $\mathcal{F}'$.

Proof. Obtain D' as in Proposition 4.4, then expand the weights in the fault gadget via Lemma 4.5:

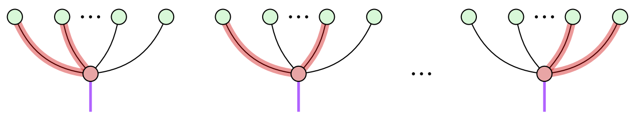


□

A similar result was already presented in [RPK25, Prop. 6.4] as finding equivalent ZX diagrams for circuits with noise models. Although the proof is equally constructive, Theorem 4.6 is expressed for all ZX diagrams and for arbitrarily weighted atomic faults.

Remark 4.7. Using Lemma 4.5 enables a different view on the notion of 'weight'. Previously in Definition 2.11, we introduced weight as a notion that is disconnected from the notion of a weight for a multi-qubit Pauli operator (which is how many non-trivial single-qubit Paulis are contained in the operator). When considering single-qubit Pauli operators as being those that occur on their own (i.e. edge flip noise), then the weight of a fault F is the minimum number of atomic faults that must occur for the composite F to occur [RPK25].

We now see that all such notions of weight coincide when using fault gadgets without syntactic sugar: For a fault gadget with w heads, we need w single-edge Z-flips to occur before there is an undetectable Z flip propagating to the fault gadget. This is because the heads form pairwise detecting regions that can detect any number of faults that happens on less than w edges:



So when modeling an atomic fault F with weight $\mathrm{awt}(F)$, we need at least $\mathrm{awt}(F)$ single-qubit faults to occur before F occurs.

Although we would now be able to reason even without annotations, only using regular and idealised edges, we choose to not use Lemma 4.5 in any future proof. Instead, we opt to using the syntactic sugar that annotations provide in fault gadget spawning edges, as this allows us to more concisely and clearly reason about faults. Every proof could also be performed without these annotations, but would be significantly more cumbersome.

We conclude that fault gadgets provide us with a way to explicitly encode the noise model of a diagram into the diagram itself. Simply annotating the ZX diagram is only unambiguous for edge flip noise, whereas additionally using fault gadgets enables clearer handling of multi-edge faults. Furthermore, treating the noise model as part of the diagram has more refined uses than just display. In fact, fully encoding the noise model into the diagram is so useful to us that we provide its own notion:

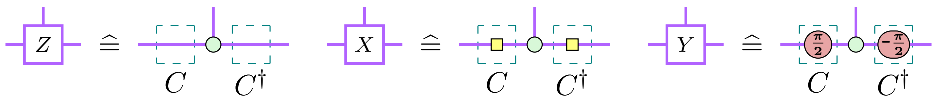
Definition 4.8 (GI-form)**.** Let D be a ZX diagram with a noise model $\mathcal{F}$, and let $D_{\mathcal{F}}$ be the diagram that incorporates $\mathcal{F}$ into D using fault gadgets. $D_{\mathcal{F}}$ is in *gadget idealised form* ('GI-form' for short), if the only non-idealised edges in $D_{\mathcal{F}}$ are the spawning edges of fault gadgets.

In the remainder of this work, we will use $D_{\mathcal{F}}$ to refer to the diagram that implements the noise model $\mathcal{F}$ on D using fault gadgets as above. If $D_{\mathcal{F}}$ is obtained as in Theorem 4.6, it is in GI-form by construction, which we will assume to be the case in the remainder of this work.

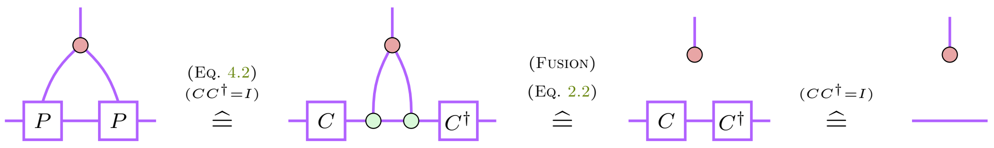
4.2 Basic Properties

In Proposition 4.3, we have already seen that fault gadgets do not change the linear map the given diagram evaluates to. Additionally, we informally reasoned that the only faults that have some effect in gadgets are the Z components of edge flips on the spawning edge. Now that we can implement the noise model of entire diagrams using fault gadgets, we can use fault-free versions of the Clifford rules to derive additional properties.

To start, we see that every non-trivial Pauli box consists of a Clifford diagram C, a green spider and the adjoint $C^\dagger$ of C:

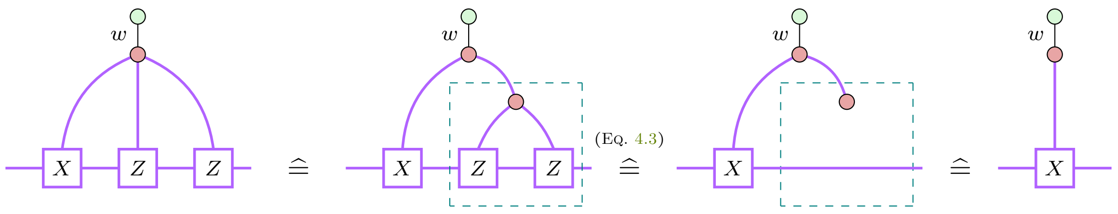
 (4.2)

Although these equalities also hold semantically for non-fault-free Pauli boxes, fully idealising all diagrams helpfully provides fault equivalence. Using these equalities, it becomes easy to see that within one fault gadget, targets on the same edge are self-inverse:

 (4.3)

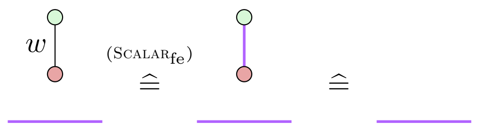
This is expected: Introducing the same edge flip twice on an edge should flip the same webs as not doing anything on that edge.

We note that in Eq. (4.3), we chose to fully omit the part of the fault gadget that is irrelevant to the targets to reduce clutter. This is always allowed; we can bundle any number of targets and unfuse a separate red spider fault-equivalently. For example, we can apply the self-inversion of targets in some larger context:



As long as one proves the desired property with just this dangling red spider, omitting the context of the fault gadget is valid. As a side effect, this makes such properties usable outside of fault gadgets.

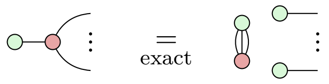
We now recall the rule (Scalar$_{\text{fe}}$) from the completeness axioms in Fig. 2. The scalar with annotation w may be seen as a fault gadget that has no targets. Using (Scalar$_{\text{fe}}$) and finally eliminating the scalar using the fault-free Clifford rules, we can fully eliminate the fault gadget:



(4.4)

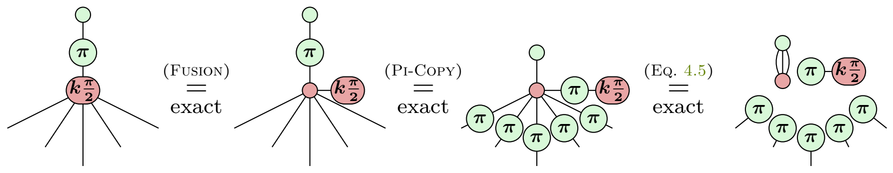
A fault gadget with no targets represents exactly the trivial fault I. We should not have this in our set of atomic faults, so removing such gadgets with Eq. (4.4) shows that fault gadgets are fully consistent with noise models as defined in Definition 2.11.

We can also show that gadgets allow us to diagrammatically treat faults up to global phase. To motivate this, we must revisit yet another scalar accurate rewrite, in this case of the (Copy) rule which we obtain from [KvdW24]:



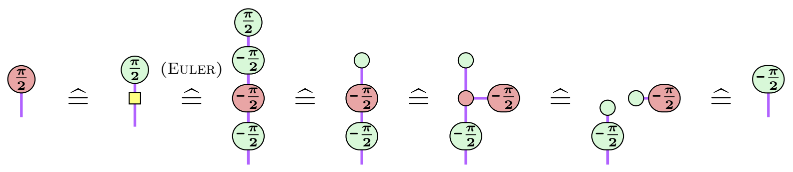
(4.5)

Then, additionally using the scalar accurate (Pi-Copy) from Eq. (2.3), we consider what happens when a gadgets red spider obtains a phase and a Z flip is placed on its spawning edge:



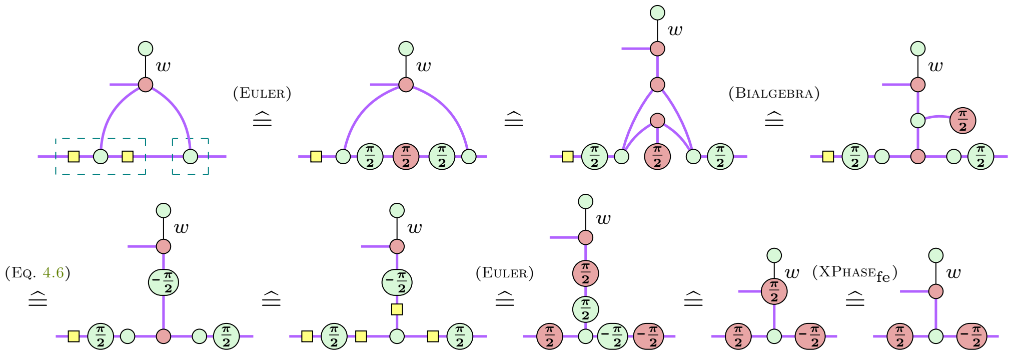
So placing a $k\frac{\pi}{2}$ phase on the red spider results in two free floating scalar diagrams apart from the fault that the gadget represents. It can be shown that the first scalar diagram is equal to $\frac{1}{\sqrt{2}}$, while the second is equal to $\sqrt{2}e^{ik\frac{\pi}{2}}$. Thus, using $k \in 0, \ldots, 3$ the total phase incurred when a fault gadget's atomic fault spawns is i^k, which means we can visualise the phase of a fault in its gadget. By (XPhase$_{\text{fe}}$), we can simply remove this phase from the gadget and thus consider all faults that only differ in phase diagrammatically equal. This makes fault gadgets consistent with us treating faults up to a scalar algebraically.

We want to immediately employ this technique to diagrammatically show $Y \propto XZ$. However, we need a small intermediate result, i.e. how we can transform a one-legged spider of one color into the other [BPW17]:



(4.6)

We can then show $Y \propto XZ$:



(4.7)

We might now encounter cases where multiple targets of different types reside on the same edge. This could prevent us from directly using Eq. (4.3) to eliminate two targets if those are not direct neighbours. However, we find that individual targets attached to gadgets commute, regardless of how they are connected to gadgets:

Proposition 4.9 (Gadgets commute)**.** Let $D_{\mathcal{F}}$ be a gadget-idealised ZX diagram based on the underlying diagram D. For a single edge e in D, targets of fault gadgets from $D_{\mathcal{F}}$ on e may be arbitrarily and fault-equivalently rearranged.

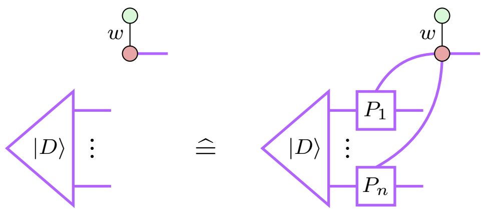
Proof. In Appendix A. □

So fault gadgets already have some limited room to move, by removing any restrictions on the order in which targets can reside on an edge. However, we can equip them with even more flexibility, showing how they can change to move *through* a diagram.

4.3 Moving Fault Gadgets

Indeed, we are missing just one result to show how fault gadgets move:

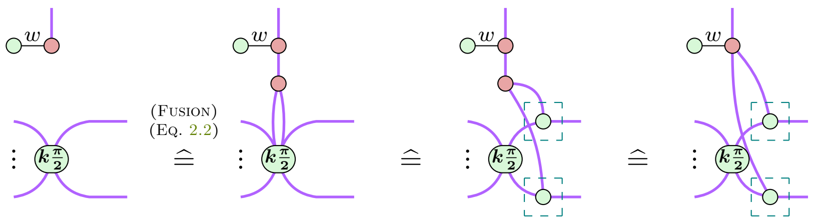
Proposition 4.10. Let D be a fully idealised ZX diagram, and let $|D\rangle$ be the diagram obtained by bending around the input wires of D. Then if $|D\rangle$ is stabilised by a Pauli operator $P_1 \otimes \cdots \otimes P_n$, it holds that



(Stab_fe)

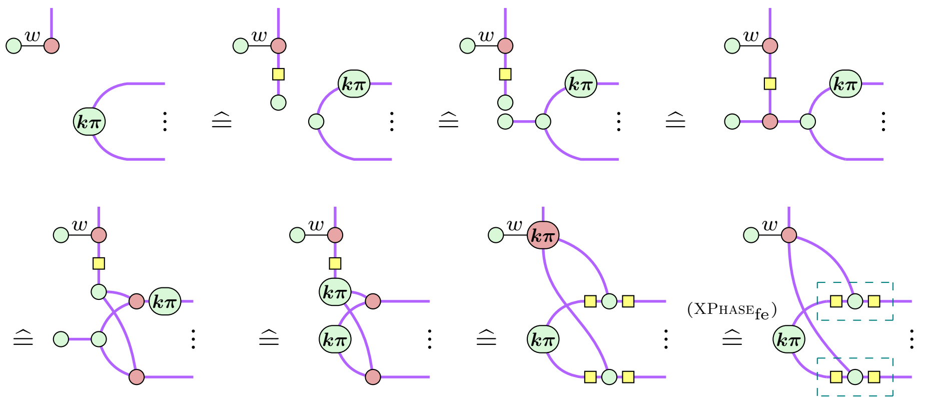
Proof. We start by proving the claim for all individual spiders that could be contained in D. By the application of (Colour), it suffices to prove for $k\frac{\pi}{2}$-phase Z-spiders. We observe that the Pauli operators stabilising a given spider are exactly those that form a valid Pauli web around the spider, so we have to be able to introduce targets according to all possible webs of the spiders.

On any spider, we can introduce targets of the spiders own colour two at a time:



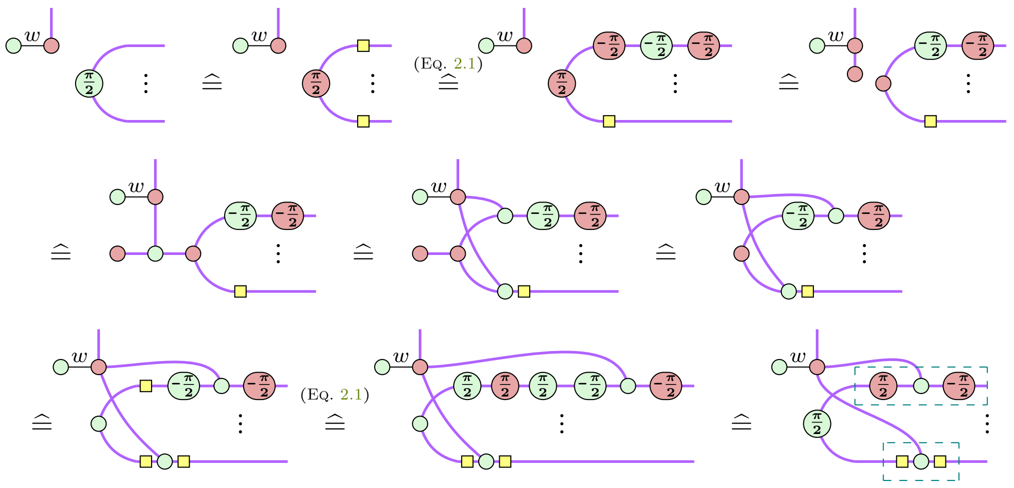
(4.8)

Then for spiders with a phase of $\{0, \pi\}$, we can introduce targets of the opposite colour on all legs:



This already suffices to prove the claim for all Pauli webs around $\{0, \pi\}$-phase spiders, and for $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$-phase spiders where the web is not of the opposite colour.

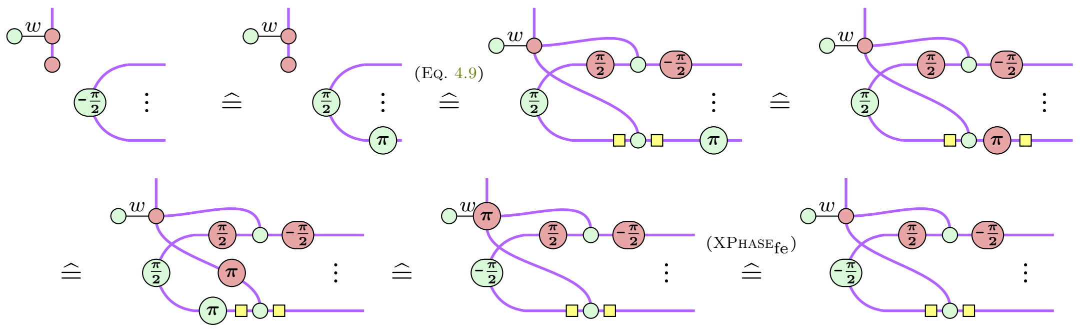
The cases of $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$-phase spiders with a web of the opposing colour remain. These webs require an odd number of legs to be highlighted in the spiders own colour. We first observe for $\frac{\pi}{2}$-phase spiders:



(4.9)

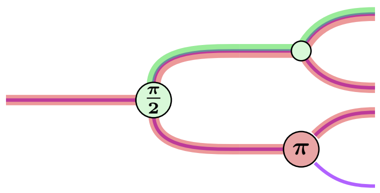
So together with Eq. (4.8) the claim holds for the case $\frac{\pi}{2}$.

Then the case of $(-\frac{\pi}{2})$-phase spiders is simply a variation of Eq. (4.9) using an additional π-phase:

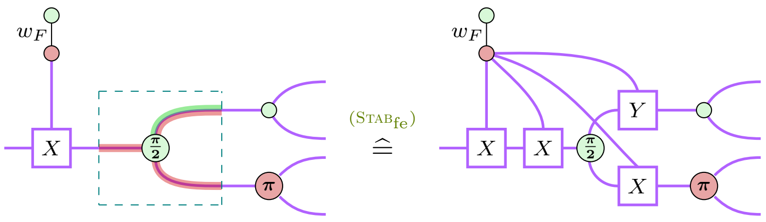


So the claim thus holds for all webs of individual spiders. But then exactly as Pauli webs compose in larger diagrams we can compose the introduction of these targets. Pauli webs must be consistent / compatible on all legs, so any edges internal in D must feature each target either zero or two times, and any boundary edges feature each target zero or one times. This allows applying Eq. (4.3) to cancel out any targets on internal edges in D, leaving only targets on boundary edges. Every Pauli web restricted to the boundary provides a possibly trivial stabiliser so this suffices to prove the original claim. □

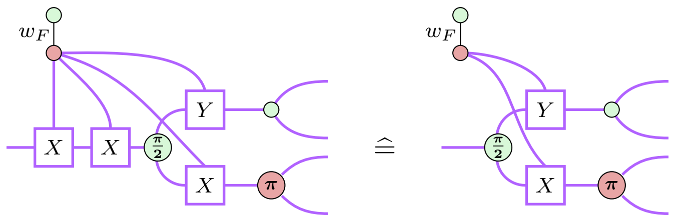
We can thus *guide* the movement of a fault gadget via a Pauli web. Consider for example the following diagram with a Pauli web:



By picking any spider inside the diagram, we can recover local stabiliser information that the Pauli web provides (if any) and apply (Stab$_{\text{fe}}$) to add targets according to this local stabiliser. Say for example, we are given the following fault gadget F and pick some spider that has highlighted legs. Then we can add targets to F as follows:



With target commutation by Proposition 4.9, removal of duplicate targets by Eq. (4.3) and decomposition of Y-targets by Eq. (4.7), we effectively recovered the scalar-ignorant rules (Fusion) (for same type targets) and (Pi-Copy) (for opposite type targets):

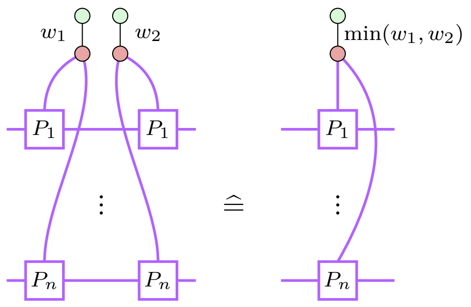


Iterating this, we can move multi-edge fault gadgets throughout the diagram, according to the same constraints that are put on π-phases by the ZX calculus. This technique will prove useful in Section 5.3.

4.4 Implementing Axioms for Multi-Edge Noise

Now that we can somewhat freely move fault gadgets around the diagram, updating their targets as we go along, we might encounter a special case: We can accumulate several fault gadgets that have an equivalent collection of targets, i.e. the same Pauli boxes on the same edges. These fault gadgets clearly represent congruent faults. Without fault gadgets, through repeated application of Proposition 3.15 we should be able to discard all but one and update its weight annotation to the minimum weight of all encountered annotations. As it turns out, Proposition 3.15 also applies to fault gadgets, and further we can show this purely diagrammatically, using its diagrammatic equivalent for edge-flips, (MERGE$_{\text{fe}}$) from Fig. 2:

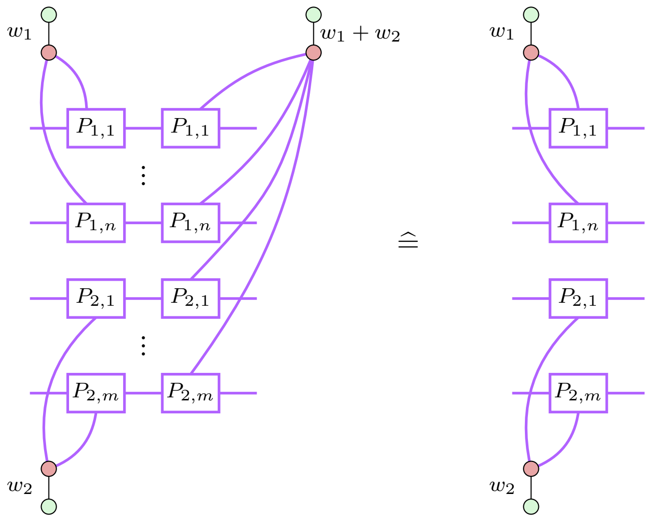
Proposition 4.11. Let $D_{\mathcal{F}}$ be a ZX diagram in GI-form, and let $F_1 \equiv F_2$ be congruent faults in $\mathcal{F}$. Then the fault gadgets for F_1, F_2 have the same targets on the same edges, and it holds that:



Proof. In Appendix A. □

However, this is not the only algebraic result that can be implemented diagrammatically. We can also apply a similar strategy to (COMB$_{\text{fe}}$) from Fig. 2, applying Proposition 3.16 to fault gadgets:

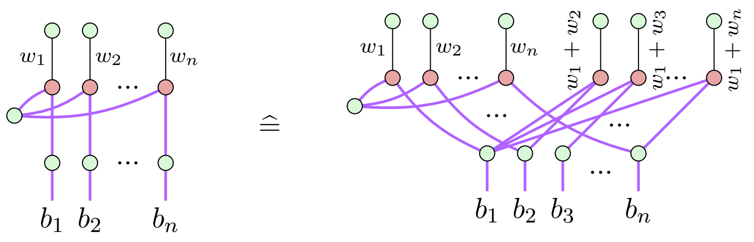
Proposition 4.12. Let $D_{\mathcal{F}}$ be a ZX diagram in GI-form, and let F_1, F_2 be arbitrary faults in $\mathcal{F}$, then it holds that:



Proof. In Appendix A. $\square$

Finally, we want to apply this to (Detect$_{\text{fe}}$), which implements Proposition 3.17 for multi-edge faults. However, we find that the axiom itself is already fairly complex. We introduce an intermediate step, that shows that from a set of detectable faults in the same detecting region, we can introduce their combinations (as a variation of (Comb$_{\text{fe}}$)):

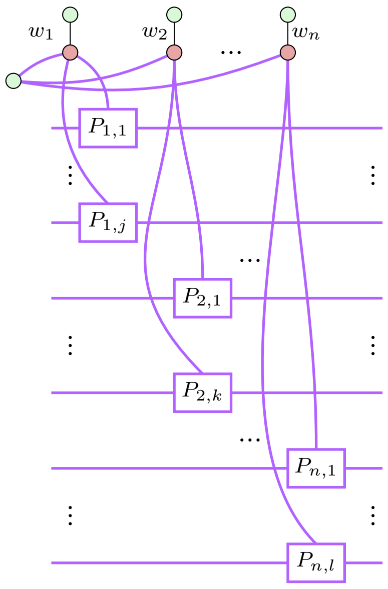
Proposition 4.13. For any $w_1, \ldots, w_n$ it holds that:



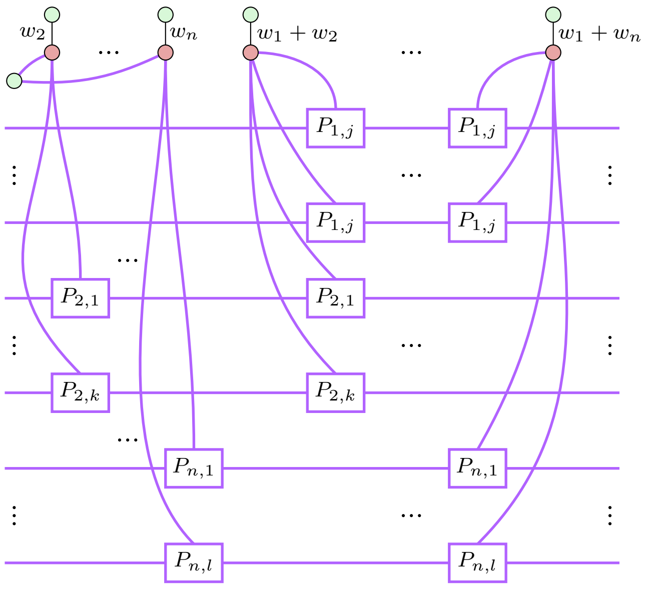
Proof. In Appendix A. $\square$

Then, we can show that (Detect$_{\text{fe}}$) may be lifted to multi-edge faults represented by fault gadgets. To be precise, in the most complex diagrammatic rewrite yet, we show that we can entirely remove the fault gadget whose spawning edge atomic fault we remove in (Detect$_{\text{fe}}$):

Proposition 4.14. For any $w_1, \ldots, w_n$ it holds that:



$\widehat{=}$



Proof. In Appendix A. □

We will make extensive use of these rules in Section 6, when considering how multiple atomic faults, represented by fault gadgets, interact diagrammatically.

5 Fault Signatures

So far, for a diagram D with a noise model $\mathcal{F}$, we implemented $\mathcal{F}$ into D to obtain a diagram $D_{\mathcal{F}}$. However, the goal is to achieve a separation in $D_{\mathcal{F}}$ between the original diagram's fault-free semantics and a diagrammatic description of $\mathcal{F}$. To make the separation process fault-equivalent, we will need to retain a fault's effect throughout. This involves finding, for some gadget of a fault F, a gadget of a fault F' with the same effect, but such that the gadget of F' only places targets on the boundary of D.

When we showed through a contrapositive that detecting regions are indeed those that detect faults in Proposition 2.33, we already constructed such boundary-only faults F' for undetectable faults. This used the fact from Proposition 2.32 that for every stabiliser of the diagram, we can find an element of the Pauli group that anticommutes only with this stabiliser. However, for detectable faults this method is insufficient; the method via Proposition 2.32 can only work for stabilising Pauli webs and not detecting regions as it would be unsound for some F' outside any detecting region to be effect equal to an F inside a detecting region. If we want to find F' even for detectable F, we need to expand the possible locations that F' may populate beyond the boundary of D.

This section first introduces 'flip operators', a notion that we use to find Pauli operators such that we can flip detecting regions individually, similar to Proposition 2.32. We use these additional operators to define 'signatures' of a fault F. A signature structurally describes an alternative F' with the same effect as F, limited to the boundary and employing flip operators. Subsequently, we implement the diagrammatic transformation to signature faults for fault gadgets.

5.1 Flip Operators

We enable flipping detecting regions individually, similar to Proposition 2.32 for stabilisers, via:

Definition 5.1 (Flip operators)**.** Given a ZX diagram D with edges E and a detecting region group $\mathcal{R}$, a *flip operator collection* is a set

$$\text{flipop} \subseteq \overline{\mathcal{P}^{|E|}}\,,$$

such that if $R_1, \ldots, R_d$ generate $\mathcal{R}$, for each R_i there is a unique combination of operators from flipop such that the composite fault flips *only* R_i.

Informally, the flip operator collection allows us to, for each independent generator R_i in a generating set $R_1, \ldots, R_d$, construct a Pauli operator that anticommutes only with R_i. Finding such a collection is possible for every diagram, through a process that adjusts the generators for $\mathcal{R}$ similar to Gaussian elimination for stabilisers [Got97, Sec. 4.1]:

Proposition 5.2. A flip operator collection exists for every ZX diagram.

Proof. Let $\mathcal{R}$ be the detecting region group of the diagram D and let $R_1, \ldots, R_d$ be generating for $\mathcal{R}$. Now choose an arbitrary region generator R_i and for it an arbitrary highlighted edge e. If e is highlighted by R_i in red, we set $p = Z$, otherwise we set $p = X$. This Pauli operator p anticommutes with the highlighting R_i defines for e and thus placing p on e would flip *at least* R_i.

To ensure it flips *only* R_i, we obtain new generators for all $j \neq i$ that p would flip:

$$R' \quad := \quad \{R_i\} \cup \{R_j \mid pR_j = R_j p\} \cup \{R_j R_i \mid R_i \neq R_j \wedge pR_j = -R_j p\}$$

Then the generator set R' is also generating $\mathcal{R}$, but for every $R_j \in R'$ except R_i that was previously flipped by p, we multiplied R_i onto it to flip it back. This results in p commuting with the highlighting e from every other detecting region R_j, so it only flips R_i.

We can iterate this procedure until we find a unique edge flip for each generator, yielding a flip operator collection. □

The choice of these flip operators is not unique: Choosing a different region or edge to start the elimination process with may lead to a different generating set or different flip operators entirely. However, once we construct a flip operator collection that is provably valid for a single generating set, it remains valid for all generating sets of $\mathcal{R}$. As in the proof above, we can simply change the generating set by multiplying one generator with all others. These operations may be exactly mirrored in the flip operator collection by multiplying the flip operator for a generator with the flip operators of others.

5.2 Signatures

The signature of a fault is now a description of its action on the boundary and used flip operators:

Definition 5.3 (Fault signature)**.** Let D be a diagram with boundary edges B and a flip operator collection flipop. The *signature* of a fault F is a tuple

$$(X_F, Z_F, L_F) \in \mathcal{P}(B) \times \mathcal{P}(B) \times \mathcal{P}(\text{flipop}),$$

1. where X_F, Z_F are the boundary edges upon which a X, Z flip should be placed, and
2. L_F is the set of flip operators that F requires,

such that instantiating the fault F_{sig} obtained by multiplying all edge flips and operators together obeys $\text{eff}(F_{sig}) = \text{eff}(F)$.

Via this definition, we ensure that the alternative faults we describe through signatures are unified in their interaction with the detecting regions. We can guarantee this as for a specific generator the combination of flip operators must be unique. Further, we can guarantee that we can always find at least one signature for any fault, regardless of the specific flip operator collection:

Proposition 5.4. Let D be a ZX diagram with a noise model $\mathcal{F}$, and let flipop be a flip operator collection for its detecting regions. Then every fault $F \in \langle \mathcal{F} \rangle$ has a signature F_{sig} defined on flipop.

Proof. Let $\langle W \rangle$ be the group of Pauli webs for D and let $R \subseteq W$ be the detecting generators. By Definition 5.1, for each $r \in R$ we can find a unique combination of flip operators from flipop such that their composite flips only r. Let this combination be given by the composite F_r. Then we define:

$$F' \coloneqq \prod_{r \in \mathrm{eff}(F) \cap R} F_r \,.$$

So we constructed a fault F' that is guaranteed to flip the same detecting webs as F.

Then, let $S^+ \subseteq W$ be the set of stabilising generators that F flips but F' does not flip (so they still have to be added to the effect of F'), and similarly let $S^- \subseteq W$ be the set of stabilising generators that F' flips but F does not (so they have to be removed from the effect of F', i.e. flipped back). By Proposition 2.32, we can find a fault T_s for each stabiliser $s \in (S^+ \cup S^-)$ such that T_s anticommutes only with s. Finally, we construct a signature by correcting the differences as indicated by S^+, S^-:

$$F_{sig} \coloneqq F' \cdot \left(\prod_{s^+ \in S^+} T_{s^+} \right) \cdot \left(\prod_{s^- \in S^-} T_{s^-} \right) . \qquad \square$$

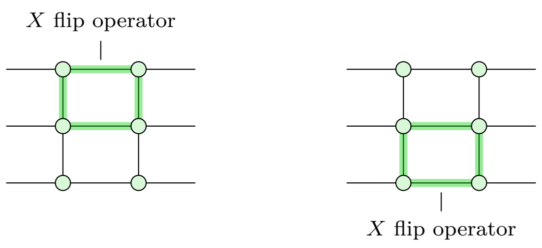
As a signature for a fault F can be instantiated to an actual fault F_{sig}, we may also model the signature via a fault gadget that places targets according to F_{sig}:

Definition 5.5 (Signature gadget)**.** Let D be a ZX diagram with a noise model $\mathcal{F}$ and a flip operator collection flipop. A fault gadget for $F \in \mathcal{F}$ is a *signature gadget* w.r.t. flipop if there is a signature defined through flipop that uses such that instancing this signature leads to a fault F_{sig} and $F \equiv F_{sig}$.

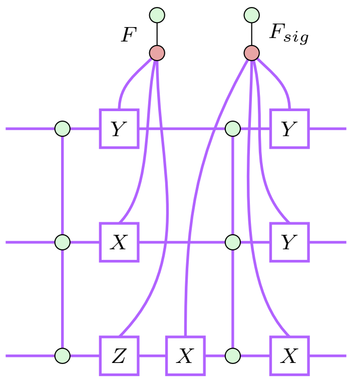
Since such F_{sig} derived from signatures of F are by definition effect equivalent to F, we are able to fault-equivalently replace F with some F_{sig} in the noise model. However, we have to show that such a replacement is possible diagrammatically. We thus proceed with a *gradual transformation* between a fault gadget and one of its signatures gadgets.

5.3 Transforming Gadgets Into Signatures

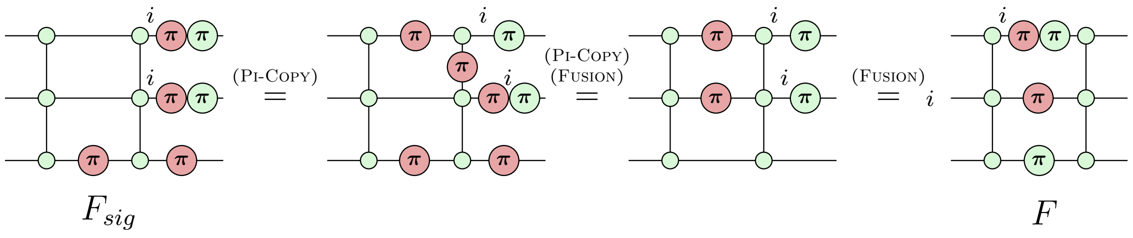
We now show that it is always possible, given two fault gadgets, to move one fault gadget such that it is exactly equal (i.e. congruent) to the other. This will be done by discovering a stabiliser between the fault gadgets, which induces a Pauli web and thus a way to rewrite one fault gadget into another with the procedure discussed in Section 4.3. For an illustrative example, we take the following diagram D that contains two detecting regions, for which we chose the indicated flip operators:



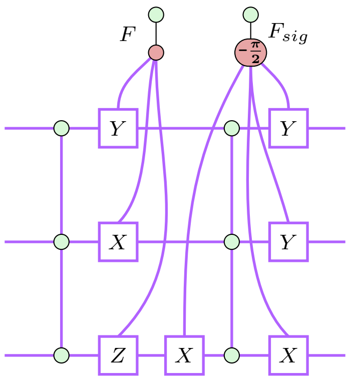
We now annotate the diagram D with simple noise model that has two atomic faults. One fault gadget represents some F, while the other represents one of its signatures F_{sig}. We thus obtain $D_{\mathcal{F}}$:



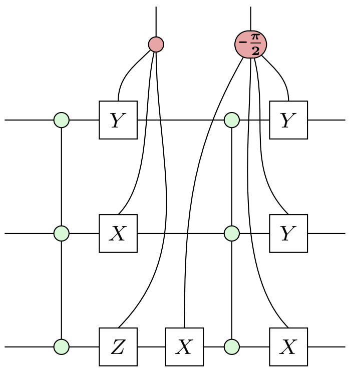
Note that since stabilisers can only have phases ± 1 and not $\pm i$, finding a stabiliser between the gadgets requires care with the scalar accurate relationship between the two faults that holds in this exact diagram configuration. In the above example, as we decompose $Y = iXZ$ and keep this order on the edges, introducing a Z fault on just the spawning edge of F_{sig} would result in a fault that can be pushed (using (Pi-Copy), (Fusion) and (Elim)) to be exactly F, up to a scalar of i:



Thus, to proceed, we need to capture this in the diagram by augmenting F_{sig} using (XPhase$_{\text{fe}}$), so that it introduces the necessary phase of i when provided with a Z-flip on the spawning edge:



Now we modify $D_{\mathcal{F}}$, by converting the green spider of both fault gadgets into a boundary. This is a distinctly different diagram, so we stop annotating the noise model for now:



We say that we *opened* the gadgets to obtain D_{open}. Note that this is not a traditional ZX rewrite, since it adds boundary vertices and thus changes the dimensionality of the possible interpretations of $D_{\mathcal{F}}$.

Because $\text{eff}(F) = \text{eff}(F_{sig})$ by definition of a faults signature, placing a π-phase on both fault gadgets would flip all webs from the effect twice, providing an overall effect of $\emptyset = \text{eff}(I)$. Similarly, we can introduce two green π-phases at the new boundary edges. However, since this also has an overall effect of $\emptyset$, it is much less a fault, but rather a new stabiliser of D_{open}. We can prove that this works for every pair of effect equivalent fault gadgets, and that in fact only such pairs provide stabilisers:

Proposition 5.6. Let $D_{\mathcal{F}}$ be a ZX diagram with two faults $F_1, F_2 \in \mathcal{F}$, implemented as fault gadgets. Further, let D_{open} be obtained from opening the fault gadgets of F_1, F_2 in $D_{\mathcal{F}}$ and let b_1, b_2 be the newly created boundary edges corresponding to the gadgets of F_1, F_2 respectively. Then there is a stabiliser of D_{open} consisting of just two Z edge flips on b_1, b_2 if and only if $\text{eff}(F_1) = \text{eff}(F_2)$.

Proof. The proof starts with assuming $\text{eff}(F_1) = \text{eff}(F_2)$ and reversibly working towards the existence of a stabiliser. We note that we can recover F_2 from the composite using

F_1, since even if they anticommute we consider faults up to global phase, and thus we can write

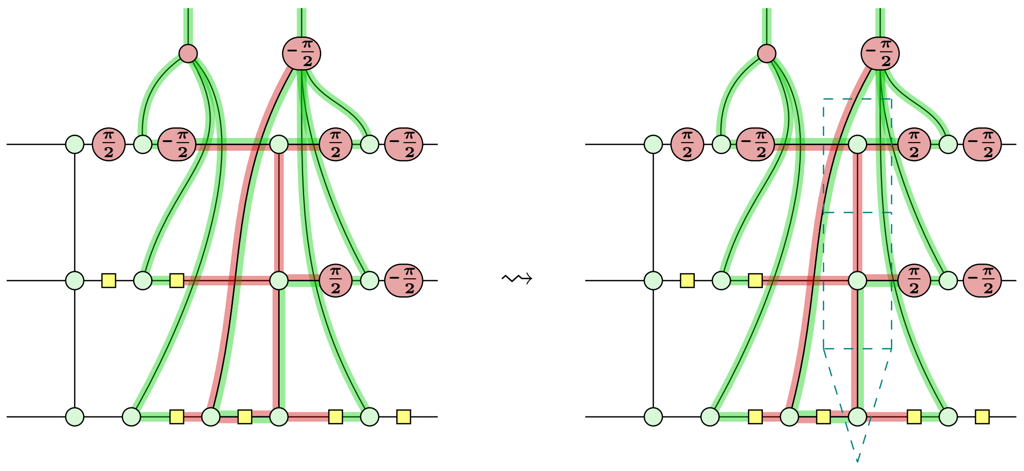
$$F_1F_2F_1 = \pm F_1F_1F_2 = F_1F_1F_2 = IF_2 = F_2\,.$$

We can then derive that the composite fault must have trivial effect:

$$\begin{aligned}\mathrm{eff}(F_1) = \mathrm{eff}(F_2) \quad &\Leftrightarrow \quad \mathrm{eff}(F_1) = \mathrm{eff}(F_1F_2F_1) \quad \Leftrightarrow \quad \mathrm{eff}(F_1) = \mathrm{eff}(F_1) \,\oplus\, \mathrm{eff}(F_1F_2)\\ &\Leftrightarrow \quad \emptyset = \mathrm{eff}(F_1F_2)\,.\end{aligned}$$

We acknowledge that since b_1, b_2 are boundary edges created from the spawning edges of the fault gadgets, placing a Z flip on either of them must have the same effect as F_1, F_2 respectively and placing a Z flip on both edges must have the same effect as the composite F_1F_2. But then placing $Z \otimes Z$ on b_1, b_2 has trivial effect and by Proposition 3.10 this is exactly the case when $\pm Z \otimes Z$ is a stabiliser of D_{open}. □

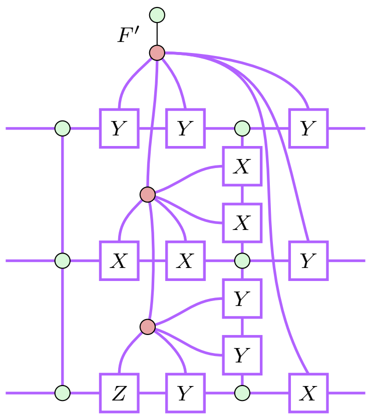
By Definition 2.27, a stabiliser can be directly associated with a stabilising Pauli web. Necessarily, we can also apply this to D_{open}, and obtain a Pauli web that highlights only the two newly obtained boundary edges in green. This Pauli web provides us with a recipe to rewrite one fault gadget into another. We section the web into blocks to encase each spider of the original D with highlighted legs separately:



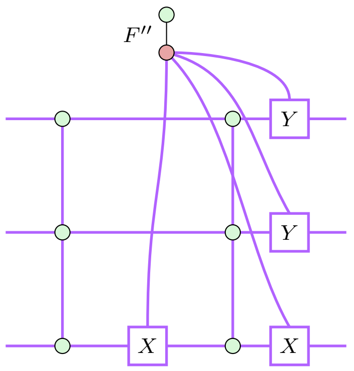
Since the local webs inside the blocks represent local stabilisers of their spiders, we can apply (Stab$_{\text{fe}}$) to add all of these stabilisers to the fault gadget for F. However, we only do this for blocks where the spider does not belong to any fault gadget, since we want to move the fault gadget only through the original diagram D, not through or into other fault gadgets[5]. Returning to $D_{\mathcal{F}}$, we now obtain a new fault F' from

[5]In particular, we want to avoid moving a fault gadget into itself, which would lack a clear interpretation.

F by adding the local stabilisers[6]:



Finally, we deduplicate targets on all edges as much as possible via gadget commutation (Proposition 4.9), converting between Y and X, Z targets (Eq. (4.7)), fault phase elimination (with (XPHASE$_{fe}$)) and target elimination (Eq. (4.3)):



Then by construction of the Pauli web, we have $F'' \equiv F_{sig}$ which concludes the rewrite.

Note that throughout the process, we only opened the diagram to derive a Pauli web as a guidance for which targets to add to F. In the diagram $D_{\mathcal{F}}$, we never actually changed the spawning edge, so the weight annotation for F directly carries over to F''. Due to this we can simply annotate F_{sig} with that same weight too.

We formalise for future reference:

Proposition 5.7. Let F_1, F_2 be two faults with $\text{eff}(F_1) = \text{eff}(F_2)$ formalised as fault gadgets in a ZX diagram $D_{\mathcal{F}}$. Then we can fault-equivalently rewrite the gadget for F_1 into a gadget that is congruent to the gadget for F_2.

Proof. Obtain the diagram D_{open} by opening both fault gadgets. By Proposition 5.6 we can obtain a $\pm Z \otimes Z$ stabiliser on the newly created boundary edges, and through Definition 2.27 we obtain a Pauli web for this stabiliser. Restricting this Pauli web to spiders that belong to the original diagram D provides a way to rewrite the gadget using (STAB$_{fe}$), (XPHASE$_{fe}$) as well as Proposition 4.9 and Eqs. (4.3) and (4.7), as outlined above. □

[6]We do not draw F_{sig} anymore and unfuse the distributing spider to reduce clutter.

6 Completeness - Part II: Normal Forms and Final Proof

Let us revise what we covered in Sections 3 to 5:

- We restated w-fault boundedness, and through it all related notions, as element-wise comparisons of the weighting function ewt for fault effects in Section 3. Additionally, we sketched how we want to obtain and enumerate this function diagrammatically by separating a diagrams fault-free semantics from the description of its noise model.
- We explicitly modeled atomic faults inside the diagram and discovered how they change when moved throughout the diagram in Section 4.
- We identified issues when trying to move detectable faults outside the diagram, and proposed the intermediate formalism of signatures to resolve them, along with a diagrammatic transformation of fault gadgets to signatures in Section 5.

Now, we are left with a diagram that has all fault gadgets collected as signature gadgets, so for a fixed choice of flip operators each gadget remains in minimal contact with the original diagram. For this last section of part concerned with completeness, complementing the work from Section 3, it remains to show that we can indeed fully disconnect signature gadgets from the original diagram, and finally fully enumerate the function ewt.

6.1 Sinks

We recall that for a diagram D with detecting region group generated by $R_1, \ldots, R_d$, we can use a flip operator collection to individually flip / 'steer' the sign of individual R_i. When used as set out in Definition 5.1, these collections suffice to define and think about faults signatures algebraically.

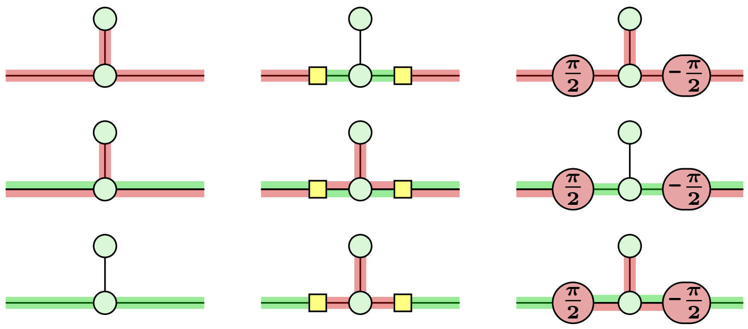
However, once we instantiate a signature to a particular fault gadget on the diagram, we notice two downsides of this definition:

1. The flip operators, living in $\overline{\mathcal{P}^{|E|}}$, could have an arbitrary number of non-trivial Pauli components, increasing complexity for handling them. Ideally, we could unify the flip operators such that every operator has exactly one and exactly the same non-trivial Pauli component. This would enable us to prove subsequent statements for this single type of operator only.
2. The resulting fault gadget is still stuck inside the detecting region and cannot leave it. Ideally, we resolve this and make some progress towards the separation between diagram semantics and its noise model.

We address both issues with one additional notion, which we will call *sinks*.

In Section 4 we already derived a construction diagrammatically modeling something that arbitrarily flips Pauli webs, while only being activated by a single edge flip: *fault* *gadgets*! We recall that a fault flips a detecting region R exactly when, seeing both as a Pauli operator, it anticommutes with R.

Similarly, we can classify fault gadgets in regard to whether they commute andor anticommute with a detecting region R. Instead of doing this algebraically, we will work on visualising this in the diagram. To start, we observe that the controlling edge coming out of a Pauli box is highlighted red if and only if that Pauli anticommutes with the edge highlighting it is placed on[7]:

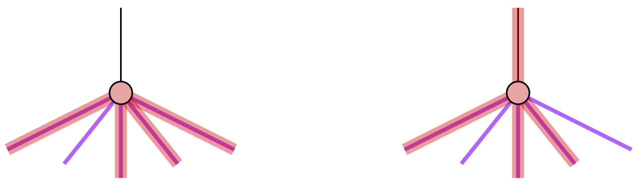


Then, similar to how we constructed fault gadgets, we can extend this to general multi-edge Pauli box constructions:

Proposition 6.1. A fault gadget represents a fault that anticommutes with a Pauli web P if and only if the gadget extends P such that the spawning edge of the gadget is highlighted red.

Proof. Let k be the number of Pauli boxes whose controlling edge must be highlighted red if they are placed on their edges as targets of the gadget. Recall that as above, these are exactly the Pauli boxes representing Pauli operators that anticommute with the Pauli represented by the edge highlighting. Further recall from Section 2.1 that two multi-qubit Pauli operators anticommute if and only if an odd number of their single-qubit operators anticommute.

Now if k is even, the red spider bundling all the controlling edges from the gadgets targets has a valid local web without highlighting the spawning edge; conversely if k is odd, the spawning edge must be highlighted as well for the web to be valid:



k is even $\qquad$ k is odd

Thus, the spawning edge is highlighted red if and only if k is odd, which as outlined is the case if and only if the fault of the gadget and P anticommute. □

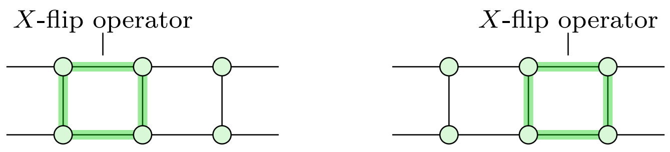
[7]This edge does not necessarily have to belong to a detecting region. Indeed, the edge can also be part of a stabilising Pauli web, in which case the red highlighting identifies Pauli boxes that would flip this stabilising web.

It is through this extension that we can employ fault gadgets to tackle the first downside of flip operators: For each flip operator L we place a fault gadget F_L into the diagram and replace L by a flip operator L' consisting only of a Z-axis flip on the spawning edge of F_L. Thus, we have significantly simplified the flip operators, so that they are completely described by a single flip on the spawning edge of the fault gadget.

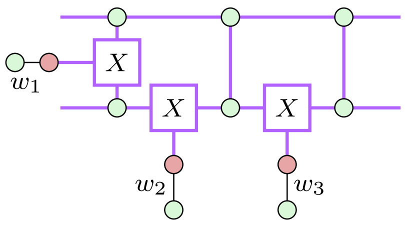
However, we do not want this new fault gadget to actually introduce any new faults. Thus, we fully idealise its spawning edge, providing us with the required notion of a 'sink':

Definition 6.2 (Sinks)**.** A *sink* for a flip operator $L \in$ flipop is a fault gadget that places targets according to L, with a fully idealised spawning edge.

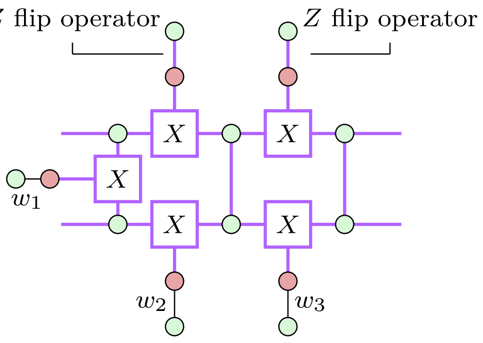
To construct an example we will be using for this section, consider the following diagram D that has two detecting region generators with single-edge flip operators chosen as indicated:



We consider a noise model $\mathcal{F}$ that models some X edge flips happening for both detecting regions and obtain its implementation with fault gadgets as $D_{\mathcal{F}}$ as indicated:

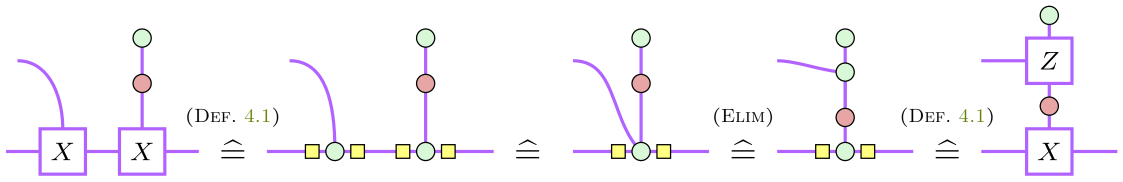


Now add sinks, i.e. fully idealised gadgets according to the flip operators, and obtain new operators unified in type:

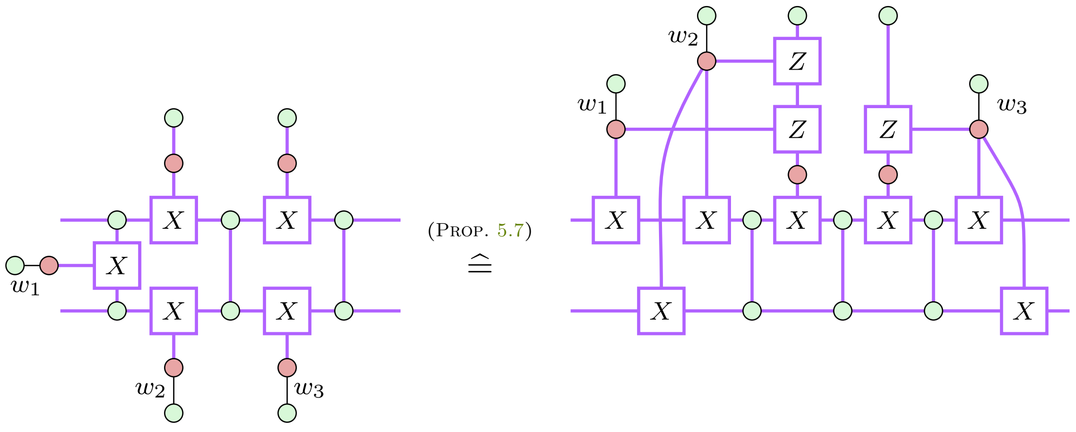


Effectively, the flip operators are shifted from a proper diagram edge into a special fault gadget. By definition, this also works for multi-edge flip operators, thus providing a way to handle such multi-edge operators with single-edge replacements.

We now transform fault gadgets into their signatures via the method discussed in Proposition 5.7. Signatures employ flip operators which are now inside fault gadgets and moving targets *inside* fault gadgets may seem like the exact thing that we wanted to avoid in Section 5.3. However, as sinks are fully idealised fault gadgets, we may view them as part of the original diagram during application of Proposition 5.7. The diagrammatic conversion still works; for example, we can move an X target inside an X sink:



We can thus move all targets anticommuting with their detecting regions onto their spawning edge. Continuing with the running example, we employ this as follows:

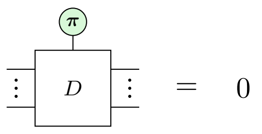


We now observe that we have yet to tackle the second downside of flip operators: Even using sinks, our targets are still inside the detecting regions. Additionally, the original diagram is modified quite heavily by now, by adding numerous fault gadgets as a standardised way to model either faults or flip operators. We proceed to show how to fully separate the gadgets describing the noise model and the original diagram, resolving both issues at once.

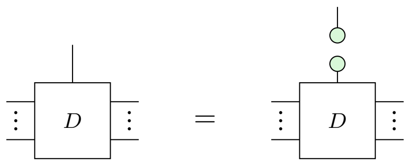
6.2 Recovering The Diagram

To start on this separation, we propose the following rewrite:

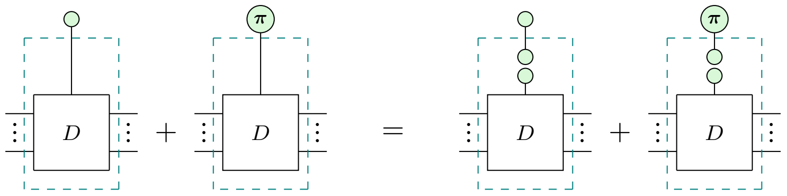
Proposition 6.3. Let D be a Clifford ZX diagram with boundary edges B, and let $b \in B$ be a boundary edge such that capping b with a green π-phase makes the composite diagram zero, i.e.:



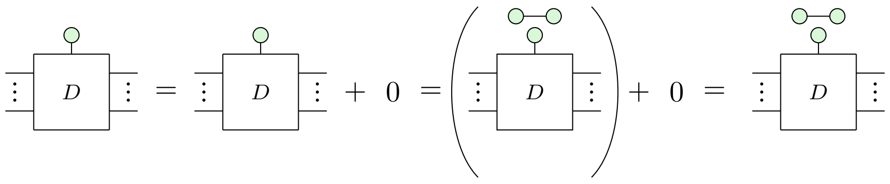
Then the following two diagrams are semantically equivalent:

 (6.1)

Proof. We employ the well known fact that any finite linear map can be expressed as a sum of its actions on all elements of an orthonormal basis [NC10]. For our purposes, we choose the X-basis, represented through green $\{0, \pi\}$-phase spiders [KvdW24]. Since the remainder of the diagram D remains unchanged, we only need to show the action on the basis elements on the boundary edge b is the same, without introducing a relative phase. That is, we must show that



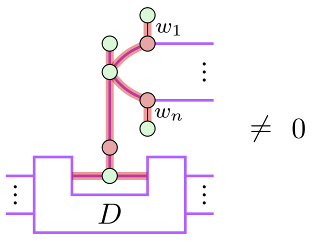
holds up to a global scalar. By assumption the second summand on the LHS is 0, the second summand on the RHS contains a 0 scalar, and thus we are left with:



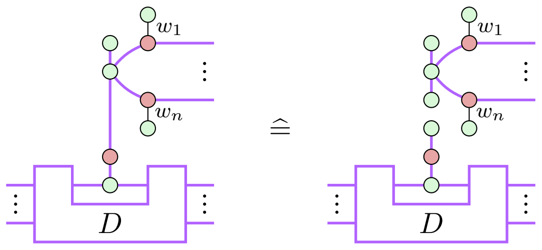
This trivially holds up to the depicted global scalar 2, as required. So the two diagrams in Eq. (6.1) are semantically equivalent. □

We immediately use Proposition 6.3 to show that we can disconnect flip operators from the detecting regions in which they are contained:

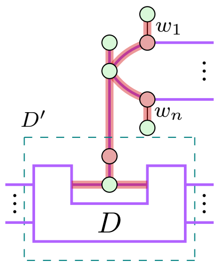
Proposition 6.4. Let D be a Clifford ZX diagram such that



where the highlighting forms a detecting region in D and the overall diagram is non-zero. Then the following is a fault-equivalent rewrite:

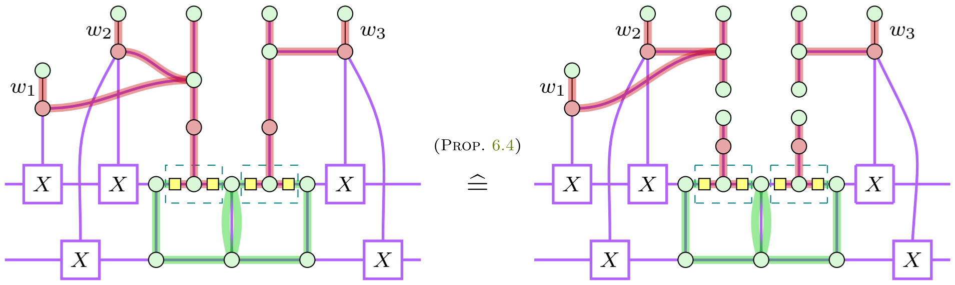


Proof. Consider the following subdiagram D':



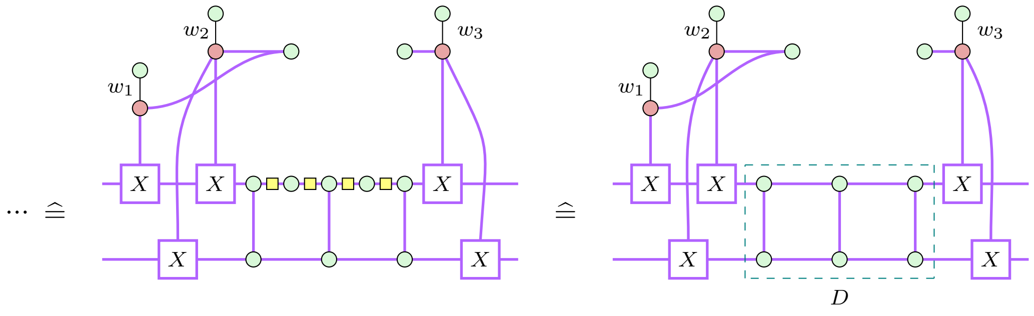
Since the red highlighting extends to a detecting region in D, capping off D' with a green π-phase would result in that detecting region being flipped and thus D' becoming zero. We then apply Proposition 6.3 as a fully idealised variant and obtain a new diagram where the sink is disconnected. Since the Clifford ZX calculus is complete [Bac14], we must be able to obtain this new diagram using just idealised Clifford rewrites. But since idealised rewrites are trivially fault-equivalent, the full rewrite is fault-equivalent. □

We continue our running example by first realising all targets inside sinks to Z spiders, fusing them together and subsequently applying the proposition we have just proven[8]:



Although the total number of detecting regions increases, the number of detecting regions that are flippable by faults stays the same. For all future rewrites, we may ignore all detecting regions in the main diagram for determining detectable effects, since there are no fault gadgets that have targets inside them.

Once we used this rewrite to disconnect the flip operators, we can deconstruct sinks again using idealised Clifford rules and finally recover the original diagram as completely idealised:

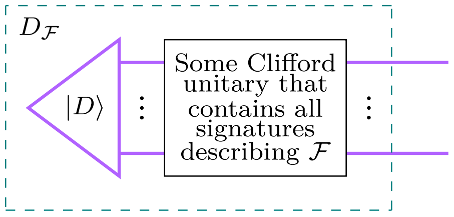


We define this to be our first normal form:

[8]Drawing two overlapping detecting regions at the same time is difficult, so we opt for bending the overlapping highlightings to distinguish them. This has no other special interpretation.

Definition 6.5 (SN-form)**.** Let D be a diagram with a noise model $\mathcal{F}$ and $D_{\mathcal{F}}$ the implementation of $\mathcal{F}$ using fault gadgets. $D_{\mathcal{F}}$ is in *signature normal form* ('SN-form' for short) if it is in GI-form, every fault gadget is a signature gadget w.r.t. to the same flip operator collection, and no fault gadget has targets inside detecting regions in D.

As in our example, the signature normal form of a diagram $D_{\mathcal{F}}$ that implements a noise model $\mathcal{F}$ on a base diagram D fits the template:


$D_{\mathcal{F}}$
$|D\rangle$
Some Clifford unitary that contains all signatures describing $\mathcal{F}$

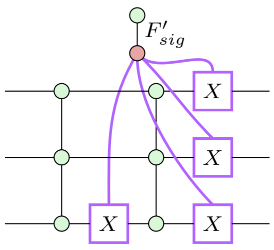
We could have brought any diagram under any weighted noise model into this form. Indeed, we can reason:

Proposition 6.6. For every ZX diagram D with some noise model $\mathcal{F}$ the implementing diagram $D_{\mathcal{F}}$ has a signature normal form into which it can be fault-equivalently rewritten.

Proof. By construction the diagram $D_{\mathcal{F}}$ is in GI-form. Then by Proposition 5.2, we can find a flip operator collection flipop, and via Proposition 5.4 there exist signatures for every fault $F \in \langle \mathcal{F} \rangle$ w.r.t. the collection flipop. We may add sinks via idealised Clifford rewrites, transform fault gadgets into their signatures via Proposition 5.7, and disconnect flip operators from sinks via Proposition 6.4. Finally, the remaining sinks are reduced via idealised Clifford rewrites, to obtain a diagram $D'_{\mathcal{F}}$.

The diagram $D'_{\mathcal{F}}$ is still in GI-form since all applied rewrites preserve it. Further, all fault gadgets have targets that are either on the boundary or activate flip operators, so every fault gadget is a signature gadget. Since disconnecting the flip operators leaves them free floating and in particular outside D, no fault gadget can have targets inside D. So $D'_{\mathcal{F}}$ is in signature normal form. $\square$

We explicitly remind ourselves that this signature normal form is not unique for two reasons. For one, we can change our set of flip operators together with the generating set for detecting regions. The choice of flip operators has an influence on the action of faults on the boundary and thus their signature. This remains true even as flip operators as such are now disconnected from the original diagram and exist free-floating, only connected to fault gadgets. For another, a single fault may have multiple distinct signatures; for example, the fault F from Section 5.3 may have the following alternative signature:

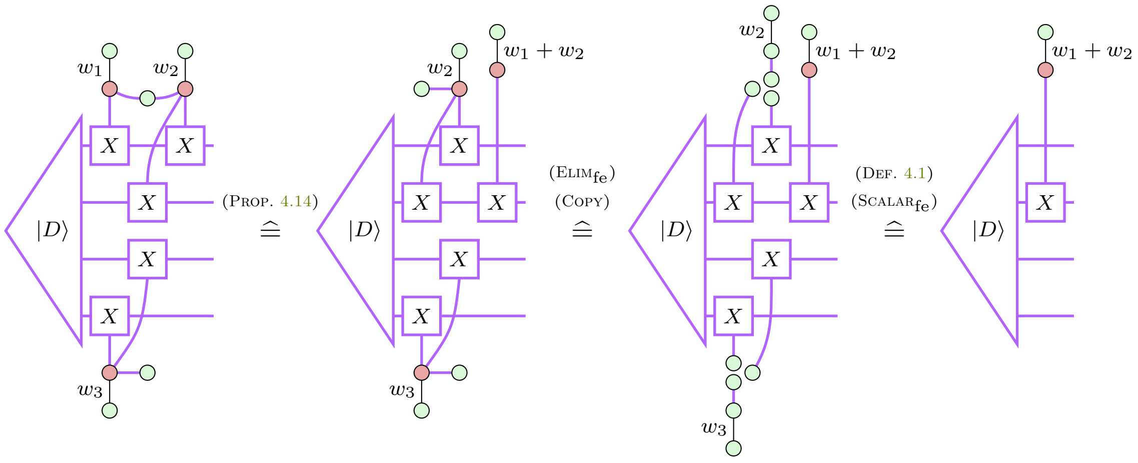


We tackle these challenges separately, and first focus on flip operators by completely eliminating them without leaving SN-form.

6.3 Enumerating Undetectable Faults

We recall that for fault equivalence through Corollary 3.13 we only need equivalence of all undetectable effects. Although we do not handle effects yet, we can already get rid of detectable signatures fault-equivalently. Signatures still describe faults, only in a more structured manner, so we could simply apply Proposition 3.17 to algebraically replace all atomic faults detected by the same detecting region.

However, this replacement is not yet purely diagrammatic. For a diagrammatic implementation of Proposition 3.17 we turn to the axiom (Detect$_{\text{fe}}$) for single-edge faults, and to Proposition 4.14 for multi-edge faults. This is applied to our running example, removing the detecting regions and thus eliminating the flip operators entirely:



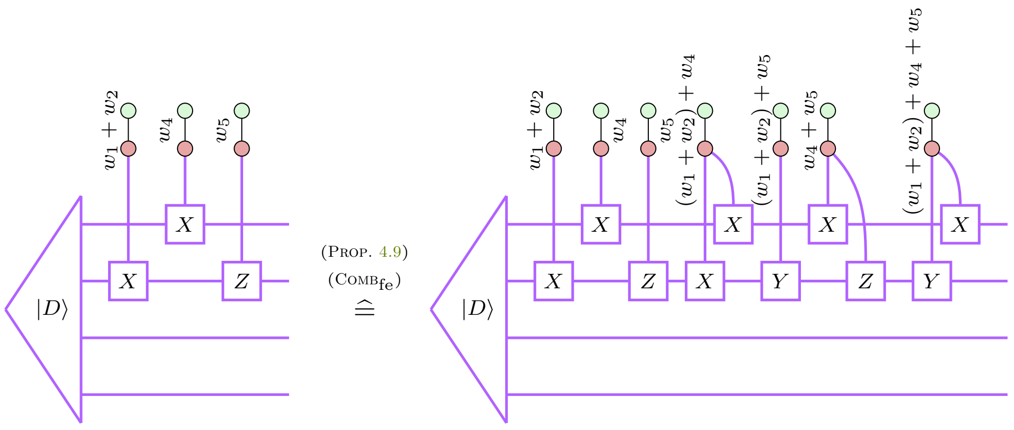
The faults weighted as w_1 and w_2 combine to yield a fault weighted as $w_1 + w_2$. The fault weighted as w_3 is eliminated entirely, since it is the only one populating its flip operator.

We can arbitrarily iterate this procedure, resulting in a diagram that requires no flip operators as it encodes no detectable faults. Instead, the diagram now encodes a set of undetectable atomic faults that is still generating for all undetectable composite faults the noise model initially described, since our transformations are fault-equivalent. Additionally, the diagram clearly remains in SN-form as all rewrites preserve it.

At this point we notice that we are not yet guaranteed to have explicitly enumerated *all* undetectable faults that might occur. To be precise, enumerating all combinations of faults detectable by a single detecting region does not consider combinations of faults from different regions. Thus, it remains to actually unfold all undetectable composite faults.

For a pair of gadgets representing faults F_1, F_2, recall that we can introduce their product $F_1 F_2$ into the noise model, either algebraically by Proposition 3.16 or diagrammatically by implementing the axiom (Comb$_{\text{fe}}$) for multi-edge noise in Proposition 4.12. Since we found that fault gadgets commute in Proposition 4.9 we can arbitrarily rearrange the fault gadgets on the boundary edges of the diagram. Thus, for every possible sequence of atomic faults in the noise model, arrange that sequence using gadget commutation and produce the resulting composite fault gadget by combining fault gadgets in pairs using Proposition 4.12.

In our example there is only a single undetectable fault left, so this operation is not particularly interesting. However, suppose we had two additional edge flips weighted as w_4 and w_5 on the boundary. They can not be part of any detecting region, so all previous operations would not affect their fault gadgets. Then we have:



The result is a full enumeration of all undetectable fault signatures the noise model could possibly generate, while remaining in SN-form the whole time. We note that since we enumerate all possible signatures that the noise model generates for each equivalence class, we in particular generate a fault gadget annotated with the minimum weight associated with faults in that class. That is, for each possible effect equivalence class, the diagram will now contain *at least one* fault gadget F for this class that is annotated with $\text{ewt}(\text{eff}(F))$, i.e. the minimum weight produced in the class.

We close by observing that the outlined naïve enumeration procedure introduces a number of fault signature factorial in the number of atomic faults, producing a large quantity of signatures that are either direct duplicates (i.e. congruent) or equivalent (i.e. flip the same webs). In the upcoming section, we handle these duplicates and find a procedure to identify a canonical signature for each equivalence class.

Remark 6.7. Even when using more sophisticated and efficient methods outside ZX diagrams to determine which exact combinations of atomic faults produce unseen composite faults, the number of signatures we are required to enumerate might still be exponential in the worst-case. This is expected, as we remind ourselves that checking fault equivalence is NP-hard [RPK25, Thm. 3.10]. The remainder of the completeness proof may be performed in a polynomial number of steps, so finding an efficient procedure here would at least provide strong evidence of P = NP.

6.4 Full Normalisation and Completeness

For each fault effect W, we remain with a collection $\bar{F}_W$ of signatures such that $\text{eff}(F) = W$ for all $F \in \bar{F}_W$. Reducing this collection down to a single representative requires characterising how the signatures in the collection differ from each other. We can prove that these differences are only up to stabilisers:

Proposition 6.8. Let D be a ZX diagram where $S_1, \ldots, S_n$ generates its stabiliser group $\mathcal{S}$, and let F_1, F_2 be two faults obtained from signatures defined through the same set of flip operators. Then we have that $\text{eff}(F_1) = \text{eff}(F_2)$ if and only if there is a subset of generator indices $J \subseteq 1, \ldots, n$ such that

$$F_1 \equiv F_2 \cdot \prod_{j \in J} S_j \,.$$

Proof. We start with the assumption $\text{eff}(F_1) = \text{eff}(F_2)$ and work towards $F_1 = F_2 \cdot S$ for some stabiliser S.

By Corollary 3.5, $\text{eff}(F_1) = \text{eff}(F_2)$ is the case if and only if $\text{eff}(F_1 F_2) = \emptyset$. We are operating only on the boundary, so we can apply Proposition 3.10 and see that we can have $\text{eff}(F_1 F_2) = \emptyset$ if and only if either $F_1 F_2$ or $-F_1 F_2$ is a stabiliser. We observe this is the case if and only if either $F_2 F_1$ or $-F_2 F_1$ are a stabiliser, regardless of whether F_1 and F_2 commute or not. Since we can fully generate the stabiliser group, this in turn is the case if and only if we are able to find a subset J of generator indices with

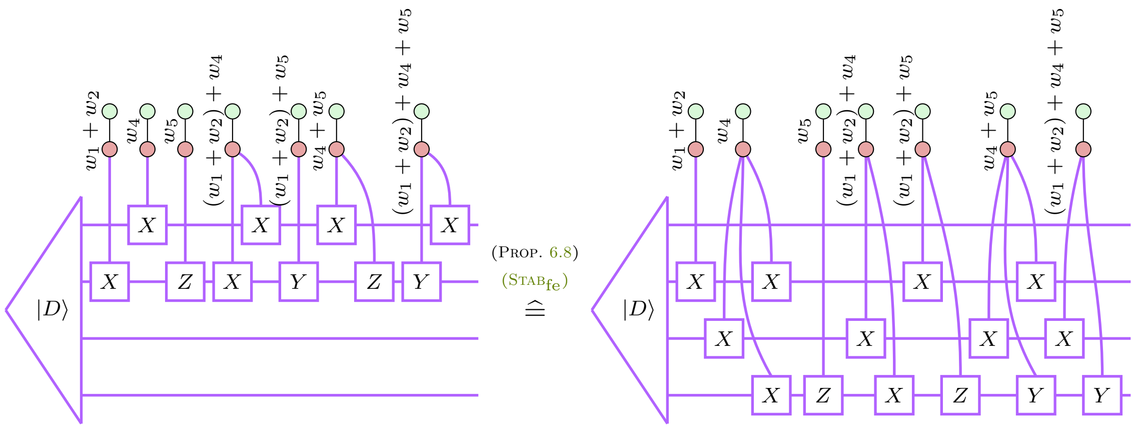
$$\pm F_2 F_1 \equiv \pm \prod_{j \in J} S_j \,.$$

We multiply both sides by F_2 and find that, since we treat faults up to their phase, we can simply discard the sign and show the claim:

$$F_2 F_2 F_1 \equiv F_2 \cdot \prod_{j \in J} S_j \quad \Leftrightarrow \quad I F_1 \equiv F_2 \cdot \prod_{j \in J} S_j \quad \Leftrightarrow \quad F_1 \equiv F_2 \cdot \prod_{j \in J} S_j \,. \qquad \square$$

These conversions between signatures can, once identified, be easily implemented in the diagram through (STAB$_{\text{fe}}$) and the target merging rule in Eq. (4.3) from Section 4. Thus, if we pick some signature in the collection $\bar{F}_W$ to be the 'canonical' representative F^*, we can fault-equivalently rewrite every other signature gadget from $\bar{F}_W$ into a gadget exactly matching F^*. It does not matter which representative we pick, as long as it is consistent in the corresponding effect classes across all semantically equivalent diagrams. Changing representatives is as easy as applying Proposition 6.8 again. For our purposes, we will use the *lexicographically smallest* Pauli operator.

In our example, the diagram has stabiliser generators $ZZII, IZZI, IIZZ$, and $XXXX$. Thus, we would apply the following rewrite:



Finally, we merge all signature gadgets with gadget deduplication through Proposition 4.11, leaving only one representative F^* per class. By construction, F^* is annotated with the minimal weight of all faults in the class.

We ultimately have a one-to-one correspondence between fault effects and these canonical representatives of signatures. We formalise this form of the diagram:

Definition 6.9 (rSN-form)**.** Let $D_\mathcal{F}$ be a ZX diagram that implements a noise model with fault gadgets. We say that $D_\mathcal{F}$ is in *reduced* signature normal form ('rSN-form' for short) if

1. $D_\mathcal{F}$ is in signature normal form, **and**
2. every gadget F^* is the lexicographically smallest in its equivalence class, **and**
3. every equivalence class has exactly one representative, **and**
4. gadgets are ordered lexicographically ascending, **and**
5. the gadget F^* is annotated with $\text{ewt}(\text{eff}(F^*))$.

We generalise from our example and observe that every diagram that implements its noise model with fault gadgets has such a normal form:

Proposition 6.10. Every ZX diagram $D_\mathcal{F}$ implementing some noise model $\mathcal{F}$ using fault gadgets has a reduced signature normal form. The gadgets in the reduced signature normal form *uniquely* enumerate the function ewt of $\mathcal{F}$ for all undetectable effects.

Proof. By Proposition 6.6, $D_\mathcal{F}$ has a signature normal form. We can remove all detectable faults with Proposition 4.14, and further generate at least one gadget for all effect classes using Proposition 4.9 and Proposition 4.12. The gadgets are reduced to the lexicographically smallest equivalent in their effect class using Proposition 6.8 and (Stab$_{\text{fe}}$), and deduplicated using the implementation of (MERGE$_{\mathrm{fe}}$) in Proposition 4.11. Thus, by construction the fault gadgets already enumerate the function ewt of $\mathcal{F}$ for all undetectable effects.

Finally, the gadgets are reordered using Proposition 4.12 to be sorted in lexicographically ascending order. As we collapsed all duplicates, this ordering is necessarily unique and so is the resulting enumeration, leaving the obtained diagram in reduced signature normal form. □

All that remains is to revisit the theorem:

Theorem 3.18 (Completeness for fault equivalence)**.** The ZX calculus given through axiomatisation in Fig. 2 is complete for fault equivalence.

Proof. Let D_1, D_2 be two ZX diagrams with respective noise models $\mathcal{F}_1, \mathcal{F}_2$ that are fault-equivalent under these noise models. Implement these noise models into the diagrams, obtaining $D_{1,\mathcal{F}_1}, D_{2,\mathcal{F}_2}$, using fault gadgets by Theorem 4.6.

By Proposition 6.10, both $D_{1,\mathcal{F}_1}$ and $D_{2,\mathcal{F}_2}$ have a reduced signature normal form $D^{\mathrm{nf}}_{1,\mathcal{F}_1}$ and $D^{\mathrm{nf}}_{2,\mathcal{F}_2}$. This normal form exactly and uniquely enumerates the effect weighting functions for $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively.

D_1 and D_2 are assumed to be fault-equivalent w.r.t. $\mathcal{F}_1, \mathcal{F}_2$, so by Corollary 3.13, their sets of undetectable effects must be the same and their effect weighting functions must agree on all undetectable effects. Then the subdiagrams containing the fault gadgets in $D^{\mathrm{nf}}_{1,\mathcal{F}_1}$ and $D^{\mathrm{nf}}_{2,\mathcal{F}_2}$ must be identical. All potential diagrammatic differences between $D^{\mathrm{nf}}_{1,\mathcal{F}_1}$ and $D^{\mathrm{nf}}_{2,\mathcal{F}_2}$ have to be inside the fully idealised D_1 and D_2. However, they must at least obey $D_1 = D_2$, as through Proposition 2.19 fault equivalence implies semantic equivalence.

Finally, all rewrites up to these reduced signature normal forms are fault-equivalent, and provably so using just the axioms in Fig. 2 and in particular reversible. As we have $D_1 = D_2$, we can rewrite the idealised D_1 into D_2 using the idealised Clifford axioms from Fig. 2, since the Clifford ZX calculus is complete for semantic equivalence [Bac14]. But then we can rewrite $D_{1,\mathcal{F}_1}$ into $D_{2,\mathcal{F}_2}$ fault-equivalently. □

So we have shown that we can obtain a ZX calculus that is complete for fault equivalence of Clifford ZX diagrams. Using the restatement of w-fault boundedness formulated in Proposition 3.12, we can obtain three directly related ZX calculi that are sound and complete for w-fault equivalence, fault boundedness, and w-fault boundedness respectively.

6.5 Extending Completeness to w-Fault Boundedness

We aim to obtain the calculus that is complete for w-fault boundedness through combining the calculus that is complete for w-fault equivalence and the calculus that is complete for fault boundedness. This is achieved through decomposing the former into a combined statement of the latter two. However, we first have to show that this decomposition itself is sound and complete.

Through transitivity of w-fault boundedness, we can directly show that a decomposition of w-fault boundedness is sound:

Proposition 6.11. For ZX diagrams D_1, D_2, D_3 with respective noise models $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ it holds that

$$D_1 \underset{w}{\hat{=}} D_2 \text{ and } D_2 \hat{\leq} D_3 \qquad \Longrightarrow \qquad D_1 \underset{w}{\hat{\leq}} D_3$$

Proof. $D_1 \underset{w}{\hat{=}} D_2$ requires $D_1 \underset{w}{\hat{\leq}} D_2$ by definition, and $D_2 \hat{\leq} D_3$ is syntactic sugar for $D_2 \underset{\infty}{\hat{\leq}} D_3$, so Proposition 2.23 is directly applicable to show the claim. □

It remains to show that we can decompose *every* statement of $D_1 \underset{w}{\hat{\leq}} D_2$ for arbitrary noise models attached to D_1, D_2, i.e. that the decomposition we will use is complete. We do this by creating an interim diagram D_1', that has exactly the undetectable effects W of D_1 with effect weight $\text{ewt}(W) \geq w$ removed. Interestingly, we can simply reuse the reduced signature normal form of Proposition 6.10: This form exactly enumerates the ewt of D_1 in fault gadgets and is by definition fault-equivalent to it. Then we simply remove all fault gadgets that have an annotation at or above w.

To formalise:

Proposition 6.12. The decomposition of w-fault boundedness into w-fault equivalence and fault boundedness is complete, i.e. for every two diagrams D_1, D_2 with respective noise models $\mathcal{F}_1, \mathcal{F}_2$ and $D_1 \underset{w}{\hat{\leq}} D_2$ there is a diagram D_3 with noise model $\mathcal{F}_3$ such that

$$D_1 \underset{w}{\hat{=}} D_3 \quad \text{and} \quad D_3 \hat{\leq} D_2.$$

Proof. Consider $D_{1,\mathcal{F}_1}$ as the implementation of $\mathcal{F}_1$ into D_1 using fault gadgets, and let $D^{\text{nf}}_{1,\mathcal{F}_1}$ be the rSN-form obtained from $D_{1,\mathcal{F}_1}$. Then let $D^{w}_{1,\mathcal{F}_1}$ be the diagram obtained from $D^{\text{nf}}_{1,\mathcal{F}_1}$ through removing all fault gadgets F annotated with $\text{ewt}(\text{eff}(F)) \geq w$. By construction, this implies

$$D \hat{=} D_{1,\mathcal{F}_1} \hat{=} D^{\text{nf}}_{1,\mathcal{F}_1} \underset{w}{\hat{=}} D^{w}_{1,\mathcal{F}_1}\,.$$

We also observe that we have obtained a diagram which is ∞-fault-bounded by D_2: The limitation to w is now enforced in the diagram $D^{w}_{1,\mathcal{F}_1}$ itself. If $D^{w}_{1,\mathcal{F}_1}$ was not ∞-fault-bounded by D_2, there would be some fault gadget F_{bad} with an effect that has no equivalent in $\mathcal{F}_2$. But since we removed every fault that has an effect greater or equal to w it must be that $\text{ewt}(\text{eff}(F_{bad})) < w$, which directly contradicts the precondition $D_1 \underset{w}{\hat{\leq}} D_2$ as F_{bad} must also be present in $D^{\text{nf}}_{1,\mathcal{F}_1} \hat{=} D_{1,\mathcal{F}_1} \hat{=} D_1$. Thus, $D^{w}_{1,\mathcal{F}_1}$ is the diagram we seek and it holds that

$$D_1 \underset{w}{\hat{=}} D^{w}_{1,\mathcal{F}_1} \hat{\leq} D_2\,.$$

□

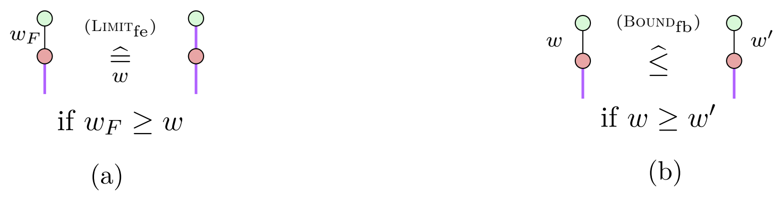


Figure 3: Axioms that in addition to Fig. 2 are required for a calculus that is complete for (a) w-fault equivalence and (b) fault boundedness.

We then provide the two axioms required for a calculus that is for w-fault equivalence and fault boundedness respectively, in Fig. 3. Of course, we have to prove that they are sound for their corresponding diagrammatic relationships as well.

Proposition 6.13. The axioms in Fig. 3 are sound w.r.t. w-fault equivalence and fault boundedness respectively.

Proof. (Limit$_{\mathsf{fe}}$) By Proposition 2.20, the RHS is fault-bounded by the LHS, so showing that $\mathsf{LHS} \underset{w}{\widehat{\leq}} \mathsf{RHS}$ remains. But the depicted noise model does not feature any non-trivial faults that have an effect weight strictly below weight w: The only non-trivial effect is that of the Z flip, which has effect $w_F \geq w$ by assumption. So the requirements of Proposition 3.12 are trivially fulfilled, so $\mathsf{LHS} \underset{w}{\widehat{\leq}} \mathsf{RHS}$ and thus $\mathsf{LHS} \underset{w}{\widehat{=}} \mathsf{RHS}$ holds.

(Bound$_{\mathsf{fb}}$) Via a similar argument to (Limit$_{\mathsf{fe}}$), the only non-trivial effect on both LHS and RHS is that of their annotated Z-flips Z_{LHS} and Z_{RHS}. The underlying diagram is the same, so similar to Corollary 3.14 it suffices to establish a direct correspondence between the fault effects.

Both Z-flips flip the same web since they are in the same position, and they are the only ones populating their respective effect classes so we have $\mathrm{wt}(Z_{\mathsf{LHS}}) = \mathrm{ewt}(\mathrm{eff}(Z_{\mathsf{LHS}}))$ and similar to the RHS. Then with the assumption $w \geq w'$ it holds that

$$\mathrm{ewt}(\mathrm{eff}(Z_{\mathsf{LHS}})) = \mathrm{wt}(Z_{\mathsf{LHS}}) = w \geq w' = \mathrm{wt}(Z_{\mathsf{RHS}}) = \mathrm{ewt}(\mathrm{eff}(Z_{\mathsf{RHS}})) \, ,$$

which in turn implies $\mathsf{LHS} \underset{\infty}{\widehat{\leq}} \mathsf{RHS}$ by Proposition 3.12, and thus by definition $\mathsf{LHS} \widehat{\leq} \mathsf{RHS}$.

□

The decomposition of w-fault boundedness is sound by Proposition 6.11, so we can further obtain a calculus that is sound for w-fault boundedness:

Corollary 6.14. The calculus obtained from adding both axioms in Fig. 3 to the axioms from fault equivalence in Fig. 2 is sound for w-fault boundedness.

Based on Theorem 3.18 we can prove completeness for w-fault equivalence and fault boundedness, and by combining them with Proposition 6.12, for w-fault boundedness:

Theorem 6.15 (Completeness for w-fault boundedness)**.** Assume the base axiomatisation in Fig. 2. Adding the two rules in Fig. 3 yields a calculus that is complete for w-fault equivalence and fault boundedness respectively. Adding both yields a calculus that is complete for w-fault boundedness.

Proof. We will show the claim for w-fault equivalence first. Let there be two diagrams D_1, D_2 with noise models $\mathcal{F}_1, \mathcal{F}_2$ such that $D_1 \underset{w}{\mathrel{\widehat{=}}} D_2$. Then obtain their implementation using fault gadgets as $D_{1,\mathcal{F}_1}, D_{2,\mathcal{F}_2}$ and the reduced signature normal forms as $D^{\text{nf}}_{1,\mathcal{F}_1}, D^{\text{nf}}_{2,\mathcal{F}_2}$ like in Theorem 3.18.

Using a similar strategy as in Proposition 6.12, we want to remove all fault gadgets representing effects with weight at or above w, yielding diagrams $D^{w}_{1,\mathcal{F}_1}, D^{w}_{2,\mathcal{F}_2}$. We can apply this rewrite diagrammatically using the rule $(\textsc{Limit}_{\text{fe}})$ from Fig. 3. Then we have

$$D^{\text{nf}}_{1,\mathcal{F}_1} \underset{w}{\mathrel{\widehat{=}}} D^{\text{nf}}_{2,\mathcal{F}_2} \qquad \Longleftrightarrow \qquad D^{w}_{1,\mathcal{F}_1} \mathrel{\widehat{=}} D^{w}_{2,\mathcal{F}_2}\,,$$

where the last property can be diagrammatically shown by Theorem 3.18, so adding $(\textsc{Limit}_{\text{fe}})$ yields a calculus that is complete for w-fault equivalence.

An analogous proof shows that instead adding $(\textsc{Bound}_{\text{fb}})$ from Fig. 3 results in a calculus complete for fault boundedness.

By Proposition 6.12, every statement of w-fault boundedness can be decomposed into a statement of w-fault equivalence and fault boundedness. So to obtain a calculus that is complete for w-fault boundedness, it suffices to add both rules from Fig. 3. □

7 Implementation

Although the ZX calculus is now proven sound and complete for fault equivalence and related notions, deriving a proof / counterexample for fault equivalence is still a manual process. The definition of fault equivalence admits a naïve approach to automation which is only suitable for the smallest circuits due to its high computational complexity. This prevents scaling the verification of fault equivalence to larger circuits and circuits with more involved noise models [RPK25].

However, the proof we give in Sections 3 to 6 for the completeness result is constructive throughout. From it, we may derive a more refined approach to automating the task of checking fault equivalence and implement it for usage in the wider research community, both in the context of the ZX calculus and fault-tolerant quantum computing. Due to the structure of the proof, namely going via a normal form that enumerates effects instead of faults, this refined algorithm admits an exponentially lower complexity than the naïve approach.

This section outlines the derivation of the algorithm. An implementation is provided as a ready-to-use Python package in [Rüs25], built on top of the `pyzx` package [KvdW20], and currently limited to equally weighted noise models. We compare the algorithm to the naïve approach derivable from the definition of fault equivalence and discuss some optimisations that may be done outside the ZX calculus. Finally, we describe how the intermediate normal forms may be used for efficiently computing additional properties of noise models beyond fault equivalence.

7.1 Automating Fault Equivalence Checks

The naïve implementation directly derivable from the definition consists of iterating through every fault in a diagram and finding some equivalent with lower weight by iterating through every possible fault in the other diagram. This can quickly become intractable: If we assume an edge flip noise model such that every edge $e \in E$ features all three axis flips X, Y, Z, the number of all possible computational faults in a diagram is $\left|\overline{\mathcal{P}^{|E|}}\right| = 4^{|E|}$, so the total number of operations would be roughly on the order of $\left(4^{|E|}\right)^2$. Even using a meet-in-the-middle approach where possible faults are precomputed for both diagrams and indexed in an access-efficient data-structure, this only reduces the time and space requirements to $4^{|E|}$. We thus propose an alternative implementation.

We recall that the completeness proof through fault effects, largely facilitated by Proposition 3.12, decouples finding all possible undetectable effects of a noise model along with their effect weight, and checking fault equivalence itself. Furthermore, finding undetectable effects can be decomposed into first finding the signatures of all atomic faults, enumerating all of their combinations, and subsequently normalising them, which was used in Section 6.

The implementation in [Rüs25] leverages a similar decomposition. Instead of computing signatures and directly starting enumeration, the signatures of atomic faults are normalised and deduplicated first (keeping the lowest weight value), ensuring that enumeration is not artificially inflated due to atomic signatures with the same effect. This is sound without further normalisation steps when choosing an appropriate canonical representative, i.e. when the set of all representatives is closed under multiplication. In our case, choosing the lexicographically smallest representative yields a set that admits this property.

Then, given two diagrams D_1, D_2 with noise models $\mathcal{F}_1, \mathcal{F}_2$, determining whether it holds that $D_1 \mathrel{\hat{=}} D_2$ w.r.t. $\mathcal{F}_1, \mathcal{F}_2$ is done as follows:

1. For both diagrams, compute detecting web generators via [Bor19, Alg. 3], obtain a flip operator collection and add sinks according to them.
2. Compute[9], normalise and deduplicate signatures for all atomic faults.
3. Disconnect sinks and subsequently eliminate detectable effects through Proposition 3.17.
4. Obtain all undetectable effects from combinations of existing undetectable effects.
5. Determine the effect from D_1 with the lowest effect weight w that has no boundary equivalent in D_2. If it exists, this directly implies $D_1 \underset{w}{\hat{\leq}} D_2$ and provides a counterexample to $D_1 \underset{w+1}{\hat{\leq}} D_2$. If it does not exist, $D_1 \underset{\infty}{\hat{\leq}} D_2$ holds.
6. Repeat the last step for the symmetric scenario to obtain a verdict for fault equivalence.

This strategy has direct implications for computational complexity. The algorithm described above first computes the effects of atomic faults in polynomial time, represented as normalised signatures. Signatures exist on the boundary and specify a subset of d flip operators from flipop. Representing the diagram as a state on n qubits we have n boundary edges, so there may be at most $\left|\overline{\mathcal{P}^n} \times \mathcal{P}(\text{flipop})\right| = 4^n 2^d$ different signatures. The normalisation procedure removes all differences that are due to stabilisers, so the total number of possible different effects is

$$\left|\left(\overline{\mathcal{P}^n} \times \mathcal{P}(\text{flipop})\right) / \mathcal{S}\right| = \frac{4^n 2^d}{2^n} = 2^{n+d}$$

Computing all undetectable effects from these atomic faults may thus result in up to 2^{n+d} combinations[10]. The number d of detecting web generators is clearly bounded by the number of edges, so the refined approach yields an exponential advantage over the naïve approach.

[9]The provided implementation computes signatures similar to how a Pauli web would be computed in Section 5.3, opening fault gadgets and sinks to obtain a Pauli web that highlights particular spawning edges green. An implementation based on the constructive proof of Proposition 5.4 is left for future work.

[10]Note that although the number of combinations may be dependent on d, there may be only 2^n *distinct* undetectable effects, so many combinations will be duplicated.

7.1.1 Further Performance Improvements

For almost all steps provided above, keeping derivations outside the ZX calculus provides further opportunities for performance improvements. We outline three such opportunities: Virtualisation of fault gadgets, compilation of signatures to integers, and combination of unfolding undetectable effects and finding the lowest weight unmatched effect.

The representation of diagrams with fault gadgets is based on one of the internal representations for ZX diagrams from the `pyzx` package [KvdW20]. To be precise, diagrams are taken to be undirected graphs with attached vertex types, where fault gadgets (including sinks) are kept 'virtual' as long as possible. That is, adding a fault gadget to the diagram does not change the diagram immediately, but simply adds an entry in an internal data structure. The representation exposes a function to instantiate all fault gadgets on a diagram, which is only done when absolutely necessary, e.g. when the diagram should be drawn with its noise model for analysis.

Keeping fault gadgets virtual removes from them any particular ordering on edges, making them commutative by default and thus respecting Proposition 4.9. Additionally, moving fault gadgets as in Section 5.3 is subject to a large boost in performance: We do not have to add (potentially multiple) individual spiders and edges to the diagram for targets on internal edges, only to subsequently remove them as those targets are merged.

Another improvement to performance is derived from handling signatures as vectors in $\mathbb{B}^{2n}$ (see Section 2.1). Multiplication up to global phase in this case becomes an XOR operation between these vectors, which is very fast on almost all modern computing resources. This is a common optimisation when handling members of a Pauli group, and is leveraged to a high degree e.g. in the package `stim` [Gid21b].

For the final improvement, we combine the steps of computing all combinations of effects and finding the lowest weight effect that has no equivalent. We sketch this approach only for the implemented variant, i.e. limited to equally weighted noise models. A generalisation to arbitrarily weighted noise models is left for future work.

We observe that if we have two undetectable faults F_1, F_2 on D_1, for which we have already found some equivalent faults F_1', F_2' with lower weight in D_2, we do not have to check their direct composite F_1F_2, since we could simply combine $F_1'F_2'$ as well to provide an equivalent. Thus, 'interesting' / unchecked undetectable faults may only be generated by combining detectable faults.

We then incrementally generate effects through iteration over detectable faults. In some iteration i, we generate all combinations of all detectable faults discovered in iteration $i-1$. Since all atomic faults are equally weighted with $\mathrm{awt}(\cdot) = 1$, we can take the weight of a generated fault to be equal to the iteration i. The weight of an effect is then the weight of the first fault with this effect that was generated, and since w-fault boundedness requires $(w-1)$-fault boundedness, the incremental procedure is sound. In the case that a counterexample to w-fault boundedness is found, i.e. some new undetectable effect is generated where no equivalent in D_2 was found yet, the entire procedure may return early and report that $(w-1)$-fault boundedness is the strongest guarantee that can be made.

A similar optimisation is applicable for combinations of existing undetectable effects, and both variants are implemented in [Rüs25].

7.2 Additional Properties

The notion of fault equivalence may be used to recover many standard definitions in the fields of quantum error correction and fault-tolerant quantum computing [RPK25]. One of the most direct results is the *distance* of a circuit, or more generally of a ZX diagram.

The distance of a diagram D under a noise model $\mathcal{F}$ is the lowest weight that $\mathcal{F}$ associates with an undetectable non-trivial fault [Got97], and it is useful for characterising which faults are not only detectable, but also *correctable* [DP23]. Using the framework of fault effects, the distance of D is equivalent to the lowest effect weight of some non-trivial effect in $\mathrm{Eff}_{\neq 0}(\mathcal{F})$. We recall that a fully idealised diagram only admits the trivial effect $\emptyset$, so following [RPK25, Prop. 3.6] we can directly describe the distance of D as statements about the fault equivalence of D and its fully idealised variant:

$$\text{distance of } D \text{ is } d \quad\Longleftrightarrow\quad \vdots\,\boxed{D}\,\vdots \;\underset{d}{\hat{=}}\; \vdots\,\boxed{D}\,\vdots \quad\text{and}\quad \vdots\,\boxed{D}\,\vdots \;\underset{d+1}{\not\hat{=}}\; \vdots\,\boxed{D}\,\vdots$$

After computing the reduced signature normal form, we can thus simply find the lowest weight effect from the enumeration with fault gadgets in linear time. However, we usually need not compute the entire rSN-form, since we can apply a similar incremental variant of determining fault boundedness as in Section 7.1.1.

Beyond these direct derivations from fault equivalence, the framework of signatures, effects, and the related normal forms from Sections 5 and 6, allows to reason about other properties of noise models. However, most of these properties are incompatible with adversarial reasoning. We recall that the essence of adversarial reasoning is that, as long as we preserve what we deem the 'worst case' for some property, we may destroy and create 'information' freely. Reasoning as in Propositions 3.15 to 3.17 (visualised as ($\textsc{Merge}_{\text{fe}}$), ($\textsc{Comb}_{\text{fe}}$) and ($\textsc{Detect}_{\text{fe}}$) in Fig. 2) preserves fault equivalence and related properties, but since we are able to introduce or remove atomic faults from the noise model such reasoning may not preserve other properties.

We note that up to and including the point of computation of all atomic signatures, the original noise model is still described in full, i.e. all atomic fault effects are preserved. It is only after the computation of atomic signatures, i.e. when modifying the first signature normal form in Section 6.3, that we apply adversarial reasoning. The two properties of noise models subsequently presented are best directly derived from such a full and non-adversarial description.

7.2.1 Detector Error Models

Fault detection in noisy Clifford quantum circuits often involves measurements of selected qubits, which may individually have their outcome flipped by faults [Got97]. These measurements may be organised into 'detecting sets' or 'detectors', which are groups of measurements whose parity of outcomes is predetermined in a noise-free setting [Der+24; RPK25]. Thus, if a fault were to occur such that odd many measurements in a detector are flipped, the parity deterministically changes and the fault becomes detectable. In such a scenario, a fault is said to have *violated* that detector.

We can now relate faults and detectors by analysing which faults would violate which detectors. This is captured in the aptly named notion of a *detector error model* [Der+24], where a popular example is the *Tanner graph* [Tan81; Loe04]. A Tanner graph is a bipartite graph that consists of vertices of two types, one representing (usually atomic) faults, the other representing detectors. Two vertices (F, d) are connected if and only if F violates d.

Tanner graphs find applications in configuring decoders, which are essential parts of an error corrected quantum circuit and, given a vector measurement outcomes, try to infer which fault occurred [Der+24]. If they are successful, that fault may be corrected; this constitutes an important cornerstone for fault-tolerant quantum computation. In a Tanner graph, one may derive whether the selection for detectors is appropriate to facilitate the decoding task: If every fault violates a unique set of detectors, the fault itself can be identified with a particular violation at runtime and subsequently corrected [DP23; Got22]. Thus, the sparsity of the Tanner graph is typically directly connected with the theoretical performance of the associated decoder. We must note that analytically finding the best Tanner graph for a given circuit is an NP-hard problem [Der+24], so good detectors are often found empirically via fast simulations, e.g. using the `stim` package [Gid21b].

We briefly return to ZX diagrams, where detectors may be identified directly with detecting regions [RPK25]. A fault may flip the sign of a detecting region, causing the entire diagram to collapse to zero. From a simulation perspective, this is interpreted as reducing the probability of obtaining the usual (fault-free) deterministic outcome of a detector to zero, ensuring that the detector is flipped which is observable.

In the framework of signatures and effects, we recall that a signature contains a description of the detecting regions that a fault flips, which is equivalent to the detectors it would violate. Thus, we can directly construct the Tanner graph for a circuit from the first signature normal form we obtain before applying adversarial reasoning.

However, signatures support a richer notion: In addition to detector violations, we also have information on the potential influence of a fault on the outputs of the diagram. Pauli operators are unitary and self-adjoint, i.e. they square to the identity, so if a Pauli operator P describes the action of a fault on the boundary we may simply use P to correct the fault. Further, we can collapse multiple equivalent faults using normalisation from Section 6.4, which does not require adversarial reasoning. If we include X, Z

'outcome' nodes in the Tanner graph and connect faults according to their signatures, we obtain what we call an *extended Tanner graph* that is in bijection to the collection of signatures we obtained for our atomic faults.

We conclude by noting that the information contained in an extended Tanner graph may be obtained through other venues. Most prominently, once a fault is fully identified by a decoder, efficient stabiliser simulation [AG04] may be used to obtain the required corrections. Signatures describe those simulation results already, and obtaining them is unified for both circuits and the more general ZX diagrams. Thus, the findings in this section effectively lifted the notion of a Tanner graph and the procedure of finding it to arbitrary Clifford ZX diagrams.

7.2.2 Logical Error Rate

Quantum error correction codes, including those arising from the stabiliser formalism, encode the space of some qubits, called *logical* qubits, onto a larger space of underlying *physical* qubits [Got97]. However, outcomes of operations on the logical space cannot be measured directly. Instead, the outcomes of subsequent physical measurements are aggregated to determine the value of the logical operator, yielding an 'observable' [Der+24]. In the presence of faults, the outcome of this observable may at times be flipped.

Beyond correcting faults, it is useful to determine just whether a fault occurred or not, i.e. detecting it, and accurately predicting whether the aforementioned observable was flipped. When an observable was flipped as the result of a fault, but the prediction is that it was not flipped, we say that a *logical error* occurred. Simulating how often such logical errors occur, i.e. sampling the circuits noise model and determining how often corrections and the subsequent flip predictions fail, yields the logical error rate [Gid21b]. A low logical error rate is directly in correspondence with good performance of a decoder in practice.

The implementation of such simulations in the `stim` package is currently limited to circuits. However, sampling from a noise model does not necessarily require a circuit. In particular, we may directly use effects obtained from normalising the first signature normal form as a basis for sampling: A faults effect encodes the flipped detectors and the faults influence on the outputs of the diagram, from we can derive whether the observable was flipped. Thus, we could sample from the range of atomic fault effects, forming the composite effect via Proposition 3.4 and applying the prediction algorithm provided in the surrounding setting. A demo for sampling (without subsequent prediction) is included in the Python package from [Rüs25].

Beyond lifting Tanner graphs and empirically determining the logical error rate to the general realm of Clifford ZX diagrams, the framework of signatures and effects may enable a simplified or even more computationally efficient derivation of many other properties from noise models. We leave such insights to future work.

8 Future Work

This work opens up a variety of venues for future work. As outlined in [RPK25], the perspective of fault equivalence may be able to describe many different concepts in the fields of quantum error correction and fault-tolerant quantum computing. With our completeness result, we pave the way for verifying fault tolerance automatically and scale. Additionally, the framework of signatures and effects is highly versatile: Some introductory examples of its applications were provided in Section 7.

We move on to briefly sketch three concrete directions for future work. First, the implementation described in Section 7 does not yet implement interfaces with high usability, and further may be subject to many optimisations inside those implementations. Second, we discuss how the results of this work may be used with more general noise models, in particular those that are either not adversarial or provide continuous rather than discrete notions of likelihood as an extension to the brief discussion in Section 7.2. Third, we initiate a generalisation of this work beyond two-level systems into systems with higher dimensionality.

8.1 Extensions and Optimisations

The implementation as provided in [Rüs25] and described in Section 7 currently only supports equally weighted noise models. Thus, a direct extension would provide support for weighted noise models, which requires adaptation of the optimisations outlined in Section 7.1.1.

Beyond this, the implementation may be subject to additional performance improvements. For example, instead of calculating web generators at the very beginning of signature computation, we could apply local adversarial reasoning to collapse some effect classes first. Depending on the noise model in use and the reasoning that is performed, lowering the number of signatures that need to be obtained through e.g. Proposition 5.4 may provide an overall benefit to performance. Note that this does not necessitate that the subsequently derived properties are invariant under adversarial reasoning. When collapsing k faults into a single class before signature computation, we can provide k copies of the resulting signature and assign the original weights to restore a non-adversarial description of the original noise model.

This may be particularly useful when dealing with edge flip noise models, since given a spider s, one may immediately collapse all faults on its legs of the same colour as s onto a single leg by reasoning with the (Fusion) rule. Y-flips may be decomposed into individual X, Z flips which are processed separately, as long as we retain the original weight of the Y-flip and produce a combined signature accordingly. Such a change drops the number of faults to be handled roughly by the average degree of connectivity of the diagram. More sophisticated and thus usually more computationally expensive approaches could start grouping faults across entire networks of spiders with the same colour. Finally, this change facilitates partitioning the diagram, allowing multiple threads to operate on the diagram at the same time.

Another approach might be to improve the computation of web generators themselves. The algorithm from [Bor19, Alg. 3] requires solving a round of Gaussian elimination over a matrix scaling with the number of spiders inside the diagram. For particularly large diagrams this becomes intractable, even though, e.g. for deep diagrams with a low number of qubits, subsequent derivation of fault equivalence might be fast. Thus, future efforts might study whether Pauli webs can be obtained incrementally or at least with improved scaling, and whether obtaining them can be done in synergy with incrementally computing signatures.

A focus on potential improvements during computation of signatures is projected to provide the most benefits, since these have direct impacts on properties beyond just fault equivalence. Beyond signatures, implementations for individual properties may be further optimised, e.g. one might derive a more efficient iteration technique for determining fault equivalence that improves on that from Section 7.1.1. It remains to be determined whether such changes actually provide worthwhile benefits in scenarios that are deemed realistic.

8.2 Beyond Adversarial Weighted Noise Models

As we outlined in Section 2.3, a possibility future work is considering more general noise models and whether we can obtain a completeness result for fault equivalence with respect to them. We explore both notions of non-adversariality and continuous likelihood separately.

8.2.1 Non-Adversarial Reasoning

Requiring information beyond the worst case, i.e. only considering the minimum weight of a fault in a single effect class, prevents direct usage of Propositions 3.15, 3.17 and 4.12. The main benefit of adversariality is that it allows to construct 'local' rewrites. To be precise, the rewrites can have a reduced scope and make fewer to none assumptions about their surroundings, as long as they preserve the worst case, regardless of whether they actually handle the worst case when applied or not.

This simplification is unavailable when reasoning non-adversarially. Thus, going beyond adversarial reasoning might require non-local rewrites, which are difficult to apply in distributed settings and thus difficult to generalise to parallel computing.

8.2.2 Continuous Likelihood Models

We could replace the weight annotation on fault gadgets with a more general annotation, that allows continuous values like probabilities, instead of discrete values from $\mathbb{N}_{\neq 0}$. As long as we continue to use annotations, this might provide a direct completeness result for these noise models.

However, these annotations may not be directly seen as syntactic sugar anymore, since Lemma 4.5 is not easily generalisable for fractional values. Thus, unweighted or even weighted edge flip noise might not be universal for all noise models in such a setting, requiring adaptation of Theorem 4.6.

8.3 Beyond the Qubit ZX Calculus

Beyond the qubit ZX calculus, recent developments have resulted in similar calculi for arbitrary but fixed d-dimensional systems, referred to as the qudit ZX calculus [Ran14; Wan22], or even mixed finite integer dimensions, referred to as the 'finite-dimensional ZX calculus'. Both are provably complete in a general setting [PSW25]. However, our results are restricted to the Clifford fragment. For this fragment, we in particular require both universality and completeness which has so far only been derived for e.g. prime-dimensional qudit calculi [BC22; Poó+23], so we focus on such 'qupit' systems.

We reason that the construction of fault gadgets must admit a generalisation to qupit calculi. In particular, a fault gadget consists of Pauli boxes, which are in turn characterised by a Clifford operation C that sends a single Z Pauli operator to the Pauli operator corresponding to the box in question. Using the universality result for stabiliser qupit calculi [BC22], there must thus be an implementation in the calculus for this Clifford operation C.

The resulting fault gadgets can return to using weight annotations only for Z flips, as the complexity of higher-dimensional rotations is handled by the Pauli boxes. Using completeness of these calculi, we should be able to recover most of the properties listed in Section 4.2 for such more general fault gadgets. Open considerations however include the precise handling of Pauli webs in such systems, thus we leave a full and precise generalisation for future work.

9 Conclusion

In this work, we proved that there exists an axiomatisation of a ZX calculus that is sound and complete for showing fault equivalence between diagrams, and we further motivated and supplied the individual elements of this axiomatisation. To achieve this, we extended the framework around the notion of fault equivalence and derived notions like fault boundedness with the framework of fault signatures and fault effects, with which we were able to restate the aforementioned notions such that they are easier to verify. We showed how this framework can be implemented and used diagrammatically and how many intuitive properties of noise models are reflected when shown inside a diagram.

At the centre of the diagrammatic implementation we found fault gadgets, a tool thus far used as a static tracker for potential faults. We showed that fault gadgets are indeed capable of guided movement through a diagram, giving them characteristics of a dynamic tracker. Further, we showed that fault gadgets may be used for multiple purposes, including harnessing other fault gadgets as in the case of sinks.

Even though we introduced and formalised more general noise models, i.e. those that can assign arbitrarily assign weights to faults, we showed using fault gadgets that there is a direct diagrammatic interpretation such that it suffices to describe a noise model that only allows Z axis flips on particular edges, all with the same weight. This recovers the usability of noise models that are commonly used in fault-tolerant quantum computing, and provides a notion of universality for them.

As a direct derivation from the completeness result, we introduced an algorithm for checking fault equivalence with an exponential improvement in computational complexity when compared to the naïve approach derivable from the definition of fault equivalence. We discuss how this algorithm was implemented in a ready-to-use Python package, and what optimisations were applied to improve tractability of the problem. Additionally, we reasoned that the framework of faults and effects may be used to recover additional properties of noise models and related notions, such as decoders and logical error rate, which are contained in the implementation as well. This lifts such properties to being easily defined on general Clifford ZX diagrams, which facilitates an increased interaction between the work on circuits and the work on ZX diagrams when modeling noise. Finally, we provided multiple extension points for future work, including improvements to the implementation and generalising the completeness result for more expressive noise models and higher-dimensional ZX calculus variants.

Looking forward, we anticipate that the results of this work aid in identifying novel fault-tolerant implementations of existing quantum programs, as well as verifying the correctness of existing implementations. In particular, we project the derived software package to find applications in handling larger diagrams, where manual reasoning becomes cumbersome or even intractable, and fully automated testing that may lead to automated discovery of fault-tolerant implementations. Therefore, this work provides and extends a valuable formal foundation that facilitates manual as well as automated reasoning about noise, fault equivalence, and ultimately fault tolerance.

References

[AG04] Scott Aaronson and Daniel Gottesman. “Improved simulation of stabilizer circuits”. In: *Phys. Rev. A* 70 (5 Nov. 2004), p. 052328. DOI: 10.1103/PhysRevA.70.052328. URL: https://link.aps.org/doi/10.1103/PhysRevA.70.052328.

[Bac+17] Dave Bacon et al. “Sparse Quantum Codes From Quantum Circuits”. In: *IEEE Transactions on Information Theory* 63.4 (2017), pp. 2464–2479. DOI: 10.1109/TIT.2017.2663199.

[Bac14] Miriam Backens. “The ZX-calculus Is Complete for Stabilizer Quantum Mechanics”. In: *New Journal of Physics* 16.9 (Sept. 17, 2014), p. 093021. ISSN: 1367-2630. DOI: 10.1088/1367-2630/16/9/093021. arXiv: 1307.7025 [quant-ph]. URL: http://arxiv.org/abs/1307.7025v1.

[BC22] Robert I. Booth and Titouan Carette. “Complete ZX-Calculi for the Stabiliser Fragment in Odd Prime Dimensions”. In: *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*. Ed. by Stefan Szeider, Robert Ganian, and Alexandra Silva. Vol. 241. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 24:1–24:15. ISBN: 978-3-95977-256-3. DOI: 10.4230/LIPIcs.MFCS.2022.24. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.MFCS.2022.24.

[BKW22] Niel de Beaudrap, Aleks Kissinger, and John van de Wetering. “Circuit Extraction for ZX-Diagrams Can Be #P-Hard”. en. 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.

[Bom+24] Hector Bombin et al. “Unifying Flavors of Fault Tolerance with the ZX Calculus”. In: *Quantum* 8 (June 18, 2024), p. 1379. ISSN: 2521-327X. DOI: 10.22331/q-2024-06-18-1379. arXiv: 2303.08829 [quant-ph]. URL: http://arxiv.org/abs/2303.08829v3.

[Bor19] Coen Borghans. “ZX-calculus and Quantum Stabilizer Theory”. MA thesis. Radboud University, 2019. URL: https://www.cs.ox.ac.uk/people/aleks.kissinger/papers/borghans-thesis.pdf (visited on 08/06/2025).

[BPW17] Miriam Backens, Simon Perdrix, and Quanlong Wang. “A Simplified Stabilizer ZX-calculus”. In: *Electronic Proceedings in Theoretical Computer Science* 236 (Jan. 2017), pp. 1–20. ISSN: 2075-2180. DOI: 10.4204/eptcs.236.1. URL: http://dx.doi.org/10.4204/EPTCS.236.1.

[CB18] Christopher Chamberland and Michael E. Beverland. “Flag fault-tolerant error correction with arbitrary distance codes”. In: *Quantum* 2 (Feb. 2018), p. 53. ISSN: 2521-327X. DOI: 10.22331/q-2018-02-08-53. URL: https://doi.org/10.22331/q-2018-02-08-53.

[CD11] Bob Coecke and Ross Duncan. “Interacting Quantum Observables: Categorical Algebra and Diagrammatics”. In: *New Journal of Physics* 13.4 (Apr. 14, 2011), p. 043016. ISSN: 1367-2630. DOI: 10.1088/1367-2630/13/4/043016. arXiv: 0906.4725 [quant-ph]. URL: http://arxiv.org/abs/0906.4725v3.

[Der+24] Peter-Jan H. S. Derks et al. *Designing Fault-Tolerant Circuits Using Detector Error Models*. Dec. 25, 2024. DOI: 10.48550/arXiv.2407.13826. arXiv: 2407.13826 [quant-ph]. URL: http://arxiv.org/abs/2407.13826v2. Pre-published.

[DP09] Ross Duncan and Simon Perdrix. “Graph States and the Necessity of Euler Decomposition”. In: *Proceedings of the 5th Conference on Computability in Europe: Mathematical Theory and Computational Practice*. CiE ’09. Heidelberg, Germany: Springer-Verlag, 2009, pp. 167–177. ISBN: 9783642030727. DOI: 10.1007/978-3-642-03073-4_18. URL: https://doi.org/10.1007/978-3-642-03073-4_18.

[DP23] Nicolas Delfosse and Adam Paetznick. *Spacetime codes of Clifford circuits*. 2023. arXiv: 2304.05943 [quant-ph]. URL: https://arxiv.org/abs/2304.05943v2.

[Gid21a] Craig Gidney. *Stim: A fast stabilizer circuit library. Main README*. Sept. 2021. URL: https://github.com/quantumlib/Stim/blob/main/README.md (visited on 08/07/2025).

[Gid21b] 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.

[Got09] Daniel Gottesman. *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation*. Apr. 16, 2009. DOI: 10.48550/arXiv.0904.2557. arXiv: 0904.2557 [quant-ph]. URL: http://arxiv.org/abs/0904.2557v1. Pre-published.

[Got22] Daniel Gottesman. *Opportunities and Challenges in Fault-Tolerant Quantum Computation*. 2022. arXiv: 2210.15844 [quant-ph]. URL: https://arxiv.org/abs/2210.15844v1.

[Got97] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. 1997. arXiv: quant-ph/9705052 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9705052v1.

[Got98] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. 1998. arXiv: quant-ph/9807006 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9807006v1.

[Hua+23a] Eric Huang et al. “Tailoring Three-Dimensional Topological Codes for Biased Noise”. In: *PRX Quantum* 4.3 (Sept. 2023). ISSN: 2691-3399. DOI: 10.1103/prxquantum.4.030338. URL: http://dx.doi.org/10.1103/PRXQuantum.4.030338.

[Hua+23b] Jiaxin Huang et al. “Graphical CSS Code Transformation Using ZX Calculus”. In: *Electronic Proceedings in Theoretical Computer Science* 384 (Aug. 2023), pp. 1–19. ISSN: 2075-2180. DOI: 10.4204/eptcs.384.1. URL: http://dx.doi.org/10.4204/EPTCS.384.1.

[KvdW20] Aleks Kissinger and John van de Wetering. “PyZX: Large Scale Automated Diagrammatic Reasoning”. In: *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*. Ed. by Bob Coecke and Matthew Leifer. Vol. 318. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2020, pp. 229–241. DOI: 10.4204/EPTCS.318.14.

[KvdW24] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-calculus and Quantum Compilation*. Preprint, 2024.

[Leu+97] Debbie W. Leung et al. “Approximate quantum error correction can lead to better codes”. In: *Physical Review A* 56.4 (Oct. 1997), pp. 2567–2573. ISSN: 1094-1622. DOI: 10.1103/physreva.56.2567. URL: http://dx.doi.org/10.1103/PhysRevA.56.2567.

[Loe04] H.-A. Loeliger. “An introduction to factor graphs”. In: *IEEE Signal Processing Magazine* 21.1 (2004), pp. 28–41. DOI: 10.1109/MSP.2004.1267047.

[MBG23] 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: https://doi.org/10.22331/q-2023-11-07-1172.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press, 2010.

[NW17] Kang Ng and Quanlong Wang. “A universal completion of the ZX-calculus”. In: (June 2017). DOI: 10.48550/arXiv.1706.09877. URL: https://arxiv.org/abs/1706.09877v1.

[Poó+23] Boldizsár Poór et al. “The Qupit Stabiliser ZX-travaganza: Simplified Axioms, Normal Forms and Graph-Theoretic Simplification”. In: (2023). arXiv: 2306.05204 [quant-ph]. URL: https://arxiv.org/abs/2306.05204v2.

[Pre15] John Preskill. *Quantum computation lecture notes*. 2015. URL: https://www.preskill.caltech.edu/ph219/ (visited on 08/07/2025).

[PSW25] Boldizsár Poór, Razin A. Shaikh, and Quanlong Wang. “ZX-calculus is Complete for Finite-Dimensional Hilbert Spaces”. In: *Electronic Proceedings in Theoretical Computer Science* 426 (Aug. 2025), pp. 127–158. ISSN: 2075-2180. DOI: 10.4204/eptcs.426.5. URL: http://dx.doi.org/10.4204/EPTCS.426.5.

[Ran14] André Ranchin. “Depicting qudit quantum mechanics and mutually unbiased qudit theories”. In: *Electronic Proceedings in Theoretical Computer Science* 172 (Dec. 2014), pp. 68–91. ISSN: 2075-2180. DOI: 10.4204/eptcs.172.6. URL: http://dx.doi.org/10.4204/EPTCS.172.6.

[RPK24] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Floquetifying Stabiliser Codes with Distance-Preserving Rewrites*. Dec. 16, 2024. DOI: 10.48550/arXiv.2410.17240. arXiv: 2410.17240 [quant-ph]. URL: http://arxiv.org/abs/2410.17240v2. Pre-published.

[RPK25] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. *Fault Tolerance by Construction*. 2025. arXiv: 2506.17181 [quant-ph]. URL: https://arxiv.org/abs/2506.17181v2.

[Rüs25] Maximilian Rüsch. *Project code for this work*. 2025. URL: https://github.com/maximilianruesch/fault-gadgets.

[SZ14] Christian Schröder de Witt and Vladimir Zamdzhiev. "The ZX-calculus is incomplete for quantum mechanics". In: *Electronic Proceedings in Theoretical Computer Science* 172 (Dec. 2014), pp. 285–292. ISSN: 2075-2180. DOI: 10.4204/eptcs.172.20. URL: http://dx.doi.org/10.4204/EPTCS.172.20.

[Tan81] R. Tanner. "A recursive approach to low complexity codes". In: *IEEE Transactions on Information Theory* 27.5 (1981), pp. 533–547. DOI: 10.1109/TIT.1981.1056404.

[TFK23] Alex Townsend-Teague, Julio Magdalena de la Fuente, and Markus Kesselring. "Floquetifying the Colour Code". In: *Electronic Proceedings in Theoretical Computer Science* 384 (Aug. 30, 2023), pp. 265–303. ISSN: 2075-2180. DOI: 10.4204/EPTCS.384.14. arXiv: 2307.11136 [quant-ph]. URL: http://arxiv.org/abs/2307.11136v2.

[Tuc20] David Kingsley Tuckett. "Tailoring surface codes: Improvements in quantum error correction with biased noise". (qecsim: https://github.com/qecsim/qecsim). PhD thesis. University of Sydney, 2020. DOI: 10.25910/x8xw-9077.

[vdWet20] John van de Wetering. *ZX-calculus for the working quantum computer scientist*. 2020. arXiv: 2012.13966 [quant-ph]. URL: https://arxiv.org/abs/2012.13966v1.

[Vil19] Renaud Vilmart. "A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2019, pp. 1–10. DOI: 10.1109/LICS.2019.8785765.

[Wan22] Quanlong Wang. *Qufinite ZX-calculus: a unified framework of qudit ZX-calculi*. 2022. arXiv: 2104.06429 [quant-ph]. URL: https://arxiv.org/abs/2104.06429v5.

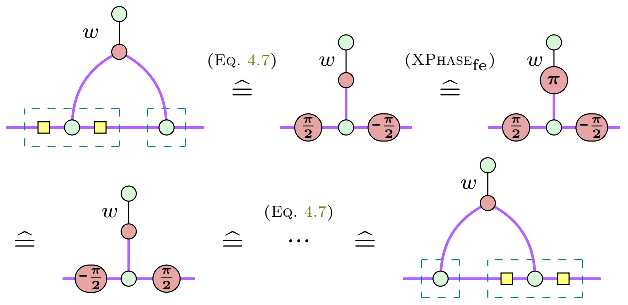
Appendices

A Missing Proofs for Fault Gadgets

Proposition 4.9 (Gadgets commute)**.** Let $D_{\mathcal{F}}$ be a gadget-idealised ZX diagram based on the underlying diagram D. For a single edge e in D, targets of fault gadgets from $D_{\mathcal{F}}$ on e may be arbitrarily and fault-equivalently rearranged.

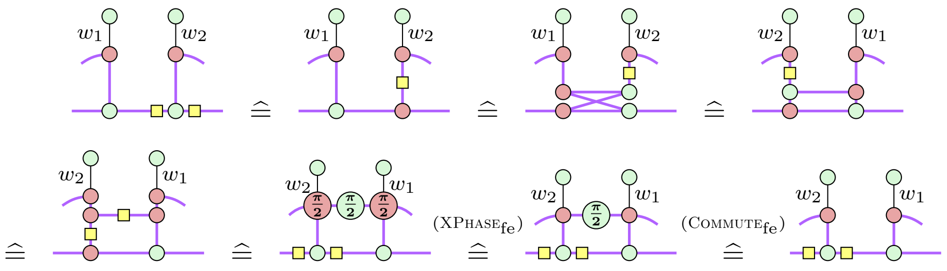
Proof. We need to show that every pair of Pauli boxes attached to fault gadgets commutes. If the Pauli boxes are of the same type, they trivially commute, similar to the process in Eq. (4.3). The order in which we consider the boxes does not matter, so we have three remaining cases ZX, ZY and XY.

If the boxes are attached to the same gadget, we can get that ZX commutes by composing to Y using Eq. (4.7), applying a π-phase copy and decomposing in a mirrored manner:



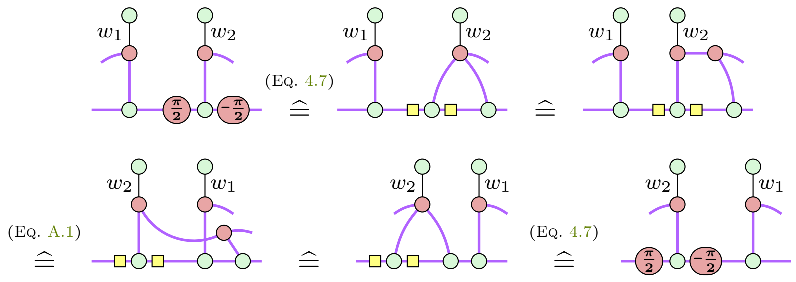
The remaining two cases follow directly, since Y boxes may be decomposed into X and Z boxes, which commute, and compose to Y on the other side again. So it remains to show the three cases where the boxes are attached to different gadgets.

We first show using the set of axioms from Fig. 2 that Z and X Pauli boxes commute:

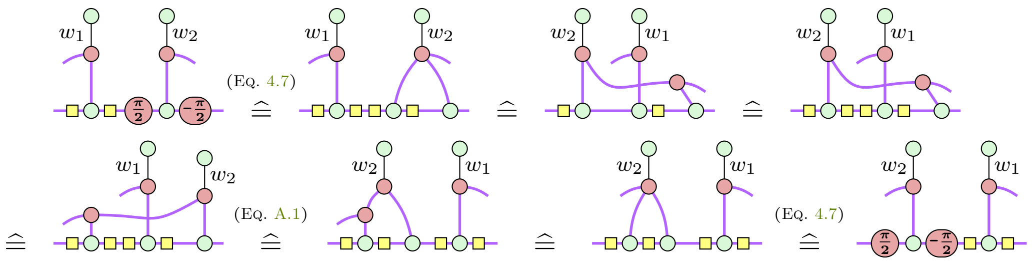


(A.1)

Finally, we employ the decomposition of Y boxes into X, Z boxes from Eq. (4.7) to show the remaining two cases:

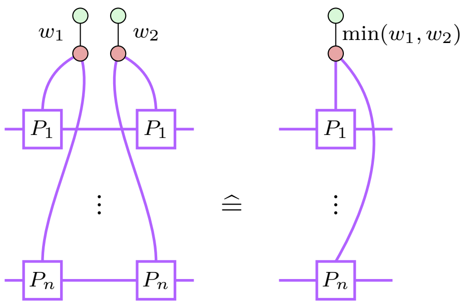


and for X and Y:

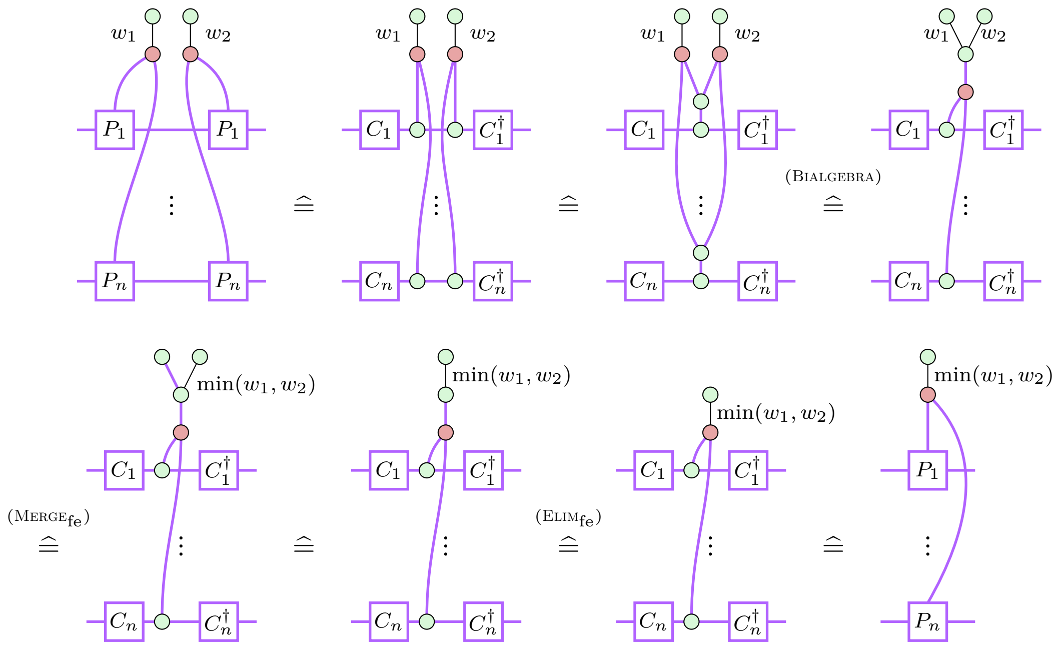


□

Proposition 4.11. Let $D_{\mathcal{F}}$ be a ZX diagram in GI-form, and let $F_1 \equiv F_2$ be congruent faults in $\mathcal{F}$. Then the fault gadgets for F_1, F_2 have the same targets on the same edges, and it holds that:

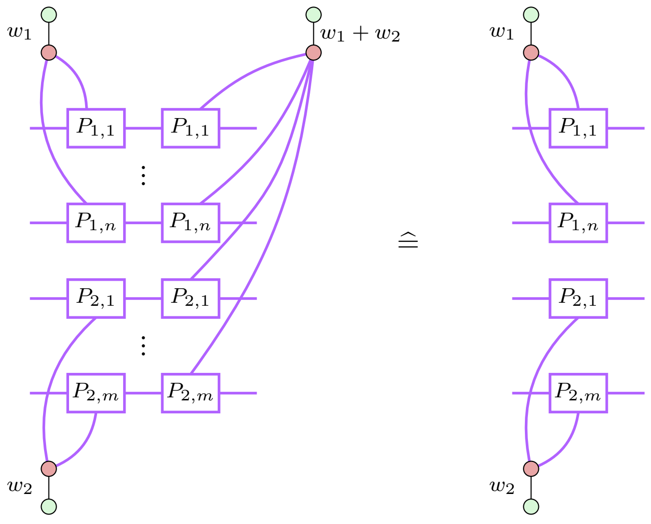


Proof.

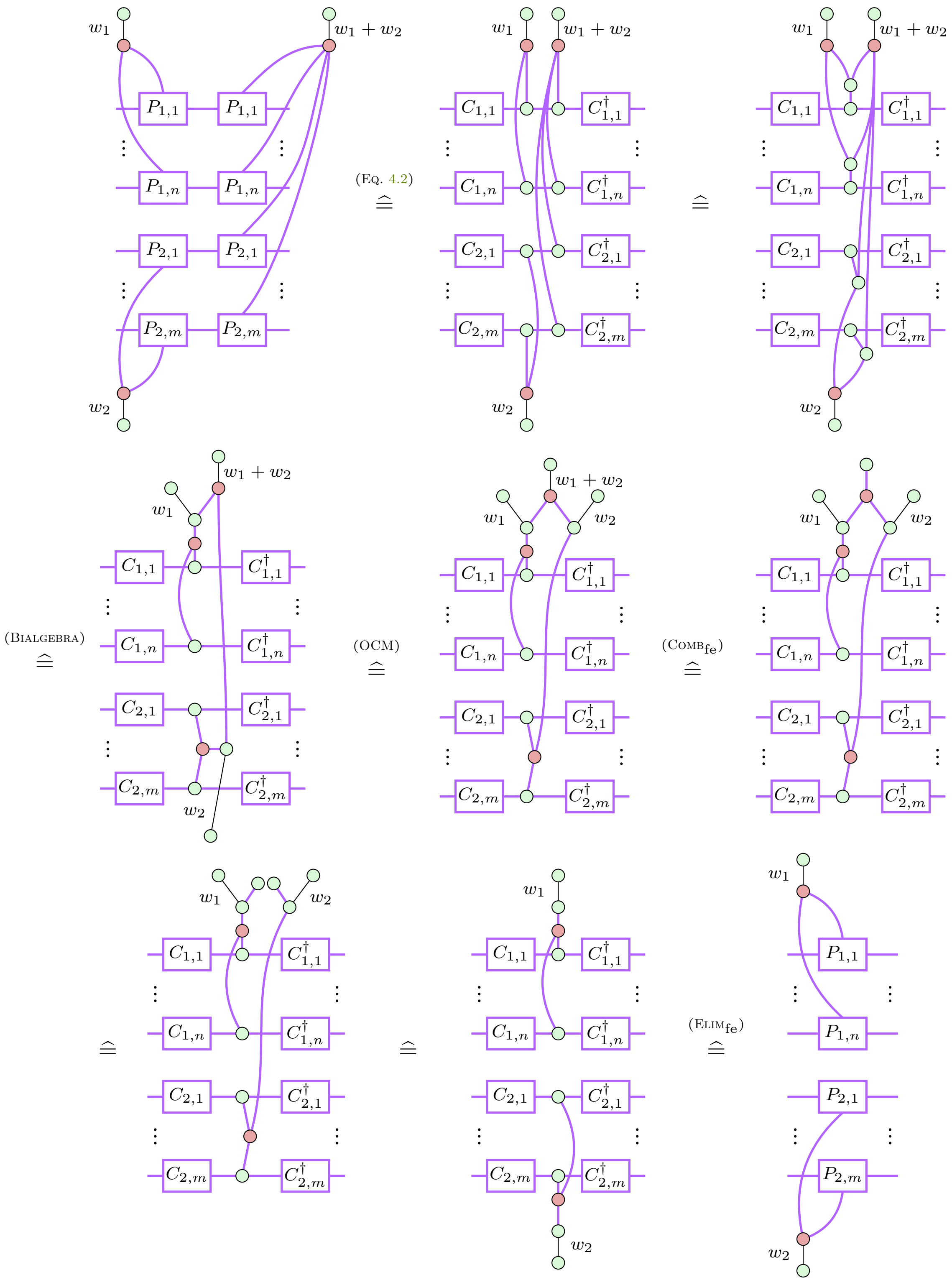


□

Proposition 4.12. Let $D_{\mathcal{F}}$ be a ZX diagram in GI-form, and let F_1, F_2 be arbitrary faults in $\mathcal{F}$, then it holds that:

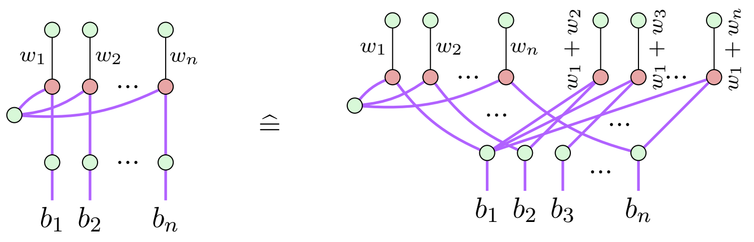


Proof.

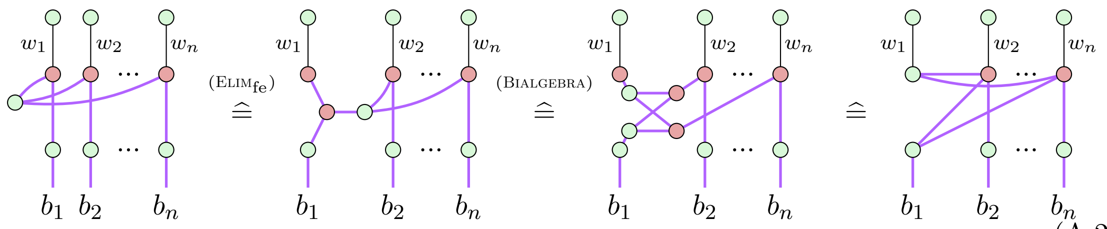

w_1
$w_1 + w_2$
w_2
$P_{1,1}$
$P_{1,n}$
$P_{2,1}$
$P_{2,m}$
$C_{1,1}$
$C_{1,n}$
$C_{2,1}$
$C_{2,m}$
$C^\dagger_{1,1}$
$C^\dagger_{1,n}$
$C^\dagger_{2,1}$
$C^\dagger_{2,m}$
(Eq. 4.2) $\widehat{=}$
$\widehat{=}$
(Bialgebra) $\widehat{=}$
(OCM) $\widehat{=}$
(Comb$_{\text{fe}}$) $\widehat{=}$
$\widehat{=}$
$\widehat{=}$
(Elim$_{\text{fe}}$) $\widehat{=}$

□

Proposition 4.13. For any $w_1, \ldots, w_n$ it holds that:

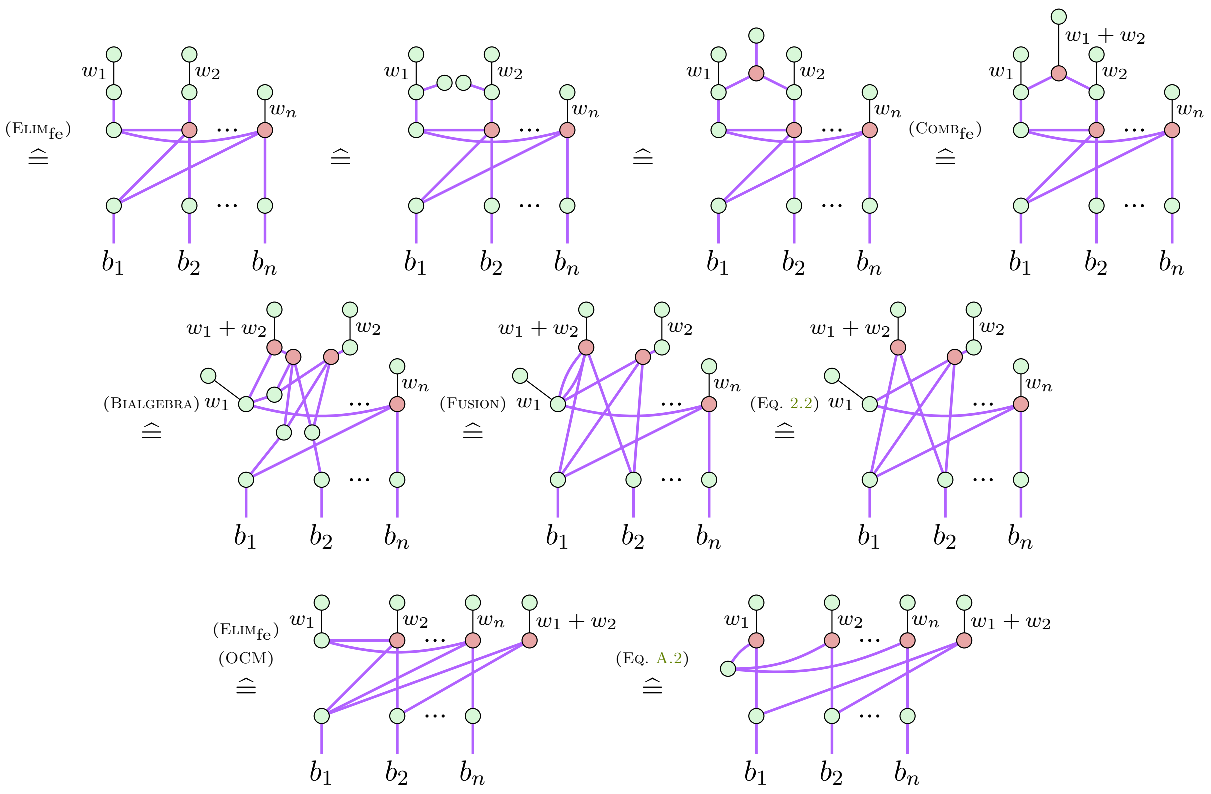


Proof. First, observe that we can bring the diagram into a form where we deem the fault with w_1 as 'active for combination':



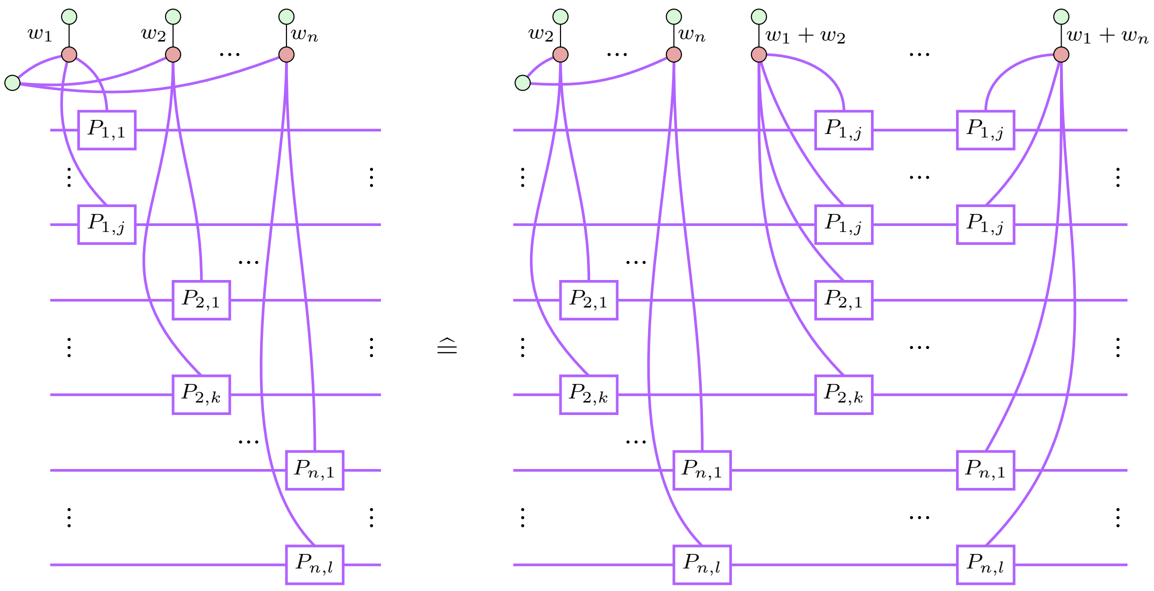
(A.2)

After this activation, we apply the rule (Comb$_{\text{fe}}$) to generate the composite fault for w_1, w_2 and use Eq. (A.2) to 'deactivate' w_1 again:

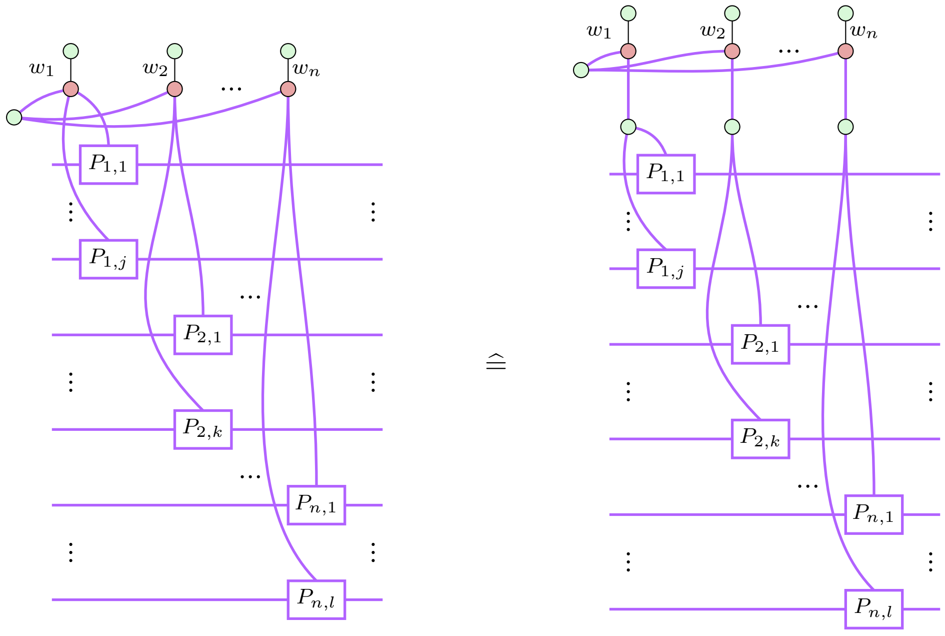


So we have recovered the original diagram with a composite fault attached. This procedure can clearly be iterated to generate composite faults of w_1 with $w_3, \ldots, w_n$, and thus yields the claim. □

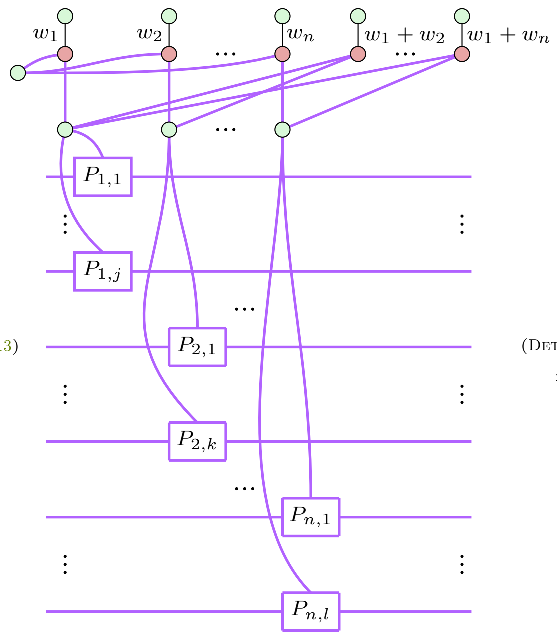
Proposition 4.14. For any $w_1, \dots, w_n$ it holds that:



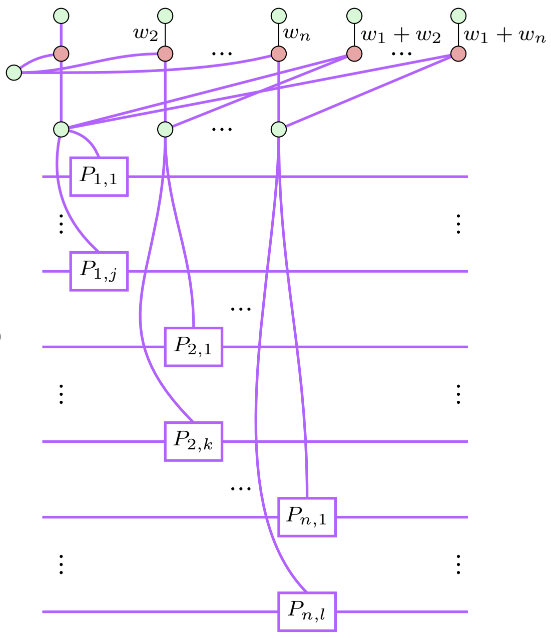
Proof.



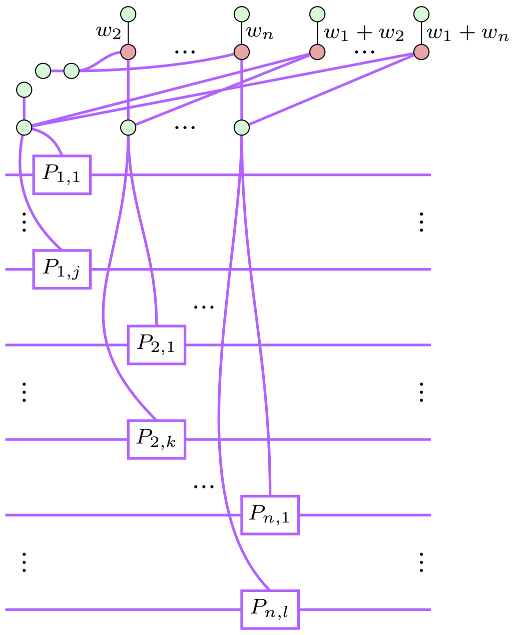
$\overset{(\text{Prop. } 4.13)}{\triangleq}$



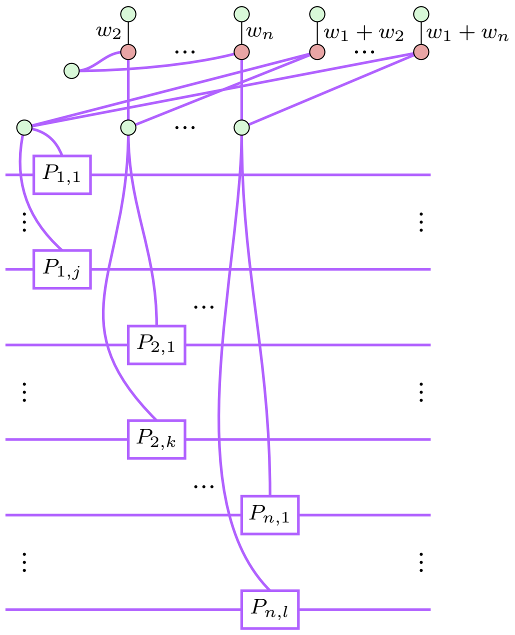
$\overset{(\text{Detect}_{\text{fe}})}{\triangleq}$



$\triangleq$



$\triangleq$



At this point, the spawning edges connected to the same Pauli boxes can be seperated into distinct fault gadgets similar to the proof of Proposition 4.12, finishing the claim. □