Optimising Clifford Circuits with Quantomatic

Andrew Fagan
andrew.fagan.2014@uni.strath.ac.uk

Ross Duncan
ross.duncan@strath.ac.uk

Department of Computer and Information Sciences
University of Strathclyde
26 Richmond Street, Glasgow, United Kingdom

We present a system of equations between Clifford circuits, all derivable in the ZX-calculus, and formalised as rewrite rules in the Quantomatic proof assistant. By combining these rules with some non-trivial simplification procedures defined in the Quantomatic tactic language, we demonstrate the use of Quantomatic as a circuit optimisation tool. We prove that the system always reduces Clifford circuits of one or two qubits to their minimal form, and give numerical results demonstrating its performance on larger Clifford circuits.

1 Introduction

Remarkable advances in the past two years have seen quantum computing hardware reach the point where the deployment of quantum devices for non-trivial tasks is now a near-term prospect. However, these machines still suffer from severe limitations, both in terms of memory size and the coherence time of their qubits. It is therefore of paramount importance to extract the most useful work from the fewest operations: a poorly optimised quantum program may not be able to finish before it is undone by noise.

In this paper we study the automated optimisation of *Clifford circuits*. Clifford circuits are not universal for quantum computation – they are well known to be efficiently simulable by a classical computer [2] – however adding any non-Clifford gate to the Cliffords yields a set of approximately universal operations hence it is likely that the vast majority of operations in any quantum program will be Clifford operations, and hence reducing the Clifford depth and gate count of a circuit will have substantial benefit.

A secondary reason to focus on Cliffords is that the corresponding subtheory of quantum mechanics is well-understood in terms of the ZX-calculus [7]. Backens [3] has shown that the ZX-calculus is sound and complete for *stabilizer quantum theory* – that is the fragment of quantum mechanics containing only the Clifford operations and states which can be produced from them. While recent extensions to the ZX-calculus have been proposed which are complete for the Clifford+T fragment [15] and for the full qubit theory [21], both these extensions are significantly larger and more complex than the stabilizer subtheory, and both axiomatisations are undergoing rapid development. Clifford circuits therefore provide a stable platform to develop techniques which could later be extended to a universal language.

We present a circuit optimiser which has been developed as a *simplification procedure* in the graphical proof-assistant Quantomatic [16, 17]. By working inside the proof-assistant we obtain an important benefit: alongside the optimised circuit our optimiser produces a formal proof certifying the correctness of its circuit transformations.

Quantomatic is a flexible graph rewrite engine which is especially well-adapted to working with the ZX-calculus. It is designed for interactive use, and a typical session involves interleaving

Submitted to:
QPL 2018

automated tactics[1], backtracking, and manually choosing rewrites. However, our circuit optimiser is designed to operate without user intervention, and we have necessarily made extensive use of the system's powerful tactic combinators and its built-in Python interpreter. In doing so, we have constructed the largest and most sophisticated Quantomatic development to date.

As a graph rewriting engine, Quantomatic normally inspects only the local subgraph structure of a term while searching for a rewrite to apply. However, quantum circuits have a global causal structure which must be maintained to preserve the property of "being a circuit". A key contribution of this work is the development of techniques to infer the global structure and use it to guide the rewrite process.

Selinger [23] presented generators and relations for all the n-qubit Clifford groups as a rewrite system over string diagrams. This work has some similarities to that: most notably, we also treat Clifford circuits as a free PROP subject to an equivalence relation defined on small subgraphs. However Selinger's approach is based on transforming circuits to a standard form which is often larger than the smallest equivalent circuit, whereas we aim for minimal forms. Selinger's rewrite system has a much larger number of rules than the axioms of the ZX-calculus. Maslov and Roetteler [20] demonstrate that all Clifford unitaries can be implemented in depth no more than 7 stages deep, using the $\wedge Z$ as the only two-qubit gate. The $\wedge Z$ has formal disadvantages in ZX-calculus, so in our development we have used the CNOT and swap gates instead. In the examples we have tested, our optimiser usually attains similar depth over our gate set; however on some examples it halts while some simplifications are still possible. Quantomatic has previously been used to produce mechanised correctness proofs for quantum error correcting codes [13, 6, 10], and many of the techniques used in these works are applied here.

We assume that the reader is at least somewhat familiar with both the ZX-calculus and the Quantomatic system; an accessible introduction to both is found in the earlier formalisation effort [13].

Acknowledgements The authors wish to thank Aleks Kissinger and Hector Miller-Bakewell for their technical assistance throughout this project.

The Quantomatic project files All the proofs which appear in this paper and its appendix are publicly available as a downloadable Quantomatic project at `https://gitlab.cis.strath.ac.uk/kwb13215/Clifford-Quanto/`.

2 The Clifford Group

Let $\omega = e^{i\pi/4}$. The nth Clifford group is the group of unitary matrices acting on $\mathbb{C}^{2^n}$, finitely generated by the matrices

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad V = \frac{1}{\sqrt{2}} \begin{pmatrix} \omega & \bar{\omega} \\ \bar{\omega} & \omega \end{pmatrix} \quad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \sigma = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

under tensor product and matrix composition. Observe that S and V are both order 4, the Pauli matrices are given by $Z = S^2$ and $X = V^2$, and the usual Hadamard matrix is obtained

[1] A *tactic* is a short program which automates a particular proof strategy.

as $H = \bar{\omega}SVS$. Throughout this paper we will quotient the group by global scalar factors, so that $A = zA$ for all non-zero $z \in \mathbb{C}$. The resulting quotient yields the nth *reduced* Clifford group, which we denote $\mathcal{C}_n$.

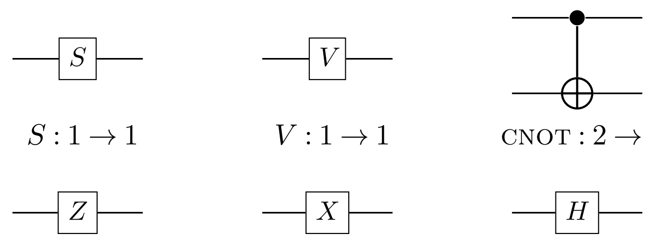
$\mathcal{C}_n$ is finite for all n and the order of the group is given by the expression[2]:

$$|\mathcal{C}_n| \;=\; \prod_{j=1}^{n} 2(4^j - 1)4^j \;=\; 2^{2n+1}(2^{2n}-1)\,|\mathcal{C}_{n-1}|\,.$$

The order of $\mathcal{C}_n$ grows very quickly: $|\mathcal{C}_1| = 24$, $|\mathcal{C}_2| = 11520$, $|\mathcal{C}_3| = 92827280$, and so on.

The Clifford group [14] is the *normalizer* of the Pauli group. Let $\mathcal{P}_n$ be the subgroup of $\mathcal{C}_n$ generated by the Z and X operators[3]. Then for all $C \in \mathcal{C}_n$ and all $P \in \mathcal{P}_n$ we have $CP = P'C$ for some $P' \in \mathcal{P}_n$. This crucial fact will heavily exploited in our circuit optimisation procedure.

A more holistic view of the Clifford group is as a class of *circuit diagrams* whose gates are the generators of the group. This can be formalised as a free †-PROP [19, 18] generated by morphisms



$S : 1 \to 1$ $V : 1 \to 1$ CNOT $: 2 \to 2$

$Z : 1 \to 1$ $X : 1 \to 1$ $H : 1 \to 1$

We refer to this PROP as **Cliff**. Note that the swap σ is automatically present due to the PROP structure. The morphisms Z, X, and H are not essential, but it will be convenient later to include them among the generators. The matrix valuations above suffice to define a standard interpretation functor $[\![\cdot]\!]_C : \mathbf{Cliff} \to \mathbf{fdHilb}$, such that $[\![\mathbf{Cliff}(n,n)]\!]_C = \mathcal{C}_n$. We will use this PROP only to define the translation from **Cliff** to the ZX-calculus as a functor; we refer the interested reader to Selinger [23] for more details of this perspective.

Remark 2.1. Note that **Cliff**, as a free PROP, does not include any non-trivial equations between Clifford circuits and is not therefore a presentation of the Clifford *group* itself: **Cliff** distinguishes between different implementations of the same unitary map, which are identified in the image of $[\![\cdot]\!]_C$. In particular $\mathbf{Cliff}(n,n)$ is infinite for all $n > 0$.

3 The ZX-calculus

The ZX-calculus [7] is a formal graphical notation for representing quantum states and processes, and an equational theory for reasoning about them. It is *universal*, meaning that every linear map has a corresponding ZX-calculus term, and *sound*, meaning that any equation derivable in the calculus is true in its standard Hilbert space interpretation.

Definition 3.1. A *term*, or *diagram*, of the ZX-calculus is a finite undirected open graph whose boundary is partitioned into *inputs* and *outputs*, and whose interior vertices are of the following types:

[2]The only scalars which arise in the quotient are ω^k for $k \in \mathbb{N}$ so the usual Clifford group is eight times larger.
[3]Under our quotient we have $Y = ZX = XZ$ so these generators suffice.

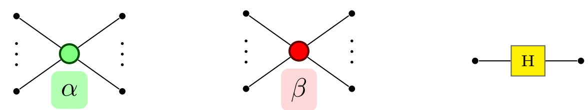


Figure 1: Allowed interior vertices

- $Z(\alpha)$ vertices, labelled by an angle α where $0 \leq \alpha < 2\pi$. These are depicted as green or light grey circles; if $\alpha = 0$ then the label is omitted.
- $X(\beta)$ vertices, labelled by an angle β, where $0 \leq \beta < 2\pi$. These are are depicted as red or dark grey circles; again, if $\beta = 0$ then the label is omitted.
- H vertices; unlike the other types H vertices are constrained to have degree exactly 2. They are depicted as yellow squares.

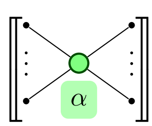
Two terms are considered equal if they are isomorphic as framed labelled graphs.

The allowed vertex types are shown in Figure 1. We adopt the convention that inputs are on the left, and outputs on the right.

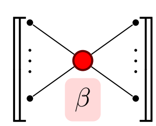
Compound terms may be formed by joining some number (maybe zero) of the outputs of one term to the inputs of another. Given a diagram $D : n \to m$ we define its *adjoint* $D^\dagger : m \to n$ to be the diagram obtained by reflecting the diagram around the vertical axis and negating all the angles. Thus the terms of the ZX-calculus naturally form a a †-PROP [8, 9], just like the Clifford circuits of the previous section. The three types of single vertex shown in Figure 1 can then be seen as the generators of this PROP, which we call **ZX**.

Remark 3.2. We consider the *scalar-free* fragment of the ZX-calculus; or equivalently, we quotient everything by non-zero scalar factors, just as we did in the previous section. This greatly reduces the complexity of the diagrams and poses no danger; Backens has demonstrated how to modify the calculus to preserve equality while respecting scalars [4].

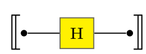
Definition 3.3. Given a *zx*-term $D : m \to n$, its *standard interpretation* is a linear map $[\![D]\!] : (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes n}$ defined on single vertices as follows:



$$= \begin{cases} |0\rangle^{\otimes m} & \mapsto \quad |0\rangle^{\otimes n} \\ |1\rangle^{\otimes m} & \mapsto \quad e^{i\alpha}|1\rangle^{\otimes n} \end{cases}$$



$$= \begin{cases} |+\rangle^{\otimes m} & \mapsto \quad |+\rangle^{\otimes n} \\ |-\rangle^{\otimes m} & \mapsto \quad e^{i\beta}|-\rangle^{\otimes n} \end{cases}$$



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

Since the above diagrams are the generators of **ZX**, the above definition extends uniquely to a strict †-symmetric monoidal functor $[\![\cdot]\!] : \mathbf{ZX} \to \mathbf{fdHilb}$.

The ZX-calculus has a rich equational theory based on the theory of Frobenius-Hopf algebras [7, 9]. Various axiomatisations have been proposed ([11, 12, 5, 22, 15, 21]) with various advantages and drawbacks. Here we adopt the scheme of Backens [3] which is clean, concise, and adequate for the treatment of the Clifford group. These are shown in Figure 2. Note that, due to the H-commute rule, the colour swapped versions of all the rules are admissible, and we shall use these colour swapped versions without further comment.

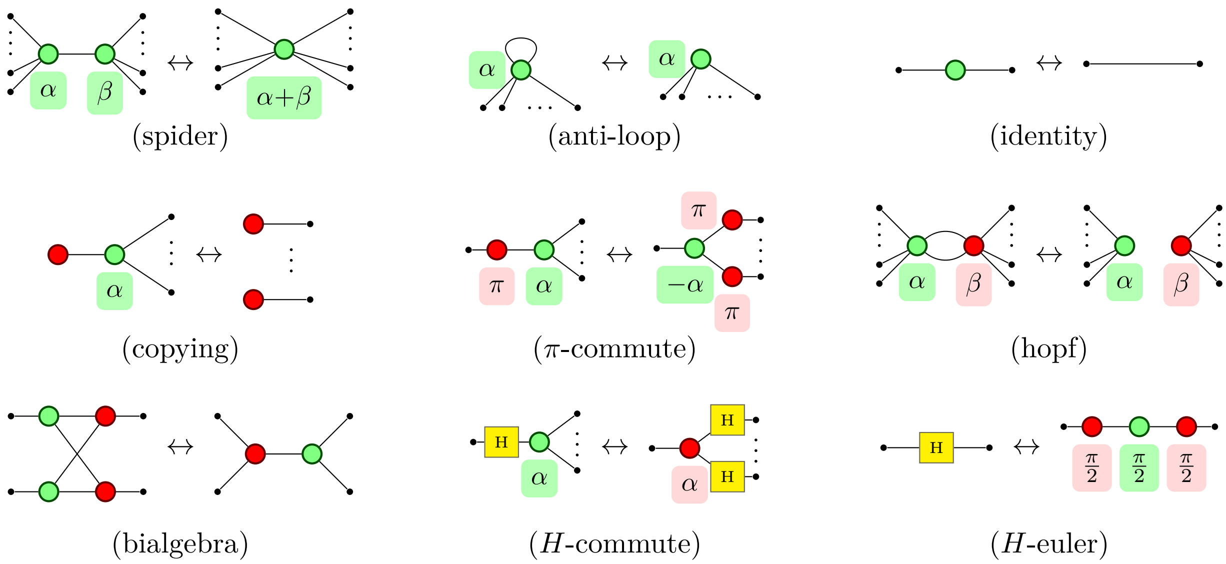


Figure 2: The rules of the ZX-calculus. Note all arithmetic is modulo 2π.

Definition 3.4. Let $\leftrightarrow$ be the one-step rewrite relation on the terms of **ZX** generated by the pairs of terms shown in Figure 2; let $\overset{*}{\leftrightarrow}$ be the least equivalence relation containing $\leftrightarrow$. We say that **ZX** terms a and b are *equivalent* when $a \overset{*}{\leftrightarrow} b$.

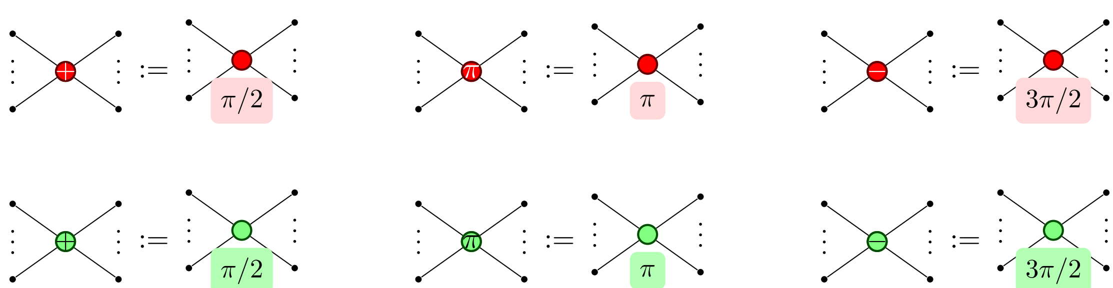
We reserve the notation $a = b$ for the case where a and b are equal as graphs, in the sense of Definition 3.1.

4 Representing Clifford Circuits in the ZX-calculus

For the rest of the paper we will use only the *stabilizer* sub-language of the full ZX-calculus.

Definition 4.1. Let $\mathbf{ZX_s}$ denote the sub-PROP of **ZX** obtained by restricting to those terms whose Z and X vertices are labelled only by angles $\alpha \in \{0, \pi/2, \pi, 3\pi/2\}$.

Since only four vertex labels can occur in the terms of $\mathbf{ZX_s}$, we will adopt a more compact notation as shown in below.



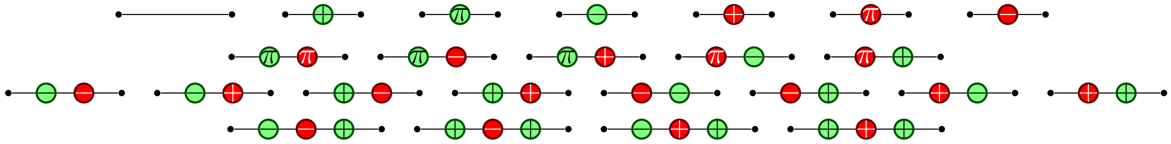
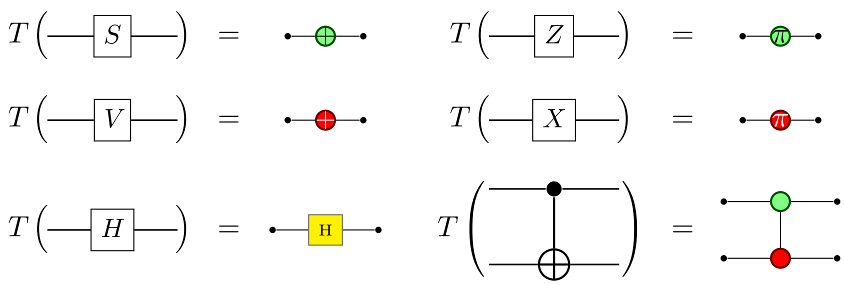


Figure 3: $\mathfrak{C}_1$: standard minimal forms for single-qubit Cliffords

We define the *translation* functor $T : \textbf{Cliff} \to \mathbf{ZX_s}$ on the generators as shown below.



This extends to the whole PROP in the usual way. Further we have the following:

Proposition 4.2. *For all $c : n \to m$ in* **Cliff** *we have $[\![c]\!]_C = [\![T(c)]\!]$.*

It is immediate from the rules of Figure 2 that if $a \overset{*}{\leftrightarrow} b$ then a is in $\mathbf{ZX_s}$ if and only if b is. Further, the equational theory of $\mathbf{ZX_s}$ is sound and complete for its standard interpretation.

Theorem 4.3 (Backens [3])**.** *Let a and b be terms of $\mathbf{ZX_s}$. Then $a \overset{*}{\leftrightarrow} b$ if and only if $[\![a]\!] = [\![b]\!]$.*

This theorem guarantees that if a given Clifford circuit has a smaller equivalent circuit then there is a proof of their equivalence in the ZX-calculus. The challenge is to find it. We start by considering minimal forms for $\mathcal{C}_1$ and $\mathcal{C}_2$.

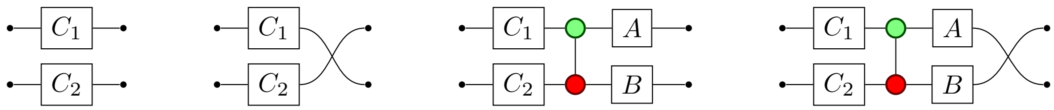
4.1 1-Qubit Cliffords

Let $\mathfrak{C}_1$ denote the 24 diagrams shown Figure 3. Note that all of them are in the image of the translation functor T, so they correspond to Clifford circuits. Further, each of the diagrams has a distinct interpretation under $[\![\cdot]\!]$, so they cover $\mathcal{C}_1$ and, by soundness, they are not equivalent.

Proposition 4.4. *Given $t \in \textbf{Cliff}(1,1)$ then $T(t) \overset{*}{\leftrightarrow} c$ for some $c \in \mathfrak{C}_1$.*

Proof. From its type, we have $[\![t]\!]_C \in \mathcal{C}_1$ and since $|\mathcal{C}_1| = 24$, necessarily $[\![t]\!]_C = [\![c]\!]$ for some $c \in \mathfrak{C}_1$, and thus by Proposition 4.2 $[\![T(t)]\!] = [\![c]\!]$. From this point the result follows by Theorem 4.3, however since we will require an effective procedure later, we offer a constructive proof which produces the required c.

Suppose the vertices of $T(t)$ are only of one colour; then $T(t)$ reduces by the spider rules to a graph with at most one vertex, hence it is in $\mathfrak{C}_1$. Otherwise, by the same reasoning, $T(t) \overset{*}{\leftrightarrow} t_1$ where t_1 has vertices of alternating colour, with none labelled by zero. If t_1 contains a Pauli it can commute with its neighbour; if t_1 contains more than two vertices, then two vertices of the same colour are now adjacent, so the Pauli can be eliminated by the spider rule. Hence, either t_1

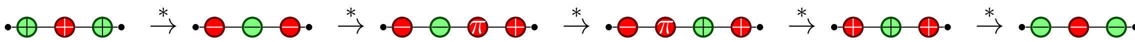


where $C_1, C_2 \in \mathfrak{C}_1$, $A \in \mathfrak{A}$ and $B \in \mathfrak{B}$.

Figure 4: $\mathfrak{C}_2$: standard minimal forms for two-qubit Cliffords

---

can be rewritten to t_2 containing no Pauli, or t_1 has only two vertices. In this second case, up to commuting Paulis, $t_1 \in \mathfrak{C}_1$. Now consider the diagram h shown below; we have rewrites:



By similar reasoning we can partition the 16 possible 3-vertex candidates for t_2 into four equivalence classes, each having a representative in $\mathfrak{C}_1$. Finally, suppose that t_2 has more than three vertices; then it contains a three vertex subsequence, and by the above any such subdiagram can rewrite to one with the opposite colouring, and via the spider rule $t_2 \overset{*}{\leftrightarrow} t_3$ where t_3 has fewer vertices. Hence, by induction, t can always rewrite to an element of $\mathfrak{C}_1$. $\square$

By enumerating all diagrams with fewer than four vertices, it's easy to check that no $c \in \mathfrak{C}_1$ is equivalent to a smaller diagram, so these diagrams are the minimal forms for $\mathcal{C}_1$.

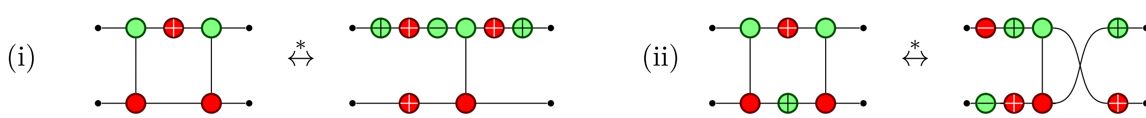
4.2 2-Qubit Cliffords

Let $\mathfrak{C}_2$ be the set of diagrams defined in Figure 4. Notice that $\mathfrak{C}_2$ has $24^2(1+1+9+9) = 11520$ elements. As in the case of $\mathfrak{C}_1$, all the elements of $\mathfrak{C}_2$ are evidently circuits, and it can be mechanically checked that they are all distinct under the standard interpretation. Hence we adopt $\mathfrak{C}_2$ as the normal forms for $\mathcal{C}_2$. Again, by the completeness of the axioms (Thm 4.3) we can immediately conclude that every 2-qubit Clifford circuit is equivalent to some $k \in \mathfrak{C}_2$. Zooming in a little closer we have the following:

Proposition 4.5. *Let $k \in \mathfrak{C}_2$ and let g be a generator of $\mathcal{C}_2$ as a* ZX*-calculus term; then $g \circ k \overset{*}{\leftrightarrow} k'$ for some $k' \in \mathfrak{C}_2$.*

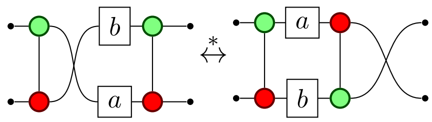
Proof. Notice that the 1-qubit Cliffords C_1 and C_2 which occur in k play no part in the computation, hence we ignore them. Further, for any $u \in \mathfrak{C}_1$, we have $(u \otimes \mathrm{id}) \circ \textsc{cnot} = (a \otimes \mathrm{id}) \circ \textsc{cnot} \circ (u_1 \otimes u_2)$ for some $u_1, u_2 \in \mathfrak{C}_1$ and $a \in \mathfrak{A}$, either by commuting vertices of the same colour, or taking out factors of Paulis. Similarly $(\mathrm{id} \otimes u) \circ \textsc{cnot} = (\mathrm{id} \otimes b) \circ \textsc{cnot} \circ (u_1 \otimes u_2)$ for some $b \in \mathfrak{B}$. Hence the only non-trival cases to consider are:

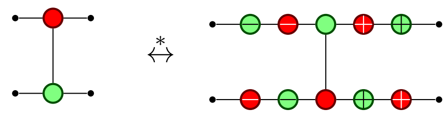
1. $g = \textsc{cnot}$ and $k = (a \otimes b) \circ \textsc{cnot}$. There are nine possibilities for this case, of which we consider two here. If $a = x_+$ and $b = \mathrm{id}$, then by using the bialgebra rule we reduce as shown in (i) below. If $a = x_+$ and $b = z_+$, then a similar argument yields (ii).



They remaining cases can all be derived from these two, except the case where $a = b = \mathrm{id}$, which is trivial.

2. $g = \textsc{cnot}$ and $k = (b \otimes a) \circ \sigma \circ \textsc{cnot}$. In this case we have (iii) below. However, by exploiting

(iii) 

(iv) 

the H rules we have the equivalence (iv) so this case can be reduced to the previous one.

3. $g = \textsc{cnot}$ and $k = \sigma$. Again we can apply (iv).

□

As in the single qubit case these reductions will provide the basis for the rewrite rules of our optimiser.

5 Quantomatic

We briefly sketch the Quantomatic system; for a full description see [17]; to obtain it, see [16].

Quantomatic is an interactive theorem prover which can prove equations between terms of the ZX-calculus. The user draws the desired term in the graphical editor, and builds the proof by applying *rewrite rules* to the current graph. A rewrite rule is a *directed* equation, i.e. a pair of graphs which have the same boundary. Rules are applied in two-step process. First the system searches for subgraphs of the term which are isomorphic to the LHS of the rule; each such subgraph is called a *match*. Quantomatic displays all the possible matches of the chosen rule in the current term, and allows the user to select where the rule is to be applied. The matched subgraph is then replaced by the RHS of the rule to produce a new term. A proof therefore consists of a sequence of terms linked by the application of a particular rule at a particular location in the term.

Quantomatic allows the user to define automated proof tactics, known as simplification procedures or *simprocs*. The name is slightly misleading since there is no need for a simproc to actually *simplify* the graph, in the sense of reducing the number of vertices or edges; however failure to do so will typically result in non-termination. Simprocs are written in the Python programming language, interpreted by the built-in Jython interpreter [1], and augmented with various tactic combinators. The most useful of these are:

- REWRITE(r) : given a rule r, apply the rewrite to the first match found.
- REWRITE_METRIC(r, m) : given a rule r and a *metric* function $m : \mathbf{ZX} \to \mathbb{N}$ apply the rewrite to the first match found which *reduces* the metric.
- REWRITE_TARGETED(r, v, t) ; given a rule r, a vertex v in the LHS of the rule, and a *targeting* function $t : \mathbf{ZX} \to \mathrm{Vert}$ apply the rewrite r to the first match where vertex v of the rule is matched to the vertex $t(G)$ in the term.

All of these combinators can also accept a list of rules, in which case the rewrites are attempted in the order of the list. Simprocs can be combined in sequence, and also using the combinator REDUCE(s) which repeats the simproc s until no new rewrites can be performed.

An important point is that when multiple matches are obtained, the system will select one without user intervention. Which rewrite is selected depends on the internal representation of the term, and is effectively non-deterministic.

The axioms of the ZX-calculus, considered as a rewrite system, are neither terminating nor confluent: for example, if $G \to G'$ via the π-commute rule then the rule can be applied again immediately to rewrite $G' \to G$. This and similar cases can easily lead to a non-terminating rewrite strategy. Further although many of the axioms can be oriented in a direction which reduces the graph complexity, the resulting rewrite system is no longer complete with respect to the standard interpretation: in some cases it is necessary to increase the graph size to derive a desired equation.

6 Optimising Clifford Circuits with Quantomatic

In this section we describe our optimisation procedure and summarise the results. The reasons for our choices are discussed in the following section.

By *circuit* we mean a ZX-calculus term which is the image of the translation functor $[\![\cdot]\!] : \mathbf{Cliff} \to \mathbf{ZX}$ defined in Section 4. Note that since **Cliff** is a †-PROP and $[\![\cdot]\!]$ is a strict monoidal functor, the circuits form a sub-PROP of **ZX**: they are closed under composition and tensor product; however they are not closed with respect to the equivalence relation $\overset{*}{\leftrightarrow}$.

Most ZX-calculus terms are *not* circuits: some of these don't correspond to Clifford unitaries, and so do not concern us, however there are many non-circuits which are equivalent to circuits via rewriting. We don't attempt to find circuit forms for such terms. Rather, knowing that the input is a circuit, we use rewrites which replace circuits with circuits, and thus stay at all times within the class of circuits.

The axioms of the ZX-calculus calculus are mostly equations between graphs which are evidently not circuits. Therefore our first step is to derive new equations between circuits which will serve as rewrites for use in the main simproc. These rules are listed in Appendix B.

While the derived rules are "obviously" equations between circuits, the circuit structure of a larger graph may not be apparent by looking at a local subgraph. It's necessary to look at the entire graph in order to guide the rewrite engine.

Definition 6.1. A ZX-calculus term is said to be *circuit-like* if (i) it has the same number of inputs and outputs (ii) its maximum vertex degree is 3, and all its degree 3 vertices have zero phase (iii) it is a simple graph (iv) it has a vertex-disjoint path cover with the following properties:

- Each path p starts at an input and ends at an output.
- If a pair of vertices are connected but covered by different paths, then they have different colours.
- If we direct all the edges in the path cover according to the path direction, and all edges not in the path are directed from green to red, then the resulting digraph is acyclic.

Proposition 6.2. *A* **ZX** *term is circuit-like if and only if it is a circuit.*

Intuitively the path cover specifies which edges of the **ZX** term are the "qubits" of the circuit; those edges not in the path cover define a CNOT acting on two qubits. Assuming that the graph is a circuit, the path cover is relatively easy to compute by starting at each input and finding a successor among its neighbours. If there is ever a choice of successor for a given vertex this choice can be resolved by looking at the other wires. This information is needed frequently by our optimisation procedure. If an unambiguous path cover cannot be constructed then the procedure halts with an error – in this case the input graph was not a circuit.

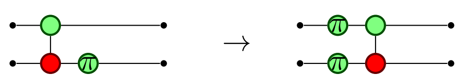
Our optimiser can be summarised as the repeated alternation of two phases:

- Simplification: apply everywhere possible a strictly reducing set of rules.
- Commutation: using targeting and metric rewrites, apply reversible rules to (i) move Pauli operators closer to the inputs. (ii) move CNOT and TONC gates which act on the same two qubits closed together.

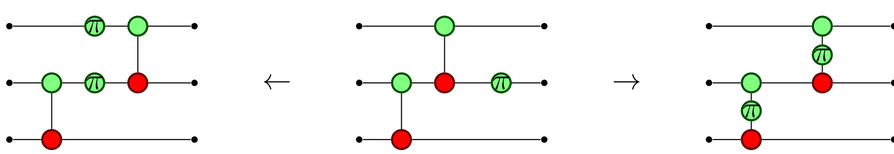
In order to reduce the number of rules, there is an initial phase which replaces all Hadamard, $Z(-\frac{\pi}{2})$ and $X(-\frac{\pi}{2})$ gates with S, V and Paulis as appropriate. The single qubit gates are then "tidied up" by a final pass.

The simplification phase is straightforward, exploiting the different cases of 4.5 as the key rules. The commutation phase is more delicate, and makes heavy use of the path cover computation.

For the single-qubit commutation rules we use targeting function which traverses each path starting from the input and applies the rule at the first possible position in the graph. This works very well and, in concert with the simplification rules, is guaranteed to reduce all single qubit Cliffords to one of the forms of $\mathfrak{C}_1$. However this approach fails for rules where a Pauli must commute with a CNOT gate. The REWRITE_TARGETED method allows the user to identify a single vertex in the rewrite rule with a single vertex in the graph to be matched. However this is rarely sufficient to uniquely determine the overall match. For example, the following rule



can be applied in two different ways to the central diagram below:



Note that the righthand rewrite does not produce a circuit. The current implementation provides no direct way for a simproc to specify that one matching should be preferred to another, even when it has the necessary information. For this reason, we have defined a metric which favours graphs where the Paulis are nearer to the inputs, and maximally penalises graphs with vertices which are not in any path.

The approach sketched here is very conservative: the optimiser only performs rewrites which result in diagrams which meet the (also very conservative) definition of circuit, even in cases where that diagram could be later rewritten to circuit.

7 Summary of results and discussion

Since we have formalised circuits in the ZX-calculus, our measure of size is the number of non-trivial vertices in the graph. This quantity is the gate count over the generating set S, Z, V, X, and CNOT, although each CNOT contributes 2 to the measure. A less obvious consequence

width	depth	input size	output size	size reduction	proof steps	time $\pm\sigma$
1	20	10.8	2.4	0.26	19.6	7 ± 3
2	20	30.2	9.0	0.30	78.9	1021 ± 680
3	20	44.2	14.7	0.32	148	3144 ± 2189
5	20	78.5	22.6	0.28	239	6679 ± 3091

Table 1: Summary of results on randomly generated circuits. Random circuits were generated with the program `randomCliffs.py` which is available from the project repository. Tests were performed on a basic desktop PC.

of the ZX-calculus formalism is that the swap gate is not represented at all, and CNOT gates may act on any qubits without penalty. This may or may not be realistic for a given quantum architecture, but it is a fundamental feature of the ZX-calculus, and removing it would require a string diagrammatic formalism that takes into account the planarity of diagrams. This would take us far beyond the scope of this work.

Our optimiser reduces all one- and two-qubit circuits to elements of $\mathfrak{C}_1$ and $\mathfrak{C}_2$ respectively; however in larger circuits it does not always find a minimal form. Notably our ruleset does not include any reductions for circuits of three or more qubits, so there are relatively simple CNOT circuits which cannot be reduced. Further, due to the extreme conservativity of our metric and targeting procedures, the optimiser does not always find valid rewrites among the ruleset. However, on a test set of randomly generated Clifford circuits it typically produces significant reductions of circuit size; see Table 1.

There is striking increase in compute time per proof step as the width of the circuit increases. This can be attributed to the increased number of CNOT gates: as the average vertex degree increases, the number of candidate matches including a given vertex increases geometrically. Since many of these matches will be invalid as circuit rewrites, a lot of time is spent searching for matches which are then rejected. Further, wider circuits require two-qubit commutation rules, which are metric-driven rather than targeted, and hence impose a much greater computational load.

Indeed, the main difficulty we have faced is taking control of the matching engine. Targeted rewriting can select good rewrite positions with high efficiency but does not discriminate between good and bad rewrites at that position. Metric-driven rewrites can perform such discrimination but at the cost of losing control over the search. Combining these features in a single combinator would greatly ease this difficulty.

Further, the targeting and metric functors are required to be stateless; this imposes another avoidable performance cost, as the path cover of the graph must be recomputed on every invocation, regardless of whether a rewrite was performed. Given the poor performance of interpreted python compared to the scala core, significant speed-up could be achieved by providing a range of built-in metrics for use in simprocs.

8 Conclusions and Future Work

Since Quantomatic was designed for interactive use, its simproc environment is currently quite limited. In repurposing the system to operate as a fully automated circuit optimisation tool, we have run up against its current limits, and our frequent resort to brute force and perversity is a reflection of these limitations, rather than the authors' mental states[4]. These limitations are not fundamental, and could be removed by minor extensions of the simproc API. Although there is much scope for improvement in our optimiser, we have demonstrated that the Quantomatic system can serve as the basis for an effective circuit optimisation tool.

References

[1] *The Jython Project : Python for the Java Platform.* Available at `http://www.jython.org`.

[2] S. Aaronson & D Gottesman (2004): *Improved Simulation of Stabilizer Circuits. Phys. Rev. A* 70(052328).

[3] Miriam Backens (2014): *The ZX-calculus is complete for stabilizer quantum mechanics. New Journal of Physics* 16(9), p. 093021, doi:10.1088/1367-2630/16/9/093021. Available at `http://stacks.iop.org/1367-2630/16/i=9/a=093021`.

[4] Miriam Backens (2015): *Making the stabilizer ZX-calculus complete for scalars.* In Chris Heunen, Peter Selinger & Jamie Vicary, editors: *Proceedings of the 12th International Workshop on Quantum Physics and Logic (QPL 2015), Electronic Proceedings in Theoretical Computer Science* 195, pp. 17–32, doi:10.4204/EPTCS.195.2.

[5] Miriam Backens, Simon Perdrix & Quanlong Wang (2016): *A Simplified Stabilizer ZX-calculus.* In R. Duncan & C. Heunen, editors: *Proceedings 13th International Conference on Quantum Physics and Logic (QPL 2016), Electronic Proceedings in Theoretical Computer Science* 236, pp. 1–20, doi:10.4204/EPTCS.236.1.

[6] Nicholas Chancellor, Aleks Kissinger, Stefan Zohren & Dominic Horsman (2016): *Coherent Parity Check Construction for Quantum Error Correction. arXiv.org preprint* (arXiv:1611.08012).

[7] Bob Coecke & Ross Duncan (2011): *Interacting Quantum Observables: Categorical Algebra and Diagrammatics. New J. Phys* 13(043016), doi:10.1088/1367-2630/13/4/043016. Available at `http://iopscience.iop.org/1367-2630/13/4/043016/`.

[8] Bob Coecke, Ross Duncan, Aleks Kissinger & Quanlong Wang (2015): *Generalised Compositional Theories and Diagrammatic Reasoning.* In G. Chirabella & R. Spekkens, editors: *Quantum Theory: Informational Foundations and Foils*, Springer, doi:10.1007/978-94-017-7303-4_10.

[9] Ross Duncan & Kevin Dunne (2016): *Interacting Frobenius Algebras are Hopf.* In Martin Grohe, Eric Koskinen & Natarajan Shankar, editors: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, LICS '16, ACM, pp. 535–544, doi:10.1145/2933575.2934550.

[10] Ross Duncan & Maxime Lucas (2014): *Verifying the Steane code with Quantomatic.* In Bob Coecke & Matty Hoban, editors: *Proceedings 10th International Workshop on Quantum Physics and Logic (QPL 2013), Electronic Proceedings in Theoretical Computer Science* 171, pp. 33–49, doi:10.4204/EPTCS.171.4.

[11] Ross Duncan & Simon Perdrix (2009): *Graph States and the necessity of Euler Decomposition.* In K. Ambos-Spies, B. Löwe & W. Merkle, editors: *Computability in Europe: Mathematical Theory and Computational Practice (CiE'09), Lecture Notes in Computer Science* 5635, Springer, pp. 167–177, doi:10.1007/978-3-642-03073-4.

[12] Ross Duncan & Simon Perdrix (2014): *Pivoting makes the ZX-calculus complete for real stabilizers.* In Bob Coecke & Matty Hoban, editors: *Proceedings 10th International Workshop on Quantum Physics and Logic, Castelldefels (Barcelona), Spain, 17th to 19th July 2013, Electronic Proceedings in Theoretical Computer Science* 171, Open Publishing Association, pp. 50–62, doi:10.4204/EPTCS.171.5.

[4] However the first author's mental state did take a beating from the lack of debugging facilities.

[13] Liam Garvie & Ross Duncan (2017): *Verifying the Smallest Interesting Colour Code with Quantomatic*. In: *Proceedings of QPL 2017, to appear*.

[14] Daniel Gottesman (1999): *The Heisenberg Representation of Quantum Computers*. In S. P. Corney, R. Delbourgo & P. D. Jarvis, editors: *Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, International Press, pp. 32–43.

[15] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2017): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. *arXiv preprint*.

[16] A. Kissinger, L. Dixon, R. Duncan, B. Frot, A. Merry, D. Quick, M. Soloviev & V. Zamdzhiev: *Quantomatic*. https://sites.google.com/site/quantomatic/. Available at `https://sites.google.com/site/quantomatic/`.

[17] Aleks Kissinger & Vladimir Zamdzhiev (2015): *Quantomatic: A Proof Assistant for Diagrammatic Reasoning*. In P. Amy Felty & Aart Middeldorp, editors: *Automated Deduction - CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, Springer, pp. 326–336, doi:10.1007/978-3-319-21401-6_22.

[18] Stephen Lack (2004): *Composing PROPs*. *Theory and Applications of Categories* 13(9), pp. 147–163.

[19] Saunders Mac Lane (1965): *Categorical Algebra*. *Bull. Amer. Math. Soc.* 71, pp. 40–106.

[20] Dmitri Maslov & Martin Roetteler (2017): *Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations*. *arXiv preprint*.

[21] Kang Feng Ng & Quanlong Wang (2017): *A universal completion of the ZX-calculus*. ArXiv:1706.09877.

[22] Simon Perdrix & Quanlong Wang (2016): *Supplementarity is Necessary for Quantum Diagram Reasoning*. In Piotr Faliszewski, Anca Muscholl & Rolf Niedermeier, editors: *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 58, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 76:1–76:14, doi:10.4230/LIPIcs.MFCS.2016.76. Available at `http://drops.dagstuhl.de/opus/volltexte/2016/6506`.

[23] Peter Selinger (2015): *Generators and relations for n-qubit Clifford operators*. *Logical Methods in Computer Science* 11(2:10), doi:10.2168/LMCS-11(2:10)2015.

A Electronic Resources

All the work described in this paper is available as a downloadable quantomatic project from the following URL.

`https://gitlab.cis.strath.ac.uk/kwb13215/Clifford-Quanto/`

Please download it and play around!

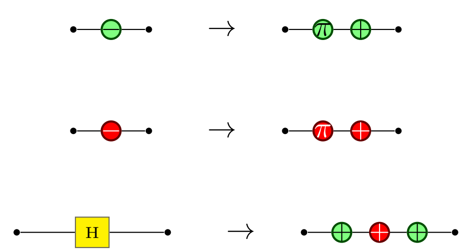
B Main Rules

The following derived rules are the core of the circuit transformations in the main proof development. The rules themselves are found in `derived-rules/` in the Quantomatic project, and their derivations are found in `derived-rules-proofs/`.

In all diagrams inputs are to the left, outputs to the right.

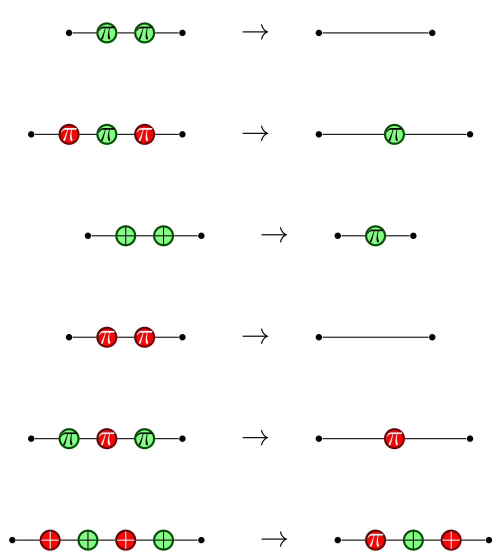
B.1 Init

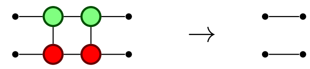
These rules remove all minus and H nodes from the graph.

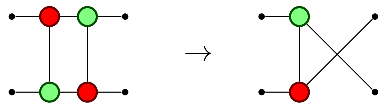


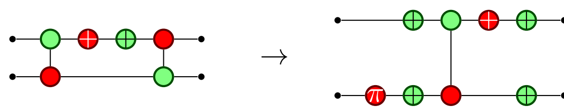
B.2 Always Rules

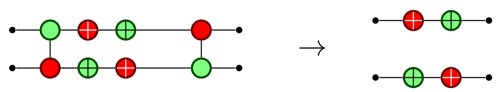
These are strictly reducing so can always be applied.

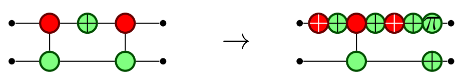


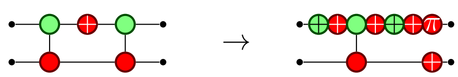


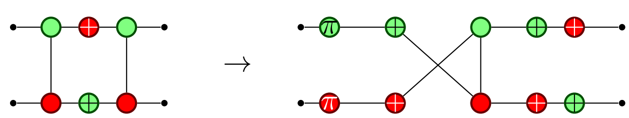




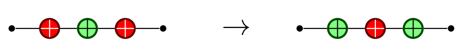








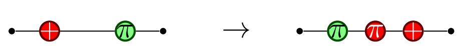
B.3 EulerProc

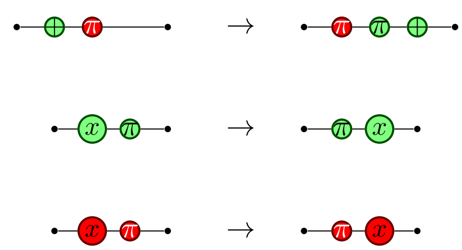


The inverse is also used.

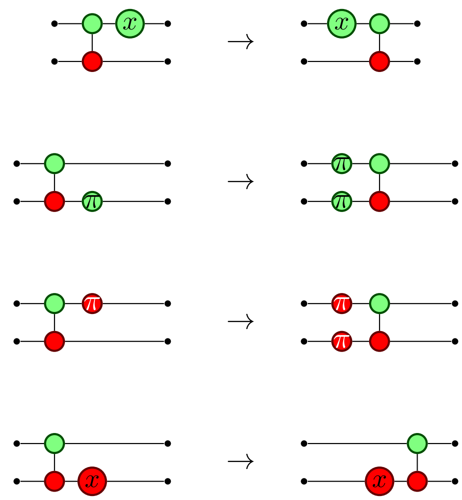
B.4 Pauli Commutation rules

Each of these is managed by a custom simproc which searches for a good place to apply them.



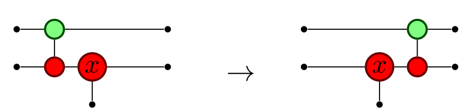


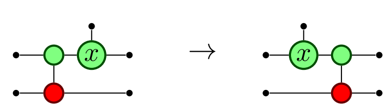
These rules are used by the Pauli Commutation proc, which is based on REDUCE_METRIC.



B.4.1 C2 Rules

These two alternate in the C2Proc simproc. Both have a custom targetting routine to select where they should be applied.



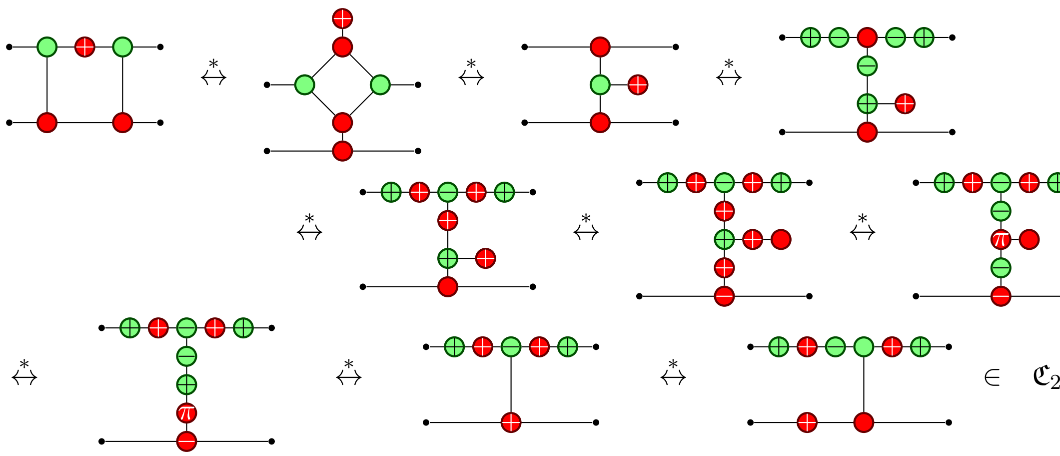


C Proof of Proposition 4.5

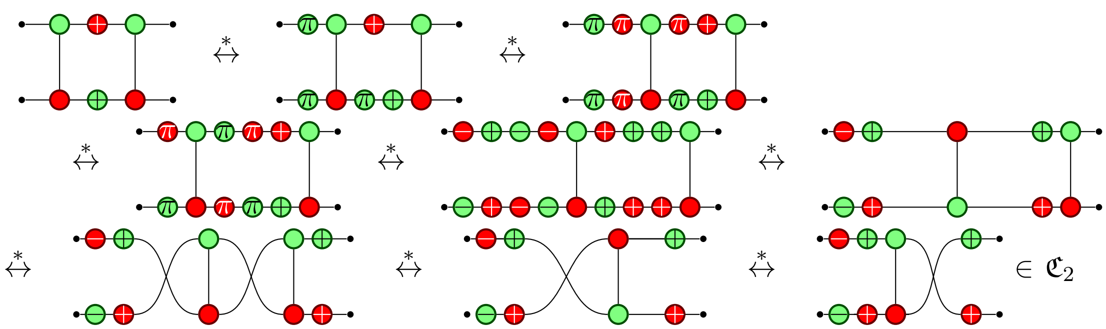
Proposition 4.5 Let $k \in \mathfrak{C}_2$ and let g be a generator of $\mathcal{C}_2$ as a ZX-calculus term; then $g \circ k \overset{*}{\leftrightarrow} k'$ for some $k' \in \mathfrak{C}_2$.

Proof. Notice that the 1-qubit Cliffords C_1 and C_2 which occur in k play no part in the computation, hence we ignore them. Further, for any $u \in \mathfrak{C}_1$, we have $(u \otimes \mathrm{id}) \circ \textsc{cnot} = (a \otimes \mathrm{id}) \circ \textsc{cnot} \circ (u_1 \otimes u_2)$ for some $u_1, u_2 \in \mathfrak{C}_1$ and $a \in \mathfrak{A}$, either by commuting vertices of the same colour, or taking out factors of Paulis. Similarly $(\mathrm{id} \otimes u) \circ \textsc{cnot} = (\mathrm{id} \otimes b) \circ \textsc{cnot} \circ (u_1 \otimes u_2)$ for some $b \in \mathfrak{B}$. Hence the only non-trival cases to consider are:

1. $g = \textsc{cnot}$ and $k = (a \otimes b) \circ \textsc{cnot}$. There are nine possibilities for this case, of which we consider two here. (i) $a = x_+$ and $b = \mathrm{id}$:

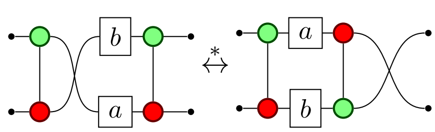


(ii) $a = x_+$ and $b = z_+$:



They remaining cases can all be derived from these two, except the case where $a = b = \mathrm{id}$, which is trivial.

2. $g = \textsc{cnot}$ and $k = (b \otimes a) \circ \sigma \circ \textsc{cnot}$. We have



but now TONC can be substituted using (**??**), reducing this case to the previous one.

3. $g = \textsc{cnot}$ and $k = \sigma$. As above, we rewrite $\textsc{cnot} \circ \sigma$ to $\sigma \circ \textsc{tonc}$ and apply (**??**).

∎