GENERATORS AND RELATIONS FOR SOME CLASSES OF QUANTUM CIRCUITS

by

Xiaoning Bian

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2023

Table of Contents

List of Figures

Abstract

We give three finite presentations in terms of generators and relations for three groups of operators. The operators are frequently used in quantum computation. The result can potentially be used to optimize quantum circuits. Two of the three proofs use the Reidemeister-Schreier theorem. Both the Reidemeister-Schreier theorem and the two proofs using it have been formally verified in the proof assistant Agda.

First, we give a finite presentation of the group of 2-qubit Clifford+T operators. The proof applies the Reidemeister-Schreier theorem to a finite presentation of a supergroup of the 2-qubit Clifford+T group that was given by Greylyn. The process generates hundreds of relations. We then simplify them to 20 relations using a *Pauli rotation representation* as an *almost-normal form*.

Second, we give a finite presentation of $U_n(\mathbb{Z}[\frac{1}{2}, i])$, the group of unitary $n \times n$-matrices with entries in $\mathbb{Z}[\frac{1}{2}, i]$. Amy, Glaudell, and Ross showed that this group is generated by a certain set of two-level matrices, and they also gave an exact synthesis algorithm that converts any element of $U_n(\mathbb{Z}[\frac{1}{2}, i])$ to a word in these generators. We say that a word is in *normal form* if it is an output of the exact synthesis algorithm. Using what we call *semantically guided rewriting*, we show that each word in the two-level generators is equivalent to a normal form under the congruence generated by finitely many relations.

Third, we give a finite presentation of the group of 3-qubit Clifford+CS operators. Amy, Glaudell, and Ross proved that up to a condition on the determinant, the n-qubit Clifford+CS group is exactly $U_{2^n}(\mathbb{Z}[\frac{1}{2}, i])$. In particular, the 3-qubit Clifford+CS group is a subgroup of $U_8(\mathbb{Z}[\frac{1}{2}, i])$ of index 2. Using a similar technique as in our first result, we repeatedly apply the Reidemeister-Schreier theorem to the presentation of $U_8(\mathbb{Z}[\frac{1}{2}, i])$ (from our second result), and then simplify thousands of relations into 17 relations. To do the simplification, we devise an almost-normal form for Clifford+CS operators. We also show that the 3-qubit Clifford+CS group, which is of course infinite, is the amalgamated product of three finite subgroups.

Acknowledgements

First of all, I would like to thank my advisor Peter Selinger for teaching me both research and other things. He guided me into the research world, helped me significantly in every research project of mine, and gave me tons of help in non-research aspects.

This research was supported by a departmental scholarship, a Killam Scholarship and several of Peter's grants — a DARPA research contract, an AFOSR research grant, an NSERC Discovery Grant, and an NSERC Accelerator Supplement.

Thanks to my supervisory committee Dorette Pronk and Neil J. Ross, and my external examiner Simon Perdrix for reading my work and providing feedback. Thanks to Wang Quanlong, Niel de Beaudrap, Frank Fu, Matt Amy, and Alexis Bernadet for stimulating discussions.

Thanks to the graduate coordinators Sara Faridi, David Iron, and Theo Johnson-Freyd, and to the department staff, especially Maria, for helping me many times with administrative tasks. Thanks to the department technical support person Balagopal Pillai for helping me many times with the usage of the department cluster.

Part of this research was done while I was visiting the math department of Technische Universität Darmstadt in the Spring and Summer of 2016, and the Simons Institute for the Theory of Computing at the University of California, Berkeley in the Fall of 2016.

Chapter 1

Introduction

For certain problems such as factoring, quantum algorithms give an exponential speedup over the best know classical algorithms [35]. For this reason, quantum computing has been a very active area of research. Many quantum algorithms use *quantum circuits* as building blocks. This thesis contributes to the theory of quantum circuits. Specifically, we find finite presentations for two classes of quantum circuits, namely 2-qubit Clifford+T circuits and 3-qubit Clifford+CS circuits. We also find a finite presentation of $U_n(\mathbb{Z}[\frac{1}{2}, i])$, the group of unitary $n \times n$-matrices with entries in $\mathbb{Z}[\frac{1}{2}, i]$. Recall that a presentation is given by a set of generators and a complete set of relations.

The class of Clifford operators is generated by tensor product and composition of the following operators:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \; H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; \omega = e^{\frac{\pi}{4} i} = \frac{1+i}{\sqrt{2}}, \; CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Here S is called the *phase gate*, H is called the Hadamard gate, ω is a scalar that is an 8-th root of unity, and CZ is the controlled Z-gate. The class of Clifford+T operators is generated by the Clifford operators plus the following T-gate:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \omega \end{bmatrix}.$$

The class of Clifford+CS operators is generated by the Clifford operators plus the controlled S-gate:

$$CS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}.$$

Note that $CZ = CS^2$. We also use the following gate:

$$K = \omega^\dagger H = \frac{1}{1+i}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Here i is the complex unit, and K is a scaled version of H.

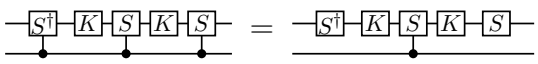
The class of Clifford+T circuits, and especially the simplification of Clifford+T circuits, is a topic of current interest in quantum computing [5, 13, 14, 24, 26, 29]. The Clifford+T gate set is both universal [30] and convenient for quantum error correction [12], and is therefore one of the preferred gate sets for fault-tolerant quantum computing. Generally, in a fault-tolerant regime, applying a Clifford gate is some orders of magnitude cheaper than applying a T-gate, and therefore, it is sensible to try to simplify circuits so as to minimize the T-count (the number of T-gates) [4]. Many methods for doing so have been proposed in the recent literature, including methods based on matroid partitioning [3], Reed-Muller codes [5], and ZX calculus [13, 14, 26].

It is well-known that certain useful classes of quantum circuits correspond to particular subrings of the complex numbers. Kliuchnikov et al. showed that a unitary operator is representable by a 1-qubit Clifford+T circuit if and only if its entries belong to the ring $\mathbb{Z}[1/\sqrt{2}, i]$ [27]. Giles and Selinger extended the result to the n-qubit case [16]. Amy et al. studied the correspondences of several restricted classes of Clifford+T circuits to the rings $\mathbb{Z}[1/2], \mathbb{Z}[1/\sqrt{2}], \mathbb{Z}[1/i\sqrt{2}]$, and $\mathbb{Z}[1/2, i]$. There is one crucial concept shared by these works, i.e., the *exact synthesis algorithm*, which decomposes a given unitary with a certain requirement on its entries into a product of one- and two-level matrices. Then they show the one- and two-level matrices are representable by circuits. We say that a word in one- and two-level matrices is in *normal form* if it is an output of the exact synthesis algorithm. Based on this normal form, several presentations by generators and relations have been found. Greylyn [21] gave a presentation for $U_4(\mathbb{Z}[1/\sqrt{2}, i])$, and Li et al. [28] gave a presentation for $U_n(\mathbb{Z}[\frac{1}{2}])$.

The class of Clifford+CS circuits is also a topic of current interest. Just like Clifford+T circuits, the class of Clifford+CS circuits is universal for quantum computing. Amy, Glaudell, and Ross gave a characterization of the group of n-qubit

Clifford+CS operators, showing that, up to a trivial condition on the determinant, a matrix is in this group if and only if it is unitary and its matrix entries belong to the ring $\mathbb{Z}[\frac{1}{2}, i]$ [2]. As a consequence of this, or alternatively since the CS gate is representable as a Clifford+T circuit with T-count 3, the Clifford+CS group is a subgroup of Clifford+T; see also [6]. Like the T-gate, the CS-gate is a non-Clifford gate that is relatively expensive to perform in a fault-tolerant regime. In [22], Haah and Hastings showed how to construct a fault-tolerant CS-gate via magic state distillation. It therefore makes sense to try to minimize the number of CS-gates. In [15], Garion and Cross described a CS- and CX-optimal canonical form for the 2-qubit group generated by the gates $\{X, T, CX, CS\}$. Glaudell, Ross, and Taylor gave a CS-optimal normal form for 2-qubit Clifford+CS circuits [18].

Many strategies for minimizing T-count or CS-count are based on *rewriting*, which means applying algebraic laws, such as $T^2 = S$, or such as



to replace subcircuits by equivalent ones. In designing such rewriting techniques, it can be useful to have a complete set of such algebraic laws, i.e., a set of equations by which any circuit can in principle be rewritten into any equivalent circuit. While a complete set of such equations does not in itself guarantee the existence of good rewriting strategies, it can be a useful tool for designing such strategies.

1.1 Outline

This thesis is organized as follows: Chapter 2 is a summary of standard concepts of quantum computation, the presentation of $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$, rings, monoid presentations, amalgamation, and proof assistants, which are needed throughout the thesis. Chapter 3 gives a finite presentation for the group of 2-qubit Clifford+T circuits. Chapter 4 gives a finite presentation for $U_n(\mathbb{Z}[\frac{1}{2}, i])$, the group of unitary $n \times n$-matrices with entries in $\mathbb{Z}[\frac{1}{2}, i]$. Chapter 5 gives a finite presentation for the group of 3-qubit Clifford+CS operators, and shows that it is an amalgamated product of three of its finite subgroups. Chapter 6 contains some concluding remarks.

1.2 Contributions

Chapters 3, 4, and 5 contain my original contributions, which are based on three published papers, all coauthored with my supervisor Peter Selinger. The results of Chapter 3 were obtained jointly by my coauthor and me, and my contribution was the simplification of the relations. We contributed equally to the results of Chapter 4. The results of Chapter 5 were obtained by myself.

During my Ph.D. study, I published two other papers [14, 13] (both coauthored with Harny Wang and Niel de Beaudrap) about Clifford+T circuit optimization using ZX-calculus, which are not included in this thesis.

Chapter 2

Preliminaries

2.1 Quantum computation

We only introduce the basic ideas of quantum computation. For more information about quantum computation, please refer to [30, 33]. The basic concepts of quantum computation are *states* and *operations* that act on states. Let $\mathbb{C}$ be the field of complex numbers. We write $\mathbb{C}^n$ for the space of n-dimensional column vectors.

2.1.1 n-qubit states

Definition 2.1.1. An n-qubit state is a unit vector in $\mathbb{C}^{2^n}$.

In other words, the state space is the set of 2^n-dimensional complex unit vectors. For example, the space of 1-qubit states is the unit sphere in the 2-dimensional complex vector space. Modulo scalars, it can be identified with the real 3-sphere, which is called the *Bloch sphere* in quantum computation.

For $\mathbb{C}^{2^n}$, we use the "ket-binary" notation to denote the standard basis. For example, when $n = 2$:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

In general

$$|b_n b_{n-1} ... b_0\rangle = e_k, \text{ where } k = \sum_{j=0}^{n} b_j 2^j, \text{ and } e_k \text{ is the } k\text{-th standard basis vector.}$$

Note that $b_n b_{n-1} ... b_0$ is the binary representation of k, and that the standard basis starts from e_0. In the following, we will identify each natural number index with its binary representation. One convenience of this notation is

$$|a_m a_{m-1} ... a_0\rangle \otimes |b_n b_{n-1} ... b_0\rangle = |a_m a_{m-1} ... a_0 b_n b_{n-1} ... b_0\rangle.$$

Another way to look at this is that every basis vector of $\mathbb{C}^{2^{m+n}}$ is a tensor product of basis vectors of $\mathbb{C}^{2^m}$ and $\mathbb{C}^{2^n}$. We can also identify $\mathbb{C}^{2^n}$ with the n-fold tensor product of $\mathbb{C}^2$. The ket-binary notation reflects this identification.

There are two and only two kinds of operations that can be applied to an n-qubit state — *unitary transformations* and *measurements*. Quantum algorithms are often of the form that first apply some number of unitary transformations to a an n-qubit state and then do a measurement, and based on the measurement result, some output or conclusion is drawn.

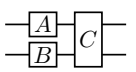
2.1.2 n-qubit unitary transformations and quantum circuits

A linear function $U : \mathbb{C}^k \to \mathbb{C}^m$ is called an *isometry* if for all v, w in $\mathbb{C}^k$, we have $\langle Uv, Uw\rangle = \langle v, w\rangle$. U is called *unitary* if it is an invertible isometry. Note that U is unitary if and only if $UU^\dagger = I$ and $U^\dagger U = I$, where $U^\dagger$ denotes the adjoint of U. For a matrix, the adjoint is the complex conjugate of the transpose.

Definition 2.1.2. An n-qubit unitary transformation is a unitary transformation on $\mathbb{C}^{2^n}$

We also refer to an n-qubit unitary transformation as an *n-qubit operator*, especially in combination with an adjective, such as 2-*qubit Clifford+T operator* and 3-*qubit Clifford+CS operator*. The unitary condition is needed to preserve the norm of a state so that applying a unitary transformation to a state produces a state.

In classical computation, any boolean function can be constructed using basic logic gates such as *and*, *or*, and *fan-out*. Here fan-out is a copy operation, i.e., a gate that inputs a boolean x and outputs the pair (x, x). One often uses circuits to describe these constructions. In quantum computation, we pick out some basic unitaries, also called *gates*, and we can construct other unitaries from these gates. Often, the construction is expressed by a *quantum circuit*. This construction only uses two operations to combine gates — tensor product and composition. Quantum circuits are a way to say when and where the tensor products and compositions are. For example, given two 1-qubit gates A, B and a 2-qubit gate C, the following quantum circuit represents the 2-qubit operator $(A \otimes B) \circ C$:



.

Not every unitary is representable by a circuit using a finite gate set, since there are uncountably many unitaries and countably many circuits over a finite gate set. A set of gates is said to be *universal* if the set of operators representable by circuits over it is *dense* in the set of all unitaries.

Consider the following gate sets (individual gate definitions are given in the introduction):

- Clifford gate set: $\{S, H, CZ, \omega\}$.
- Clifford+T gate set: $\{S, H, CZ, \omega, T\}$.
- Clifford+CS gate set: $\{S, K, CS, i\}$.

In the introduction, we also included the scalar ω in the Clifford+CS gate set. But in quantum computing, scalars are often disregarded, and to get a better connection between Clifford+CS circuits and matrix rings, we use the gate set $\{S, K, CS, i\}$ instead.

We call the operators generated by the above gate sets *Clifford operators*, *Clifford+T operators*, and *Clifford+CS operators*, respectively. We write $\mathcal{C}(n)$, $\mathcal{CT}(n)$, and $\mathcal{CCS}(n)$ for the set of n-qubit Clifford, Clifford+T, and Clifford+CS operators respectively. Note that for each n, $\mathcal{C}(n)$, $\mathcal{CT}(n)$, and $\mathcal{CCS}(n)$ all form a group. It is well-known that $\mathcal{C}(n)$ is finite for any given n (see, e.g., [34]), and therefore not universal for quantum computing. Both $\mathcal{CT}(n)$ and $\mathcal{CCS}(n)$ are universal (see, e.g., [2, 16]). Another thing to note is that CZ is contained in $\mathcal{CCS}(n)$ for $n > 1$, since $CZ = CS^2$.

Both CZ and CS are two-qubit gates. Another commonly used two-qubit gate is the *Swap* gate:

$$Swap = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

It acts on a 2-qubit basis vector by swapping the "bits", i.e.,

$$Swap|ab\rangle = |ba\rangle.$$

In general, if a 2-qubit state is a tensor product of two 1-qubit states p and q,

$$Swap(p \otimes q) = q \otimes p.$$

In this sense, *Swap* serves to swap two 1-qubit states. If a 2-qubit gate A is "insensitive" to swapping, i.e., $A = Swap\, A\, Swap$, we say A is *symmetric* on its two input qubits. Both CZ and CS are symmetric on their two input qubits.

We use the following circuit notations for S, H, T, K, CZ, and CS, respectively:

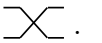
$S =$  , $H =$, $T =$, $K =$,

$CZ =$, $CS =$.

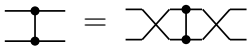
The CZ gate is usually denoted

 ,

but since it is symmetric with respect to its two qubits, we prefer the more symmetric notation shown above. We also use a similar notation for the CS gate, except that we label it with an "i". The circuit notation for the *Swap* gate is

$Swap =$  .

The symmetric property for CZ (similarly for CS) then becomes

 .

We number the lines from top to bottom (counting from 0). This allows us to also sometimes write gates in an indexed notation; for example S_j denotes an S-gate applied to the j-th qubit,

$S_1 =$ $= I \otimes S \otimes I$, and $CZ_{12} =$ $= I \otimes CZ$,

where I is the 1-qubit identity operator. In some contexts, such as to differentiate the indexed gate with other indexed symbols, we use extra parentheses, e.g., $S_{(1)} = S_1$ and $CS_{(12)} = CS_{12}$. We also use the X-gate and the controlled X-gate, which are definable as follows:

$X = HSSH$(or $KSSKi$), $CX =$ (or $\cdot\, i$).

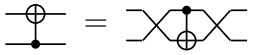
The circuit notation for the controlled X-gate is

The CX (CZ and CS) gate is a *controlled* X gate (controlled Z-gate and controlled S-gate) in the sense that applying a controlled operator to a standard basis vector doesn't change the qubit state labelled by a dot, called the *control qubit*, but changes the qubit state labelled by the corresponding gate, say X, called the *target qubit*, conditionally on the state of the control qubit. Specifically,

$$CX|0b\rangle = |0b\rangle, \quad CX|1b\rangle = |1b'\rangle, \text{where } |b'\rangle = X|b\rangle.$$

Since CX is not symmetric on its input qubits, we also have the "upside-down" version

$$CX_{10} = \quad = \quad$$



Note that we have used the indexed gate notation CX_{10} (to differ from CX) with the convention that the first index is the control and second is the target.

2.1.3 Measurement

The other operation is the measurement. A measurement is like a finite-valued random variable with "side-effect" which modifies the qubit state being measured. First we give the definition of a full measurement, which measures the whole space.

Definition 2.1.3. Given an n-qubit state $\sum_{j=00\ldots0}^{11..1} \alpha_j e_j$, a n-qubit measurement acting on it outputs j and alters it to e_j with a probability $|\alpha_j|^2$.

Note that we used the binary representation for the index j. We also say the state *collapses* to e_j after measurement. Now we give the definition of the 1-qubit measurement acting on the last qubit.

Definition 2.1.4. Given an n-qubit state $\sum_{j=00\ldots0}^{11\ldots1} \alpha_j|j\rangle$, a 1-qubit measurement acting on the last qubit outputs b and alters the state to $\sum_{j=00\ldots0}^{11\ldots1} \alpha'_{jb}|jb\rangle$ with a probability $p = \sum_{j=00\ldots0}^{11\ldots1} |\alpha_{jb}|^2$, where $b = 0, 1$, and $\alpha'_{jb} = \alpha_{jb}/\sqrt{p}$.

Note that after the measurement, the state collapses into an $(n-1)$-qubit state tensoring $|b\rangle$. Now we give the definition of the 1-qubit measurement acting on the k-th qubit.

Definition 2.1.5. Given an n-qubit state $\sum_{j=00\ldots0}^{11\ldots1} \alpha_j |j\rangle$, a 1-qubit measurement acting on the k-th qubit outputs b and alters the state to $\sum_{j=00\ldots0}^{11\ldots1} \sum_{l=00\ldots0}^{11\ldots1} \alpha'_{jbl} |jbl\rangle$ with a probability $p = \sum_{j=00\ldots0}^{11\ldots1} \sum_{l=00\ldots0}^{11\ldots1} |\alpha_{jbl}|^2$, where $b = 0, 1$, $\alpha'_{jbl} = \alpha_{jbl}/\sqrt{p}$, j is of length $k-1$, and l is of length $n-k$, and jbl is a concatenation of binary strings j, b, and l.

We can repeatedly apply a k-th qubit measurement, which gives all partial (also full) measurements. If in this way, we measure all qubits and collect all the outputs b's, the measurement result coincides with the full measurement definition.

2.1.4 One- and two-level matrices

One- and two-level matrices are another commonly used way in quantum computing to define new operators. Consider complex matrices of dimension $n \times n$. We number the rows and columns of matrices starting from zero, i.e., the entries of an $n \times n$-matrix are $a_{00}, \ldots, a_{n-1,n-1}$. We define a special class of matrices called one- and two-level matrices.

Definition 2.1.6. Given $z \in \mathbb{C}$ and $j \in \{0, \ldots, n-1\}$, the *one-level matrix* $z_{[j]}$ is

$$z_{[j]} = \begin{array}{c} \\ \vdots \\ j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & j & \cdots \end{array} \\ \begin{bmatrix} I & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & I \end{bmatrix} \end{array},$$

i.e., the matrix that is like the $n \times n$-identity matrix, except that the entry at position (j, j) is z. Similarly, given a 2×2-matrix $U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $j, k \in \{0, 1, ..., n-1\}$ with $j < k$, the *two-level matrix* $U_{[j,k]}$ is

$$U_{[j,k]} = \begin{array}{c} \\ \vdots \\ j \\ \vdots \\ k \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \cdots & j & \cdots & k & \cdots \end{array} \\ \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & a & 0 & b & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & c & 0 & d & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \end{array}.$$

Note that if $|z| = 1$, then $z_{[j]}$ is unitary; similarly, if U is unitary, then so is $U_{[j,k]}$. We say that $U_{[j,k]}$ is a two-level matrix of *type* U, and similarly, we say that $z_{[j]}$ is a one-level matrix of *type* z.

2.2 Presentation of $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$

As usual, $\mathbb{Z}$ is the ring of integers. Let $R = \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ be the smallest subring of the complex numbers containing $\frac{1}{\sqrt{2}}$ and i. Let $\omega = e^{i\pi/4}$, which is an 8th root of unity, and note that $\omega = \frac{1+i}{\sqrt{2}} \in R$. As before, $U_4(R)$ is the group of unitary 4×4-matrices with entries in R.

Greylyn [21] gave a presentation of $U_4(R)$ by generators and relations. His generators are $\omega_{[j]}$, $X_{[j,k]}$, and $H_{[j,k]}$, where $j, k \in \{0, ..., 3\}$ and $j < k$. The relations are shown in Figure 2.1. The intended interpretation of the generators is as 1- and 2-level matrices. Note that we index rows and columns of matrices starting from 0, whereas Greylyn indexed them starting from 1. Greylyn's result is the following:

Theorem 2.2.1 (Greylyn [21])**.** *A presentation of the group $U_4(R)$ is given by $(\mathcal{Y}, \Delta)$, where the set of generators is $\mathcal{Y} = \{\omega_{[j]}, X_{[j,k]}, H_{[j,k]} \mid j, k \in \{1, ..., 4\}$ and $j < k\}$, and the set of relations Δ is shown in Figure 2.1.*

2.3 Some algebra

2.3.1 Gaussian integers and Gaussian dyadics

As usual, we write $\mathbb{N} = \{0, 1, \ldots\}$ for the set of natural numbers. We also write $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ to denote the rings of integers, real numbers, and complex numbers, respectively. We write $z^\dagger$ for the complex conjugate of a complex number. Recall that in any ring, a *unit* is an invertible element.

Let $\mathbb{Z}[i] = \{a + bi \mid a, b \in \mathbb{Z}\}$ be the ring of Gaussian integers. It is a Euclidean domain, and therefore, division with remainder, greatest common divisors, divisibility, and the concept of a prime (i.e., a non-unit that cannot be written as the product of two non-units) all can be used in $\mathbb{Z}[i]$. Let $\gamma = 1 + i$. It is well-known that γ is a Gaussian prime [25]. Note that γ is a divisor of 2, and γ^2 and 2 divide each other.

(a) Order of generators:

$$\omega_{[j]}^8 = \epsilon \tag{A1}$$
$$H_{[j,k]}^2 = \epsilon \tag{A2}$$
$$X_{[j,k]}^2 = \epsilon \tag{A3}$$

(b) Disjoint generators commute:

$$\omega_{[j]}\omega_{[k]} = \omega_{[k]}\omega_{[j]}, \quad \text{where } j \neq k \tag{A4}$$
$$\omega_{[\ell]}H_{[j,k]} = H_{[j,k]}\omega_{[\ell]}, \quad \text{where } \ell \neq j,k \tag{A5}$$
$$\omega_{[\ell]}X_{[j,k]} = X_{[j,k]}\omega_{[\ell]}, \quad \text{where } \ell \neq j,k \tag{A6}$$
$$H_{[j,k]}H_{[\ell,t]} = H_{[\ell,t]}H_{[j,k]}, \quad \text{where } \{\ell,t\} \cap \{j,k\} = \emptyset \tag{A7}$$
$$H_{[j,k]}X_{[\ell,t]} = X_{[\ell,t]}H_{[j,k]}, \quad \text{where } \{\ell,t\} \cap \{j,k\} = \emptyset \tag{A8}$$
$$X_{[j,k]}X_{[\ell,t]} = X_{[\ell,t]}X_{[j,k]}, \quad \text{where } \{\ell,t\} \cap \{j,k\} = \emptyset \tag{A9}$$

(c) X permutes indices:

$$X_{[j,k]}\omega_{[k]} = \omega_{[j]}X_{[j,k]} \tag{A10}$$
$$X_{[j,k]}\omega_{[j]} = \omega_{[k]}X_{[j,k]} \tag{A11}$$
$$X_{[j,k]}X_{[j,\ell]} = X_{[k,\ell]}X_{[j,k]} \tag{A12}$$
$$X_{[j,k]}X_{[\ell,j]} = X_{[\ell,k]}X_{[j,k]} \tag{A13}$$
$$X_{[j,k]}H_{[j,\ell]} = H_{[k,\ell]}X_{[j,k]} \tag{A14}$$
$$X_{[j,k]}H_{[\ell,j]} = H_{[\ell,k]}X_{[j,k]} \tag{A15}$$

(d) $\omega_{[j]}\omega_{[k]}$ is diagonal:

$$\omega_{[j]}\omega_{[k]}X_{[j,k]} = X_{[j,k]}\omega_{[j]}\omega_{[k]} \tag{A16}$$
$$\omega_{[j]}\omega_{[k]}H_{[j,k]} = H_{[j,k]}\omega_{[j]}\omega_{[k]} \tag{A17}$$

(e) Relations for H:

$$H_{[j,k]}X_{[j,k]} = \omega_{[k]}^4 H_{[j,k]} \tag{A18}$$
$$H_{[j,k]}\omega_{[j]}^2 H_{[j,k]} = \omega_{[j]}^6 H_{[j,k]}\omega_{[j]}^3\omega_{[k]}^5 \tag{A19}$$
$$H_{[j,k]}H_{[\ell,t]}H_{[j,\ell]}H_{[k,t]} = H_{[j,\ell]}H_{[k,t]}H_{[j,k]}H_{[\ell,t]}, \quad \text{where } k < \ell \tag{A20}$$

Figure 2.1: Greylyn's relations for $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$. Whenever we use a generator $X_{[j,k]}$ or $H_{[j,k]}$, we implicitly assume that $j < k$. Here ϵ is the empty word.

Using Euclidean division, it is further easy to check that

$$\mathbb{Z}[i]/(\gamma) = \{0, 1\}, \quad \mathbb{Z}[i]/(2) = \mathbb{Z}[i]/(\gamma^2) = \{0, 1, i, 1+i\}, \quad \text{and}$$

$$\mathbb{Z}[i]/(\gamma^3) = \{\pm 1, \pm i, 0, 1+i, 1-i, 2\}.$$

Definition 2.3.1. For $x \in \mathbb{Z}[i]$, if $x \equiv 0 \pmod{\gamma}$, we say x is *even*, otherwise *odd*.

For example, $2+3i$ and $5+2i$ are odd, and $2+4i$ and $1+3i$ are even. In particular, in our context, by an even Gaussian integer we mean one that is divisible by γ, not necessarily by 2. However if $x \in \mathbb{Z}$, then x is even (odd) as an integer if and only if x is even (odd) as a Gaussian integer. Therefore, our definition of parity extends the usual one on the integers. The following lemmas are straightforward.

Lemma 2.3.2. *Let $\alpha = a + bi \in \mathbb{Z}[i]$. Then α is odd if and only if $\|\alpha\|^2 = a^2 + b^2$ is an odd integer if and only if $a + b$ is an odd integer.*

Lemma 2.3.3. *If $\alpha \in \mathbb{Z}[i]$ is odd, then $\alpha \equiv \pm 1, \pm i \pmod{\gamma^3}$. In other words, $\alpha \equiv i^e \pmod{\gamma^3}$ for some $e \in \{0, 1, 2, 3\}$. Moreover, $\alpha \equiv i^e \pmod{\gamma^2}$ for some $e \in \{0, 1\}$.*

Let $\mathbb{D} = \mathbb{Z}[\frac{1}{2}] = \{\frac{a}{2^k} \mid a \in \mathbb{Z}, k \in \mathbb{N}\}$ be the ring of *dyadic fractions*, i.e., fractions whose denominator is a power of 2. Consider $\mathbb{D}[i] = \{r + si \mid r, s \in \mathbb{D}\} = \mathbb{Z}[\frac{1}{2}, i]$. For every $t \in \mathbb{D}[i]$, there exists some natural number k such that $\gamma^k t \in \mathbb{Z}[i]$. This motivates the following definition (similar definitions are also used in [17, 21, 2, 28]).

Definition 2.3.4. Let $t \in \mathbb{D}[i]$. A natural number $k \in \mathbb{N}$ is called a *denominator exponent* for t if $\gamma^k t \in \mathbb{Z}[i]$. The least such k is called the *least denominator exponent* of t, denoted by $\mathrm{lde}_\gamma(t)$. Equivalently, the least denominator exponent of t is the smallest $k \in \mathbb{N}$ such that t can be written in the form $\frac{s}{\gamma^k}$, where $s \in \mathbb{Z}[i]$.

More generally, we say that k is a denominator exponent for a vector or matrix if it is a denominator exponent for all of its entries. The least denominator exponent for a vector or matrix is therefore the least k that is a denominator exponent for all of its entries.

Note that for $t \in \mathbb{D}[i]$, if $k = \mathrm{lde}_\gamma(t) > 0$, then $\gamma^k t$ is odd.

2.3.2 Monoid presentation

If X is a set, let us write X^* for the set of finite sequences of elements of X, which we also call *words* over the alphabet X. We write $w \cdot v$ or simply wv for the concatenation of words, making X^* into a monoid. The unit of this monoid is the empty word ϵ. As usual, we identify X with the set of one-letter words.

Let G be a monoid and let $X \subseteq G$ be a subset of G. We write $\langle X \rangle$ for the smallest submonoid of G containing X, and we say that X *generates* G if $\langle X \rangle = G$. Given any word $w \in X^*$, we write $[w]_G \in G$ for the canonical interpretation of w in G, i.e., $[-]_G : X^* \to G$ is the unique monoid homomorphism such that $[x]_G = x$ for all $x \in X$.

A *relation* over X is an element of $X^* \times X^*$, i.e., an ordered pair of words. We say that a relation (w, v) is *sound* in G if $[w]_G = [v]_G$. If Γ is a set of relations over X, we write $\sim_\Gamma$ for the smallest congruence relation on X^* containing Γ. Here, as usual, a congruence relation is an equivalence relation that is compatible with the monoid operation, i.e., such that $w \sim v$ and $w' \sim v'$ implies $ww' \sim vv'$. If $\sim$ is a congruence, then the quotient $G/\sim$ is a well-defined monoid. If $f : G \to H$ is a monoid homomorphism, then its kernel is the relation $\sim_f$ defined by $u \sim_f v$ if and only if $f(u) = f(v)$. The kernel of every monoid homomorphism is a congruence, and conversely, every congruence $\sim$ is the kernel of some homomorphism, namely of the canonical quotient map $G \to G/\sim$.

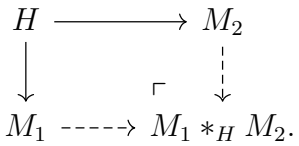
Remark 2.3.5. To check that a congruence $\sim_\Gamma$ is sound for G, it suffices to check that $[w]_G = [v]_G$ holds for all $(w, v) \in \Gamma$.

Given a set X of generators for a monoid G and a set Γ of sound relations, we say that Γ is *complete* if for all $w, v \in X^*$, $[w]_G = [v]_G$ implies $w \sim_\Gamma v$. In that case, we also say that (X, Γ) is a *presentation by generators and relations* (or simply *presentation*) of G.

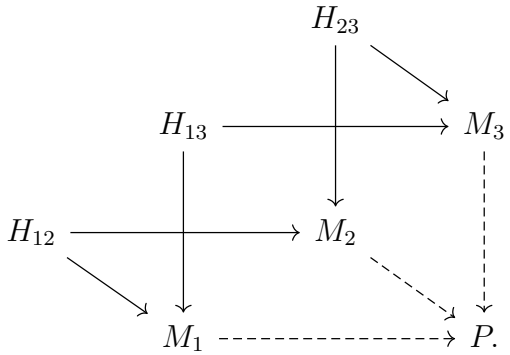
2.3.3 Amalgamation

Let us first recall the definition of an amalgamated product of two monoids. For category theorists, this is simply a pushout: Given monoids M_1, M_2, and H with morphisms $H \to M_1$ and $H \to M_2$, the amalgamated product $M_1 *_H M_2$ is the

pushout



The amalgamated product of three monoids is defined similarly. Suppose M_1, M_2, M_3, H_{12}, H_{23}, H_{13} are monoids with morphisms $H_{jk} \to M_j$ and $H_{jk} \to M_k$ for all relevant j and k. Then the amalgamated product P is the colimit of the following diagram, which generalizes a pushout:



One way in which an amalgamated product can arise is in the following situation. Suppose we have three sets of generators X, Y, and Z, and three monoid presentations $M_1 = \langle X \cup Y \mid \Gamma_1 \rangle$, $M_2 = \langle X \cup Z \mid \Gamma_2 \rangle$, and $M_3 = \langle Y \cup Z \mid \Gamma_3 \rangle$. We can take $H_{12} = \langle X \rangle$, $H_{13} = \langle Y \rangle$ and $H_{23} = \langle Z \rangle$, with the obvious maps. Then the amalgamated product P has the presentation $\langle X \cup Y \cup Z \mid \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \rangle$.

In cases where P is an infinite monoid or group, it is remarkable when M_1, M_2, and M_3 can be chosen to be finite. In that case, the slogan "the only relations that hold in P are relations that hold in a finite submonoid of P" applies.

2.4 Proof assistant Agda

2.4.1 Proof assistants

Two of our results involve a large amount of computation. We initially wrote a computer program to generate and verify the relations. However, this raised an issue: our program was large and complicated and used a variety of tactics, for which we could not claim with mathematical certainty that our program was free of

bugs, nor that it didn't use some hidden assumptions. Moreover, it would have been unreasonable for any referee to verify our calculations.

For this reason, we decided to go one step further and formalize the soundness and completeness proofs in a *proof assistant*. A proof assistant is a piece of software in which one can write definitions, theorems, and proofs, and the software will check the correctness of the proofs. Purists might object that the proof assistant is itself a piece of software that might be buggy. But, as has been argued eloquently by [19, 23], current proof assistants can be scrutinized at many levels and are many orders of magnitude more reliable than the traditional way of checking paper-and-pencil proofs. The particular proof assistant we used in this work is Agda [1].

2.4.2 Agda

We will give an example to show how formal proofs work. For a detailed introduction, please see [1]. We will define natural numbers, addition on natural numbers, and show that addition is associative. Note that lines starting with `--` are comments.

```
-- The following literally says ℕ is a Set and ℕ has two kinds of
-- elements:
-- 1) zero is an element of ℕ;
-- 2) if n ∈ ℕ, succ n is an element of ℕ.

-- Note that ‘‘:’’ in code corresponds to ‘‘∈’’ in math. Set is
-- intuitively the class of all sets.
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ

-- Implicitly, the data keyword also guarantees:
-- 1) two kinds of elements are never equal, i.e., zero ≠ succ n for
-- all n ∈ ℕ.
-- 2) succ n and succ m are never equal unless n = m.
-- 3) we can do a case distinction on elements of ℕ, which is called
-- ‘‘pattern matching’’ by computer scientists.
```

```
-- 4) ℕ is the smallest set satisfying the above conditions, or
-- equivalently for category theorists, ℕ is an initial algebra.


-- Addition is defined as follows:

-- This says + is left associative with a priority 6 (e.g., less than
-- the priority of multiplication). Left associative means that a+b+c
-- will be interpreted as (a+b)+c.
infixl 6 _+_

-- We do pattern matching on the first argument. Note that the
-- recursive call terminates since the first argument is ‘‘smaller’’
-- (in Agda lingo, structurally smaller, since it contains one less
-- succ).
_+_ : ℕ → ℕ → ℕ
zero + n = n
succ m + n = succ (m + n)

-- Before we show associativity we need to define the equality
-- predicate.  We write a == b for the claim a equals b.
infix 5 _==_

-- Equality on a set A is defined as the smallest reflexive relation.
data _==_ {A : Set} (x : A) : A → Set where
  refl : x == x

-- The following is a lemma showing that for any function f, if a == b
-- then f a == f b.
context : ∀ {A : Set} {B : Set} → {a b : A} → (f : A → B) →
  a == b → f a == f b
context f refl = refl
```

```
-- Associativity. Note that the induction hypothesis is a recursive
-- call of lemma-add-assoc on a structural smaller argument.
lemma-add-assoc : ∀ (x y z : ℕ) → (x + y) + z == x + (y + z)
lemma-add-assoc zero y z = refl
lemma-add-assoc (succ x) y z = context succ ih
  where
    ih = lemma-add-assoc x y z
```

Chapter 3

Presentation for 2-qubit Clifford+T operators

In this chapter, we give a complete set of relations for 2-qubit Clifford+T circuits. We do this in several steps. First, a presentation of the group $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$ of all unitary 4×4-matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ is known due to the work of Greylyn [21]. Second, it is known that the group of 2-qubit Clifford+T circuits is exactly the subgroup of this group consisting of matrices whose determinant is in $\{\pm 1, \pm i\}$ [16]. Third, there is a theorem in group theory called the Reidemeister-Schreier theorem, by which a complete set of relations for a subgroup can be derived from a complete set of relations for the supergroup. Fourth, since the resulting relations are very long and complicated, we simplify them.

The last two steps of this procedure (applying the Reidemeister-Schreier theorem and simplifying the resulting relations) require a large amount of algebraic manipulations. Our longest equational proof has 480 steps, each of which in turn requires a lemma or rewrite procedure whose proof itself requires many equational steps. Such proofs would be impossible to verify by hand, and even verifying them by software is error-prone since it is hard to guarantee that no unwarranted assumptions were used. For this reason, we encoded our proof in machine-checkable form, using the proof assistant Agda [1].

3.1 Statement of the main theorem

Theorem 3.1.1. *The 2-qubit Clifford+T group is presented by $(\mathcal{X}, \Gamma)$, where the set of generators is*

$$\mathcal{X} = \{\omega, H_0, H_1, S_0, S_1, T_0, T_1, CZ\},$$

and the set of relations Γ is shown in Figure 3.1.

In relations (B18)–(B20), we have used a number of abbreviations; these are defined in Figure 3.2. The empty word is denoted ϵ.

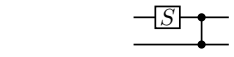
(a) Monoidal relations:

$$\omega A = A\omega, \quad \text{where } A \in \{H_i, S_i, T_i, CZ\} \tag{B1}$$
$$A_0 B_1 = B_1 A_0, \quad \text{where } A, B \in \{H, S, T\} \tag{B2}$$

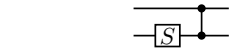
(b) Order of Clifford group elements:

$$\omega^8 = \epsilon \tag{B3}$$
$$H_i^2 = \epsilon \tag{B4}$$
$$S_i^4 = \epsilon \tag{B5}$$
$$(S_i H_i)^3 = \omega \tag{B6}$$
$$CZ^2 = \epsilon \tag{B7}$$

(c) Remaining Clifford relations:

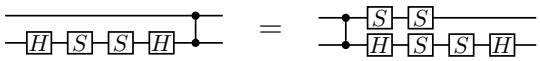


(B8)

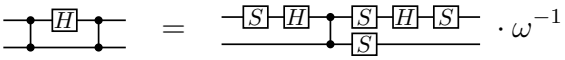


(B9)

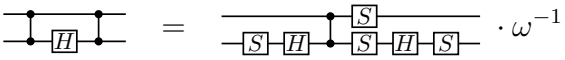
(B10)



(B11)



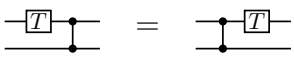
(B12)



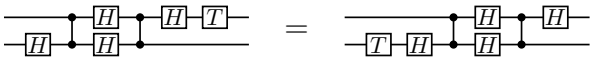
(B13)

(d) "Obvious" relations involving T:

$$T_i^2 = S_i \tag{B14}$$
$$(T_i H_i S_i S_i H_i)^2 = \omega \tag{B15}$$



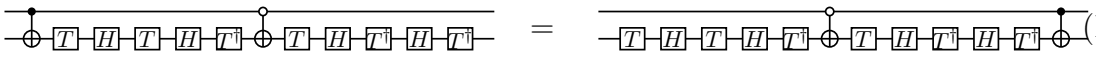
(B16)



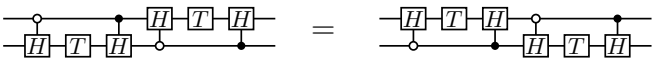
(B17)

(e) "Non-obvious" relations involving T:

(B18)



(B19)



(B20)

Figure 3.1: Relations for 2-qubit Clifford+T operators. Here $i \in \{0, 1\}$.

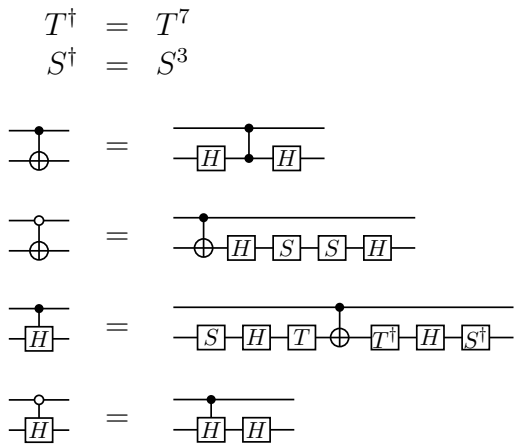


Figure 3.2: Abbreviations used in circuit notations.

3.2 Proof outline

In a nutshell, the proof can be described in a few sentences. It proceeds as follows. Let $R = \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ be the smallest subring of the complex numbers containing $\frac{1}{\sqrt{2}}$ and i, and let $G = U_4(R)$ be the group of unitary 4×4-matrices with entries in R. Then it is clear that $\mathcal{CT}(2)$ is a subgroup of G, because all of its generators belong to G. Moreover, from [16], it is known that $\mathcal{CT}(2)$ is precisely equal to the subgroup of G consisting of matrices whose determinant is a power of i. A presentation of G by generators and relations was given by Greylyn [21]. There is a general procedure, called the Reidemeister-Schreier procedure [31, 32], for finding generators and relations of a subgroup, given generators and relations of the supergroup. Applying this procedure therefore yields a complete set of relations for $\mathcal{CT}(2)$.

While in principle, the above proof outline suffices to prove Theorem 3.1.1, in practice there is a large amount of non-trivial work involved in generating and simplifying the actual relations. For this reason, we have formalized Theorem 3.1.1 and its proof in the proof assistant Agda. This allows the proof to be independently checked without too much manual work.

3.3 The Reidemeister-Schreier theorem for monoids

The Reidemeister-Schreier theorem is a theorem in group theory that allows one to derive a complete set of relations for a subgroup from a complete set of relations for

the supergroup, given enough information about the cosets. We will use a version of the Reidemeister-Schreier theorem that works for monoids, which we now describe. To our knowledge, this monoid formulation of the Reidemeister-Schreier theorem does not appear in the literature.

Definition 3.3.1. Given sets X, Y and a function $f : X \to Y^*$, let $f^* : X^* \to Y^*$ be the unique monoid homomorphism extending f. Concretely, f^* is given by $f^*(x_1 \dots x_n) = f(x_1) \cdot \ldots \cdot f(x_n)$.

More generally, given sets C, X, Y and a function $f : C \times X \to Y^* \times C$, let $f^{**} : C \times X^* \to Y^* \times C$ be the function defined by $f^{**}(c_0, x_1 \dots x_n) = (w_1 \cdot \ldots \cdot w_n, c_n)$, where $f(c_{i-1}, x_i) = (w_i, c_i)$ for all $i = 1, \dots, n$.

Note that in case C is a singleton, the functions f^* and f^{**} are essentially the same. In general, the difference is that f^{**} also keeps a "state" in the form of an element of C.

Theorem 3.3.2 (Reidemeister-Schreier theorem for monoids)**.** *Let X and Y be sets, and let Γ and Δ be sets of relations over X and Y, respectively. Suppose that the following additional data is given:*

- *a set C with a distinguished element $I \in C$,*
- *a function $f : X \to Y^*$,*
- *a function $h : C \times Y \to X^* \times C$,*

subject to the following conditions:

(a) For all $x \in X$, if $h^{**}(I, f(x)) = (v, c)$, then $v \sim_\Gamma x$ and $c = I$.

(b) For all $c \in C$ and $w, w' \in Y^*$ with $(w, w') \in \Delta$, if $h^{**}(c, w) = (v, c')$ and $h^{**}(c, w') = (v', c'')$ then $v \sim_\Gamma v'$ and $c' = c''$.

Then for all $v, v' \in X^*$, $f^*(v) \sim_\Delta f^*(v')$ implies $v \sim_\Gamma v'$.

To better understand the utility of this theorem, let us briefly provide some context. First, we note that we will be using this theorem in the case where G is a monoid, H is a submonoid of G, (Y, Δ) is a presentation of G, X is a set of generators for H, and we wish to show that some proposed set of relations Γ is complete

for H. Assuming that all hypotheses of Theorem 3.3.2 are satisfied, and further assuming that f represents the inclusion function of H into G, i.e., that for all $x \in X$, $[f(x)]_G = [x]_H$, the completeness of Γ then follows. Namely, $[v]_H = [v']_H$ implies $[f^*(v)]_G = [f^*(v')]_G$, which implies $f^*(v) \sim_\Delta f^*(v')$ by completeness of Δ, which implies $v \sim_\Gamma v'$ by Theorem 3.3.2.

To see how the theorem works, it is useful to further concentrate on the case where G and H are groups, although the theorem itself does not require this. In the case of groups, one would typically consider the set $H\backslash G = \{Hc \mid c \in G\}$ of right cosets of H in G, and one would let C be a set of chosen coset representatives. The function f is then chosen to assign to each $x \in X$ some word $w \in Y^*$ such that $[x]_H = [w]_G$. The function h is chosen to assign to each pair of a coset representative $c \in C$ and generator $y \in Y$ the unique coset representative $c' \in C$ and some word $v \in X^*$ such that $c[y]_G = [v]_H c'$. Conditions (a) and (b) are then sufficient for the set of relations Γ to be complete. In the more general case of monoids, G is not necessarily partitioned into cosets, but the method works anyway, provided that appropriate C, f, and h can be chosen.

Proof of Theorem 3.3.2. Let us say that a word $w \in Y^*$ is *special* if $h^{**}(I, w) = (v, I)$ for some $v \in X^*$. Let Y_s^* be the set of special words. By definition of h^{**}, the empty word is special and special words are closed under concatenation, so Y_s^* is a submonoid of Y^*. Moreover, the image of f is special by property (a), and therefore the image of f^* is also special. Finally, there is a translation back from special words in Y to words in X: define $g : Y_s^* \to X^*$ by letting $g(w) = v$ where $h^{**}(I, w) = (v, I)$. Clearly, g is a monoid homomorphism.

Claim A: for all $v \in X^*$, we have $v \sim_\Gamma g(f^*(v))$. Proof: Since both g and f^* are monoid homomorphisms and $\sim_\Gamma$ is a congruence, it suffices to show this in the case when $v \in X$ is a generator. But in that case, it holds by assumption (a).

Claim B: for all $w, w' \in Y^*$ and $c \in C$, if $w \sim_\Delta w'$ and $h^{**}(c, w) = (v, d)$ and $h^{**}(c, w') = (v', d')$, then $v \sim_\Gamma v'$ and $d = d'$. Proof: define a relation $\sim$ on Y^* by $w \sim w'$ if for all $c \in C$, $h^{**}(c, w) = (v, d)$ and $h^{**}(c, w') = (v', d')$ implies $v \sim_\Gamma v'$ and $d = d'$. We must show that $w \sim_\Delta w'$ implies $w \sim w'$. Since $\sim_\Delta$ is, by definition, the smallest congruence containing Δ, it suffices to show that $\sim$ is a congruence containing Δ. The fact that $\sim$ is reflexive, symmetric, and transitive is obvious from

its definition. The fact that it is a congruence follows from the definition of h^{**} and the fact that $\sim_\Gamma$ is a congruence. Finally, $\sim$ contains Δ by assumption (b).

Note that, as a special case of claim B, we also have the following: if $w, w' \in Y_s^*$ are special words, then $w \sim_\Delta w'$ implies $g(w) \sim_\Gamma g(w')$. This follows directly from the definition of g.

To finish the proof of the Reidemeister-Schreier theorem, let $v, v' \in X^*$ and assume that $f^*(v) \sim_\Delta f^*(v')$. Then we have:

$$v \ \sim_\Gamma \ g(f^*(v)) \ \sim_\Gamma \ g(f^*(v')) \ \sim_\Gamma \ v',$$

where the first and last congruence holds by claim A, and the middle one holds by the special case of claim B. Therefore, $v \sim_\Gamma v'$ as claimed. □

Corollary 3.3.3. *Let G be a monoid with presentation (Y, Δ), where $Y \subseteq G$. Suppose $H \subseteq G$ is a submonoid and X is a set of generators for H. Let Γ be a set of valid relations for H. Assume a set C and functions f and h are given, satisfying the hypotheses of Theorem 3.3.2, and assume that f represents the inclusion function of H into G, i.e., that $x \in X$, $[f(x)]_G = [x]_H$. Then Γ is a complete set of relations for H.* □

3.4 Pauli rotation representation

One of the problems we face in applying the Reidemeister-Schreier theorem is that we must show that a large number of (computer-generated) Clifford+T relations follow from the relations in Figure 3.1. It would be very useful if this task could be automated. Ideally, the relations in Figure 3.1 could be turned into a set of rewrite rules with the property that every Clifford+T circuit can be rewritten to a unique *normal form*; in that case, to show that a given relation follows from the ones in Figure 3.1, it would be sufficient to reduce the left-hand and right-hand sides to normal form and check that they are equal.

Unfortunately, no such rewrite system or normal form is known. Instead, the best we can do is a semi-automated process in which words are rewritten to something that is "almost" a normal form, i.e., not quite unique, but close enough so that many relations can be proved automatically, and the rest are more easily solvable by hand.

For this, the *Pauli rotation representation* of Clifford+T operators turns out to be useful. This representation was first described in [20, Section 3].

Recall that a Pauli operator is an operator of the form $\lambda P_1 \otimes ... \otimes P_n$ where $\lambda \in \{\pm 1, \pm i\}$ and $P_1, ..., P_n \in \{I, X, Y, Z\}$ [34]. Here $Z = SS$, and $Y = iXZ$. Note that a Pauli operator is self-adjoint if and only if $\lambda = \pm 1$. We say a self-adjoint Pauli operator is positive if $\lambda = 1$.

We start by noting that the T-gate is a linear combination of the identity I and the Pauli operator Z. Specifically:

$$T \;=\; \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \;=\; \frac{1+\omega}{2} I + \frac{1-\omega}{2} Z. \tag{3.1}$$

Therefore, an operator A commutes with T if and only if it commutes with Z. More generally, given any n-qubit non-trivial self-adjoint Pauli operator P, define

$$R_P \;=\; \frac{1+\omega}{2} I + \frac{1-\omega}{2} P. \tag{3.2}$$

Note that $R_Z = T$. We refer to the operators R_P as *(45 degree) Pauli rotations*. Note that R_P is not a Pauli operator; we call it a Pauli rotation because it is a rotation about a Pauli axis. By (3.2), it is again obvious that an operator A commutes with R_P if and only if it commutes with P. Moreover, from (3.2), we get the following fundamental property of Pauli rotations:

$$CPC^{-1} = Q \quad \text{if and only if} \quad CR_PC^{-1} = R_Q. \tag{3.3}$$

Let $Z_{(i)} = I \otimes \ldots \otimes I \otimes Z \otimes I \otimes \ldots \otimes I$ be the n-qubit Pauli operator with Z acting on the ith qubit, and similarly $T_{(i)} = I \otimes \ldots \otimes I \otimes T \otimes I \otimes \ldots \otimes I = R_{Z_{(i)}}$. Since the Clifford operators act transitively on the set of non-trivial self-adjoint Pauli operators by conjugation, for every such n-qubit Pauli operator P, there exists a (non-unique) Clifford operator C such that $CZ_{(1)}C^{-1} = P$, and therefore $CT_{(1)}C^{-1} = R_P$. We therefore see that all of the Pauli rotations are Clifford conjugates of the $T_{(1)}$-gate.

Next, we note that every Clifford+T operator can be written as a product of Pauli rotations followed by a single Clifford operator. Specifically, by definition, every Clifford+T operator can be written as

$$C_1 T_{(i_1)} C_2 T_{(i_2)} C_3 \cdots C_n T_{(i_n)} C_{n+1}.$$

For all k, let $D_k = C_1 C_2 \cdots C_k$, so that $C_k = D_{k-1}^{-1} D_k$. Then the above can be rewritten as

$$\begin{aligned} & C_1 T_{(i_1)} C_2 T_{(i_2)} C_3 \cdots C_n T_{(i_n)} C_{n+1} \\ = \; & C_1 R_{Z_{(i_1)}} C_2 R_{Z_{(i_2)}} C_3 \cdots C_n R_{Z_{(i_n)}} C_{n+1} \\ = \; & D_1 R_{Z_{(i_1)}} D_1^{-1} D_2 R_{Z_{(i_2)}} D_2^{-1} D_3 \cdots D_{n-1}^{-1} D_n R_{Z_{(i_n)}} D_n^{-1} D_{n+1} \\ = \; & R_{D_1 Z_{(i_1)} D_1^{-1}} R_{D_2 Z_{(i_2)} D_2^{-1}} \cdots R_{D_n Z_{(i_n)} D_n^{-1}} D_{n+1} \\ = \; & R_{P_1} R_{P_2} \cdots R_{P_n} D_{n+1}, \end{aligned}$$

where $P_k = D_k Z_{(i_k)} D_k^{-1}$. Therefore, every Clifford+T operator can be written as a product of Pauli rotations followed by a single Clifford operator, as claimed. It also shows that the number of required Pauli rotations is at most equal to the T-count of the original circuit. In fact, since every Pauli rotation has T-count 1, it is clear that every product of n Pauli rotations can be converted to a circuit of T-count n, and vice versa. In particular, the minimal T-count of a circuit is equal to the minimal number of Pauli rotations required to express it.

The Pauli rotation representation is not unique. There are some obvious relations:

(a) R_P and R_Q commute if and only if P and Q commute. This follows from (3.2).

(b) For any P, the operator R_P^2 is Clifford, and therefore can be eliminated, resulting in a shorter word. To see why, recall that there exists a Clifford operator C such that $R_P = C T_{(1)} C^{-1}$; therefore $R_P^2 = C T_{(1)}^2 C^{-1}$. Since $T_{(1)}^2 = S_{(1)}$ is a Clifford gate, it follows that R_P^2 is Clifford.

(c) For any non-trivial positive Pauli operator P, there exists a Clifford operator D such that $R_{(-P)} = R_P D$. Indeed, let C be a Clifford operator such that $P = C Z_{(1)} C^{-1}$. Then $-P = C(-Z_{(1)}) C^{-1} = C X_{(1)} Z_{(1)} X_{(1)} C^{-1}$. Therefore $R_{(-P)} = C X_{(1)} T_{(1)} X_{(1)} C^{-1}$. Using the relation $XTX = TS^\dagger \omega$, we have

$$R_{(-P)} = C T_{(1)} S_{(1)}^\dagger \omega C^{-1} = C T_{(1)} C^{-1} C S_{(1)}^\dagger \omega C^{-1} = R_P C S_{(1)}^\dagger \omega C^{-1}.$$

Thus, the claim holds with $D = C S_{(1)}^\dagger \omega C^{-1}$.

It is relatively easy to standardize the Pauli rotation representation modulo the above three relations: First, we eliminate any generators of the form $R_{(-P)}$ where P is a

non-trivial positive Pauli operator. This can be done from left to right, using relations from (c); the resulting Clifford operator can be shifted all the way to the end of the word using relations of the form $DR_P = R_Q D$, where $Q = DPD^{-1}$, see (3.3). Next, we use relations from (a) to swap adjacent generators when possible, for example arriving at the lexicographically smallest word that is equal to the given word up to such commuting permutations. Next, we use relations from (b) to remove any duplicates. Should there be any such duplicates, the resulting word will need to be standardized again, but since it uses fewer Pauli rotations, the process eventually terminates.

However, even when the Pauli rotation representation is standardized modulo the relations (a), (b), and (c), it is still not unique. Indeed, there are some "non-obvious" relations. In a sense, the contribution of this chapter is to state exactly what these non-obvious relations are. They turn out to be the following. Here, for brevity, we have omitted the tensor symbol $\otimes$, i.e., we wrote R_{IX} instead of $R_{I\otimes X}$.

$$\begin{aligned} R_{IX}R_{IZ}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{ZZ}R_{IX}, \\ R_{IX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{IX}, \\ R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ} &= R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ}R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}. \end{aligned}$$

These turn out to be equivalent to relations (B18), (B19), and (B20) in Figure 3.1, respectively. We will address the question of what these relations might "mean" (i.e., how one might be able to see that they are true without computing the matrices) in Section 3.7.

3.5 Soundness and completeness

Our goal is to prove that Theorem 2.2.1 implies Theorem 3.1.1. Recall that Greylyn's set of generators for $U_4(R)$ is $\mathcal{Y} = \{\omega_{[j]}, X_{[j,k]}, H_{[j,k]} \mid j, k \in \{1, ..., 4\} \text{ and } j < k\}$. Also recall that our target set of generators for $\mathcal{CT}(2)$ is $\mathcal{X} = \{\omega, H_0, H_1, S_0, S_1, T_0, T_1, CZ\}$. We fix a translation from $\mathcal{X}$ to $\mathcal{Y}^*$ as in Figure 3.3. We prove the following soundness and completeness theorems for this translation:

Theorem 3.5.1 (Soundness)**.** *For all $w, v \in \mathcal{X}^*$, $w \sim_\Gamma v$ implies $f^*(w) \sim_\Delta f^*(v)$.*

Theorem 3.5.2 (Completeness)**.** *For all $w, v \in \mathcal{X}^*$, $f^*(w) \sim_\Delta f^*(v)$ implies $w \sim_\Gamma v$.*

As already noted in Section 3.3, these two theorems, together with Theorem 2.2.1, immediately imply Theorem 3.1.1. Specifically, we have $w \sim_\Gamma v$ if and only if

$$
\begin{aligned}
f(\omega) &= \omega_{[0]}\omega_{[1]}\omega_{[2]}\omega_{[3]},\\
f(H_0) &= H_{[1,3]}H_{[0,2]},\\
f(H_1) &= H_{[2,3]}H_{[0,1]},\\
f(S_0) &= \omega_{[2]}^2\omega_{[3]}^2,\\
f(S_1) &= \omega_{[1]}^2\omega_{[3]}^2,\\
f(T_0) &= \omega_{[2]}\omega_{[3]},\\
f(T_1) &= \omega_{[1]}\omega_{[3]},\\
f(CZ) &= \omega_{[3]}^4.
\end{aligned}
$$

Figure 3.3: Translation from $\mathcal{X}$ to $\mathcal{Y}^*$.

$f^*(w) \sim_\Delta f^*(v)$ if and only if $[f^*(w)] = [f^*(v)]$ if and only if $[w] = [v]$, where the first equivalence follows from Theorems 3.5.1 and 3.5.2, the second equivalence follows from Theorem 2.2.1, and the last equivalence holds because the function f respects the interpretation.

3.6 The formal proof

Soundness and completeness are formally proved in the Agda code accompanying the paper [9] this chapter is based on. We organized the code to make it hopefully as easy as possible to verify the result. The code consists of 67 files that are listed in Figure 3.4, and which we now briefly describe.

(a) Background. The eight files in the "background" section contain general-purpose definitions of the kind that are usually found in the Agda standard library, i.e., basic properties of booleans, integers, equality, propositional connectives, etc. The reason we did not use the actual Agda standard library is that it is very large and changes frequently. We felt that it is better for our code to be self-contained rather than depending on a particular library version.

(b) Statement of the result. In these two files, we give a minimal set of definitions that allows us to *state* the soundness and completeness theorems. The file `Word.agda` defines what it means to be a word over a set of generators, as well as the inference rules we use for deriving relations from a set of axioms (such as reflexivity, symmetry, transitivity, congruence, associativity, and the left and right unit laws). Note that in

the Agda code, we define a word as a term in the language of monoids, rather than as a sequence of generators. In other words, associativity and the unit laws are treated as laws, rather than being built into the definition. The file `Word.agda` also defines the f^* operation used in the statement of the soundness and completeness theorems. The file `Generator.agda` defines the Clifford+T generators and the relations from Figure 3.1, Greylyn's generators and the relations from Figure 2.1, and the translation function f from Section 3.5. It also contains the statement of the soundness and completeness theorems, but not their proofs. The reason we state these theorems separately from their proofs is to make sure that Agda (and a human reviewer) can verify that the statement of these theorems only depends on the relatively small number of definitions given so far, and not on the much larger number of definitions and tactics used in the proof.

(c) Details of the proof. The proof of the soundness and completeness theorems relies on a large number of auxiliary definitions and lemmas, and comprises the bulk of our code with 56 files. This includes a formal proof of the Reidemeister-Schreier theorem; several tactics for automating steps in certain equational proofs; a simplified presentation of Greylyn's generators and relations, using only 5 generators and 19 relations (instead of Greylyn's original 16 generators and 123 relations), along with the proof of its completeness; a formalization of Pauli rotations and their relevant properties; as well as 46 step-by-step proofs of individual relations. These details are primarily intended to be machine-readable, and can safely be skipped by readers who trust Agda and merely want to check the proof rather than reading it. However, all of the files are documented and human-readable.

The relations in the files `Equation1.agda` to `Equation46.agda` are at the heart of the completeness proof. These are the relations that must be proved to satisfy the hypotheses of the Reidemeister-Schreier theorem. Some of these relations are trivial, such as `Equation13.agda`. Others are highly non-trivial and require almost a thousand proof steps, such as `Equation44.agda`. In particular, the proofs that require relations (B18)–(B20) from Figure 3.1 tend to be non-obvious; in fact, this is how we discovered relations (B18)–(B20) in the first place. We did not write these equational proofs by hand; instead, we used a semi-automated process where most of

the proofs were generated by a separate Haskell program and output in a format that is convenient and efficient for Agda to check. Originally, we also attempted to write Agda tactics that would allow Agda to derive these relations fully automatically; however, this failed due to performance issues with Agda.

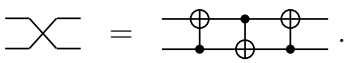
(d) Proof witness. Finally, the file `Proof.agda` contains nothing but a witness of the fact that the soundness and completeness theorems have been formally proven. A reader who wants to skip the details of the formal proof only needs to check two things: the statement of the main theorem in `Generator.agda` (to make sure the statement correctly captures what we said it does), and the fact that the Agda proof checker accepts `Proof.agda`.

3.7 Discussion of the axioms

Here, we give some further perspectives on what the axioms of Figure 3.1 might "mean", and in particular, how one might convince oneself that the relations are true without having to compute the corresponding matrices.

Note that we are not claiming that axioms (B1)–(B20) are independent; for example, (B8) clearly follows from (B14) and (B16); however, we found it useful to separate the Clifford relations from the rest, which is why (B8) was included. It would be nice to know whether axioms (B18)–(B20) are independent from the others and from each other, and this seems likely to be true, but we do not know.

The axioms in groups (a)–(c) are well-known; they merely express the Clifford relations [34] and the fact that operators on disjoint qubits commute. Relations (B14) and (B15) express the well-known facts that $T^2 = S$ and $(TX)^2 = \omega$, whereas relation (B16) holds because diagonal operators commute. Note that the upside-down version of relation (B16) was not included among our axioms; this is because it is actually derivable from the remaining axioms. Relation (B17) becomes obvious once one realizes that the swap gate can be expressed as a sequence of three controlled not-gates:



Relation (B17) is then obtained by simplifying the following, which expresses the fact

(a) Background:

`Boolean.agda`	The type of booleans.
`Proposition.agda`	Basic definitions in propositional logic.
`Equality.agda`	Basic properties of equality.
`Decidable.agda`	Some definitions to deal with decidable properties.
`Inspect.agda`	Agda's "inspect" paradigm, to assist with pattern matching.
`Nat.agda`	Basic properties of the natural numbers.
`Maybe.agda`	The "Maybe" type.
`List.agda`	Basic properties of lists.

(b) Statement of the result

`Word.agda`	Basic properties of words.
`Generator.agda`	Generators and relations for our two groups, and statement of main theorem.

(c) Proof of the result

`Word-Lemmas.agda`	Lemmas about monoids and groups, and equational reasoning.
`Reidemeister-Schreier.agda`	Two versions of the Reidemeister-Schreier theorem.
`Word-Tactics.agda`	Some tactics for proving properties of words.
`Clifford-Lemmas.agda`	A decision procedure for equality of 2-qubit Clifford operators.
`CliffordT-Lemmas.agda`	Properties and tactics for Clifford+T operators.
`Greylyn-Lemmas.agda`	Some automation for Greylyn's 1- and 2-level operators.
`Soundness.agda`	Proof of soundness.
`Greylyn-Simplified.agda`	A smaller set of generators and relations for Greylyn's operators.
`PauliRotations.agda`	Definitions, properties, and tactics for Pauli rotations.
`Equation(1-46).agda`	Explicit proofs of 46 relations required for completeness.
`Completeness.agda`	Proof of completeness.

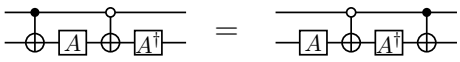
(d) Top-level proof witness

`Proof.agda`	The final witness for soundness and completeness.

Figure 3.4: List of Agda files. The files are listed in order of dependency, i.e., each file only imports earlier files.

that a T-gate can be moved past a swap-gate:

We will now focus on the "non-obvious" relations (B18)–(B20). Relations (B18) and (B19) are of the form

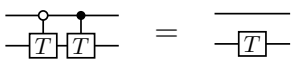


(3.4)

They hold because positively controlled gates commute with negatively controlled gates. Note that there are infinitely many relations of the form (3.4), where A is any single-qubit Clifford+T operator, but our completeness proof shows that, in the presence of the remaining axioms, two of them are sufficient to prove all the others.

Relation (B20) is more interesting. It, too, states that two operators commute, but it is less obvious why this is so. Ideally, we would be able to find some simpler and more obvious relations that imply (B20). While we have not been able to find such simpler relations in the Clifford+T generators, we can do this if we permit ourselves a controlled T-gate. Note that the controlled T-gate is not itself a member of the 2-qubit Clifford+T group, since representing it as a Clifford+T operator requires an ancilla [16]. But the use of controlled T-gates is nevertheless helpful in explaining relation (B20). We start by noting that the controlled T-gate satisfies the following obvious circuit identities (and their upside-down versions):

(3.5)



(3.6)

(3.7)

(3.8)

Identities (3.5)–(3.7) are obvious because all of the operators in them are diagonal. Identity (3.8) holds by case distinction: this circuit applies either HT or TH to the bottom qubit, depending on whether the top qubit is $|0\rangle$ or $|1\rangle$. Using these identities, we can easily prove (B20) as in Figure 3.5.

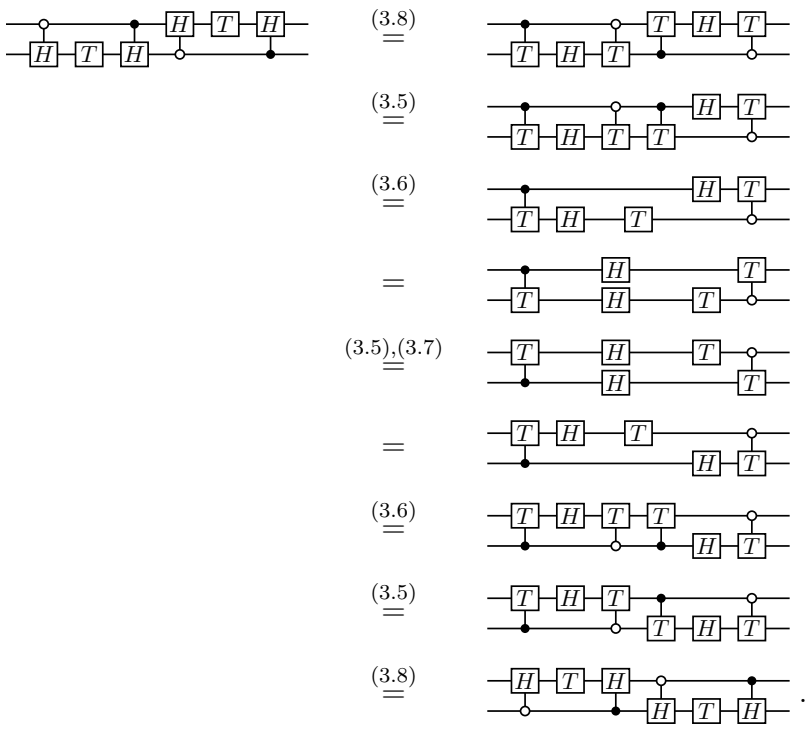


Figure 3.5: Relation (B20) is derivable using controlled T-gate.

Note that there is again an infinite family of such relations, because in the above derivation, we could have used any gate in place of H. However, due to completeness, all other such relations are consequences of (B18)–(B20) and the remaining axioms.

Another way to look at relations (B18)–(B20) is in terms of their Pauli rotation representations. As we already mentioned in Section 3.4, up to basis changes, the three relations can be written in terms of Pauli rotations, respectively as follows:

$$\begin{aligned} R_{IX}R_{IZ}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{ZZ}R_{IX}, \\ R_{IX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{ZX} &= R_{ZX}R_{IZ}R_{IX}R_{ZX}R_{ZZ}R_{IX}, \\ R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ} &= R_{YX}R_{ZY}R_{ZX}R_{XI}R_{IZ}R_{XY}R_{YZ}R_{XZ}R_{IX}R_{ZI}. \end{aligned}$$

When written in this form, the first two of these relations only use X and Z Paulis, and use only Z on the left qubit. This indicates that these relations are about controlled gates. We can also see that in both cases, the relation exchanges the positions of the leftmost R_{IX} and the rightmost R_{ZX}. The first relation can also be seen to express the fact that $R_{IZ}R_{ZZ}$ commutes with $R_{ZX}R_{IX}^{-1}$, and similarly for the second relation. The third relation again takes the form of an operator commuting with its upside-down version.

Chapter 4

Presentation for $U_n(\mathbb{Z}[\frac{1}{2}, i])$

4.1 Statement of the main theorem

Let $\mathcal{G}$ be the set of all one- and two-level matrices of types X, K, and i. We refer to the elements of $\mathcal{G}$ as the *generators*. Specifically, they are:

- $X_{[j,\ell]}$, for $0 \leq j < \ell < n$;
- $K_{[j,\ell]}$, for $0 \leq j < \ell < n$; and
- $i_{[j]}$, for $0 \leq j < n$.

Recall that $\mathbb{D} = \mathbb{Z}[\frac{1}{2}]$ and $\mathbb{D}[i] = \mathbb{Z}[\frac{1}{2}, i]$. The purpose of this chapter is to prove the following theorem:

Theorem 4.1.1 (Main theorem)**.** *Let Δ be the set of relations shown in Figure 4.1. Then $(\mathcal{G}, \Delta)$ is a presentation of $U_n(\mathbb{D}[i])$. In other words, the relations in Figure 4.1 are sound and complete for $U_n(\mathbb{D}[i])$.*

In Figure 4.1, relations (C1)–(C3) give the order of the generators. They also ensure that all of the generators (and therefore all words of generators) are invertible. Relations (C4)–(C9) state that generators with disjoint indices commute. Relations (C10)–(C14) state that $X_{[j,k]}$ can be used to swap indices j and k in any other generator. Finally, relations (C15)–(C17) state additional properties of the generators. We note that the relations in Figure 4.1 are not minimal; for example, (C1) and (C18) imply (C3). However, we think that they are a "nice" set of relations.

To prove the soundness part of Theorem 4.1.1, by Remark 2.3.5, it suffices to check that each relation in Δ is true in $U_n(\mathbb{D}[i])$. This can be verified by calculation. The remainder of this chapter is devoted to proving completeness.

For convenience, we say that two words w, u are *relationally equivalent* if $w \sim_\Delta u$ and *semantically equivalent* if $[w] = [u]$. Completeness is thus the statement that

(a) Order of generators:

$$i_{[j]}^4 = \varepsilon \tag{C1}$$
$$X_{[j,k]}^2 = \varepsilon \tag{C2}$$
$$K_{[j,k]}^8 = \varepsilon \tag{C3}$$

(b) Disjoint generators commute:

$$i_{[j]}i_{[k]} = i_{[k]}i_{[j]} \tag{C4}$$
$$i_{[j]}X_{[k,\ell]} = X_{[k,\ell]}i_{[j]} \tag{C5}$$
$$i_{[j]}K_{[k,\ell]} = K_{[k,\ell]}i_{[j]} \tag{C6}$$
$$X_{[j,k]}X_{[\ell,m]} = X_{[\ell,m]}X_{[j,k]} \tag{C7}$$
$$X_{[j,k]}K_{[\ell,m]} = K_{[\ell,m]}X_{[j,k]} \tag{C8}$$
$$K_{[j,k]}K_{[\ell,m]} = K_{[\ell,m]}K_{[j,k]} \tag{C9}$$

(c) X permutes indices:

$$i_{[k]}X_{[j,k]} = X_{[j,k]}i_{[j]} \tag{C10}$$
$$X_{[k,\ell]}X_{[j,k]} = X_{[j,k]}X_{[j,\ell]} \tag{C11}$$
$$X_{[j,\ell]}X_{[k,\ell]} = X_{[k,\ell]}X_{[j,k]} \tag{C12}$$
$$K_{[k,\ell]}X_{[j,k]} = X_{[j,k]}K_{[j,\ell]} \tag{C13}$$
$$K_{[j,\ell]}X_{[k,\ell]} = X_{[k,\ell]}K_{[j,k]} \tag{C14}$$

(d) Relations for K:

$$K_{[j,k]}i_{[k]}^2 = X_{[j,k]}K_{[j,k]} \tag{C15}$$
$$K_{[j,k]}i_{[k]}^3 = i_{[k]}K_{[j,k]}i_{[k]}K_{[j,k]} \tag{C16}$$
$$K_{[j,k]}i_{[j]}i_{[k]} = i_{[j]}i_{[k]}K_{[j,k]} \tag{C17}$$
$$K_{[j,k]}^2 i_{[j]}i_{[k]} = \varepsilon \tag{C18}$$
$$K_{[j,k]}K_{[\ell,m]}K_{[j,\ell]}K_{[k,m]} = K_{[j,\ell]}K_{[k,m]}K_{[j,k]}K_{[\ell,m]} \tag{C19}$$

Figure 4.1: A sound and complete set of relations for $U_n(\mathbb{Z}[\frac{1}{2}, i])$. In each relation, the indices are assumed to be distinct; moreover, whenever a generator $X_{[a,b]}$ or $K_{[a,b]}$ is mentioned, we assume $a < b$.

all semantically equivalent words are relationally equivalent, and soundness is the converse.

4.2 The exact synthesis algorithm

The concepts and results of this section are similar to those used in [17, 21, 2, 28].

Let $U_n(\mathbb{D}[i])$ be the group of unitary matrices with entries in $\mathbb{D}[i]$. In this section, we will show that every element of $U_n(\mathbb{D}[i])$ can be written as a product of one- and two-level matrices of the form $i_{[j]}, X_{[j,k]}$, and $K_{[j,k]}$, where i is the imaginary unit and the matrices X and K are the ones that were defined in the introduction. In other words, we will show that these matrices are *generators* for the group $U_n(\mathbb{D}[i])$. We will do so by exhibiting a particular algorithm that inputs a matrix $U \in U_n(\mathbb{D}[i])$ and outputs a corresponding word in the generators. In quantum computing, such an algorithm is called an *exact synthesis algorithm* (as opposed to an approximate synthesis algorithm, which only approximates a unitary matrix up to some given ε). The algorithm in this section is adapted from [17] and [2].

Remark 4.2.1. A slight technical inconvenience arises because the matrix K is not self-inverse. Therefore, in the following presentation we sometimes use K as a generator and sometimes its inverse $K^\dagger$. We have carefully chosen which one to use in each instance, to make the proofs in the later parts of the paper as simple as possible. However, readers who wish to ignore the difference between K and $K^\dagger$ can safely do so, because in any case we have the relation $K^\dagger = iK$, so whatever is generated by K is also generated by $K^\dagger$ and vice versa.

We start with a number of lemmas. Since $\mathbb{Z}[i]$ and $\mathbb{D}[i]$ are subrings of $\mathbb{C}$, the usual properties of complex numbers, complex vectors, and complex matrices, such as complex conjugation, inner product, norm, and conjugate transposition are naturally inherited. For example, for a vector $v \in \mathbb{D}[i]^n$, the *norm* of v is $\|v\| = \sqrt{v^\dagger v}$, where $v^\dagger$ is the conjugate transpose of v. Note that $\|v\|^2 = v^\dagger v \in \mathbb{D}$, since $v^\dagger v \in \mathbb{D}[i] \cap \mathbb{R}$. Similarly if $v \in \mathbb{Z}[i]$, then $v^\dagger v \in \mathbb{Z}$. A *unit vector* is a vector of norm 1.

If v is a vector, the notation v_j refers to its jth entry; entries are numbered starting from $j = 0$.

Lemma 4.2.2. *Let v be a unit vector in $\mathbb{Z}[i]^n$. Then v has exactly one non-zero entry, and that entry is a power of i.*

Proof. First note that for $\alpha = a + bi \in \mathbb{Z}[i]$, $\|\alpha\|^2 = a^2 + b^2$ is a non-negative integer, and $\|\alpha\| = 0$ if and only if $\alpha = 0$. Let $v = [v_0, v_1, ..., v_{n-1}]^T$. By assumption, $\sum_{j=0}^{n-1} \|v_j\|^2 = 1$. Since each $\|v_j\|^2$ is a non-negative integer, there is exact one non-zero v_j such that $\|v_j\| = 1$. Then it is easy to see that $v_j \in \{\pm 1, \pm i\}$. □

Corollary 4.2.3. *Let v be a unit vector in $\mathbb{Z}[i]^n$, and let $p \in \{0, \ldots, n-1\}$. Then there exists a matrix G that is a product of one- and two-level matrices of types i and X, such that $Gv = e_p$, where e_p is the pth standard basis vector.*

Proof. By Lemma 4.2.2, v has a unique non-zero entry $v_m \in \{\pm 1, \pm i\}$. Let $e \in \{0, 1, 2, 3\}$ such that $i^e v_m = 1$. Define

$$G = \begin{cases} i^e_{[m]} & \text{if } m = p, \\ X_{[m,p]} i^e_{[m]} & \text{otherwise.} \end{cases}$$

Then $Gv = e_p$ as desired. □

Recall from Section 2.3.1 that $\gamma = 1 + i$. Also recall the definitions of oddness and denominator exponents from Definitions 2.3.1 and 2.3.4.

Lemma 4.2.4. *Let v be a unit vector in $\mathbb{D}[i]^n$ with least denominator exponent $k > 0$, and let $w = \gamma^k v \in \mathbb{Z}[i]^n$. Then w has an even number of odd entries.*

Proof. We have

$$\sum_{j=0}^{n-1} \|w_j\|^2 = \|w\|^2 = |\gamma|^{2k} \|v\|^2 = (\gamma\gamma^\dagger)^k \equiv 0 \pmod{\gamma}.$$

Therefore, there are an even number of $j \in \{0, ..., n-1\}$ such that $\|w_j\|^2$ is odd. It follows by Lemma 2.3.2 that an even number of w_j are odd. □

For brevity, we sometimes write $\alpha(\gamma^\ell)$ to denote any member of the congruence class of α modulo γ^ℓ. In other words, we write $\alpha(\gamma^\ell)$ for any element of $\mathbb{Z}[i]$ of the form $\alpha + \beta\gamma^\ell$, when the value of β is not important.

Lemma 4.2.5 (Row operation)**.** *Let $v \in \mathbb{D}[i]^n$ be a vector with $lde_\gamma(v_j) = lde_\gamma(v_\ell) = k > 0$. Then there exists an exponent $q \in \{0,1\}$ such that, if we let $v' = K^\dagger_{[j,\ell]} i^q_{[\ell]} v$, then $lde_\gamma(v'_j), lde_\gamma(v'_\ell) < k$. The remaining entries of v are unchanged.*

Proof. Let $w_j = \gamma^k v_j$ and $w_\ell = \gamma^k v_\ell$. Then both w_j and w_ℓ are odd. By Lemma 2.3.3, there exists $q \in \{0,1\}$ such that $w_j \equiv i^q w_\ell \pmod{\gamma^2}$. Then we have

$$K^\dagger i^q_{[1]} \begin{bmatrix} w_j \\ w_\ell \end{bmatrix} = K^\dagger \begin{bmatrix} w_j \\ i^q w_\ell \end{bmatrix} = K^\dagger \begin{bmatrix} w_j \\ w_j + \alpha\gamma^2 \end{bmatrix} = \frac{1}{\gamma^\dagger} \begin{bmatrix} 2w_j + \alpha\gamma^2 \\ -\alpha\gamma^2 \end{bmatrix} = \begin{bmatrix} (w_j + \alpha i)\gamma \\ -\alpha i \gamma \end{bmatrix} = \begin{bmatrix} 0(\gamma) \\ 0(\gamma) \end{bmatrix}.$$

This means $\text{lde}_\gamma(K^\dagger i^q_{[1]} \begin{bmatrix} v_j \\ v_\ell \end{bmatrix}) < k$. Clearly, performing $K^\dagger_{[j,\ell]} i^q_{[\ell]}$ on v has the same effect, and does not change any entries of v besides v_j and v_ℓ, proving the lemma. □

Lemma 4.2.6 (Column lemma)**.** *Let v be a unit vector in $\mathbb{D}[i]^n$, and $p \in \{0, 1, ..., n-1\}$. Then there exists a matrix G that is a product of one- and two-level matrices of types X, $K^\dagger$, and i, such that $Gv = e_p$.*

Proof. By induction on k, the least denominator exponent of v, and nested induction on the number of entries of v with least denominator exponent k. If $k = 0$, then the result follows from Corollary 4.2.3. Otherwise, by Lemma 4.2.4, v has at least two entries with least denominator exponent k. Pick a pair of such entries. Lemma 4.2.5 yields some one- and two level matrices that decrease this pair's least denominator exponent. The result then follows by the induction hypothesis. □

Since each column of a unitary matrix is a unit vector, the column lemma naturally gives a way to "fix" every column of a unitary matrix to the jth standard basis vector.

Lemma 4.2.7 (Matrix decomposition)**.** *Let U be a unitary $n \times n$-matrix with entries in $\mathbb{D}[i]$. Then there exists a matrix G that is a product of one- and two-level matrices of types X, $K^\dagger$, and i such that $GU = I$.*

Proof. This is an easy consequence of the column lemma. Specifically, first use the column lemma to find a suitable G_1 such that the rightmost column of $G_1 U$ is e_{n-1}. Because $G_1 U$ is unitary, it is of the form

$$\left[\begin{array}{c|c} U' & 0 \\ \hline 0 & 1 \end{array}\right].$$

Now recursively find one- and two-level matrices to reduce U' to the identity matrix. □

Corollary 4.2.8. *A matrix U with entries in $\mathbb{D}[i]$ is unitary if and only if it can be written as a product of one- and two-level matrices of types X, K, and i.*

Proof. The right-to-left implication is obvious, because the relevant one- and two-level matrices are themselves unitary. The left-to-right implication follows from Lemma 4.2.7. Specifically, by Lemma 4.2.7, we can find a product G of one- and two-level matrices of types X, $K^\dagger$, and i such that $GU = I$. Then $U = G^{-1}$. Since $X^{-1} = X$, $(K^\dagger)^{-1} = K$, and $i^{-1} = i^3$, the inverse G^{-1} can be expressed as a product of one- and two-level matrices of the required types. □

Note that Corollary 4.2.8 implies part of Theorem 4.1.1, namely that $\mathcal{G}$ is a set of generators for $U_n(\mathbb{D}[i])$. The rest of this section is concerned with the completeness of the relations. Since the proof of Lemma 4.2.7 is constructive, it yields an algorithm that inputs an element of $U_n(\mathbb{D}[i])$ and outputs a sequence G of one- and two-level matrices of types X, $K^\dagger$, and i such that $GU = I$. We call this an "exact synthesis algorithm". In principle, there are many different ways to achieve this; for example, one step in the proof of the column lemma requires us to "pick a pair" of entries, and the computed sequence of generators will depend on which pair we pick. For the rest of this section, it is important to fix, once and for all, a particular deterministic method for making these choices. Therefore, we specify one such deterministic procedure in Algorithm 4.2.9.

In the following, by the *pivot column* of a matrix U, we mean the rightmost column where U differs from the identity matrix.

Algorithm 4.2.9 (Exact synthesis algorithm)**.**

INPUT: A unitary $n \times n$-matrix U with entries in $\mathbb{D}[i]$.

OUTPUT: A sequence of one- and two-level matrices $G_l, \ldots, G_1$ of types X, $K^\dagger$, and i such that $G_l \cdots G_1 U = I$.

STATE: Let M be a storage for a unitary $n \times n$-matrix, and let $\vec{G}$ be a storage for a sequence of one- and two-level matrices.

1. Set M to be U, and set $\vec{G}$ to be the empty sequence.

2. If $M = I$, stop, and output $\vec{G}$.

3. Let p be the index of the pivot column of M, and let k be the least denominator exponent of the pivot column of M.

   (a) If $k = 0$, let $\vec{S}$ be obtained by applying Corollary 4.2.3 to the pivot column and p.

   (b) If $k > 0$, Lemma 4.2.4 dictates there are an even number of entries that have least denominator exponent k. Let the indices of the first two odd entries be $j < \ell$. Let $\vec{S}$ be obtained by applying Lemma 4.2.5 to the pivot column and j, ℓ.

4. Set M to be $\vec{S}M$, and prepend $\vec{S}$ to $\vec{G}$. Go to step 2.

We refer to the $\vec{S}$ in step 3 as a *syllable*. We can also regard $\vec{G}$ as a sequence of syllables, so we can talk about the nth syllable in $\vec{G}$.

The fact that the algorithm is correct and terminating can be proved in exactly the same way as Lemma 4.2.7. However, we will provide a more explicit proof of correctness and termination, as this will be useful later in this chapter.

Definition 4.2.10. Let $M \in U_n(\mathbb{D}[i])$. The *level* of M, denoted by level(M), is defined as follows: If $M = I$, then level$(M) = (0, 0, 0)$. Otherwise, level(M) is a triple (p, k, m) where:

- p is the greatest index such that $Me_p \neq e_p$, i.e., the index of the pivot column.
- $k = \text{lde}(v)$, where v is the pivot column.
- m is the number of odd entries in $\gamma^k v$.

Note that p, k, m are natural numbers, so the set of all possible levels is a subset of $\mathbb{N}^3$. We use the lexicographic order on $\mathbb{N}^3$, defined by $(p, k, m) < (p', k', m')$ iff $p < p'$ or ($p = p'$ and $k < k'$) or ($p = p'$ and $k = k'$ and $m < m'$). This makes $\mathbb{N}^3$ into a well-ordered set.

Theorem 4.2.11. *Given a unitary matrix $U \in U_n(\mathbb{D}[i])$, Algorithm 4.2.9 outputs a finite sequence of one- and two-level matrices $G_l, \ldots, G_1$ such that $G_l \cdots G_1 U = I$.*

Proof. For correctness, it suffices to note that after the initialization in step 1, M and $\vec{G}$ are only updated in step 4, and at each point in the algorithm, we have $M = \vec{G}U$. The algorithm only stops when $M = I$, at which point we therefore have $I = \vec{G}U$. The only thing that remains to be shown is that the algorithm terminates.

We prove termination by well-ordered induction on the level of M. Specifically, we prove that after the algorithm reaches step 2, it always terminates. When $M = I$, this is trivially true. Otherwise, step 3 yields a syllable $\vec{S}$ such that $\text{level}(\vec{S}M) < \text{level}(M)$. Step 4 sets M to $\vec{S}M$, and loops back to step 2. At this point, since M now has a strictly smaller level, the algorithm terminates by the induction hypothesis. □

4.3 The Cayley graph

In order to conceptualize the proof of Theorem 4.1.1, it is useful to define a particular kind of infinite directed graph, which we call the *Cayley graph* of $U_n(\mathbb{D}[i])$. The vertices of the Cayley graph are the elements of $U_n(\mathbb{D}[i])$, i.e., unitary matrices with entries in $\mathbb{D}[i]$. The identity matrix I plays a special role and we call it the *root* of the graph. We often use letters such as s, r, t, q to denote vertices of the Cayley graph. Our Cayley graph contains two different kinds of edges:

- *Simple edges.* Simple edges are labelled by generators $G \in \mathcal{G}$ and denoted by single arrows $\xrightarrow{G}$. There is an edge $s \xrightarrow{G} t$ if and only if $Gs = t$. For the latter equation to make sense, keep in mind that G, s, and t are all elements of $U_n(\mathbb{D}[i])$.

- *Normal edges.* Recall from Section 4.2 that each word $\vec{S}$ that is produced in step 3 of Algorithm 4.2.9 is called a *syllable*. From now on, we often write N to denote a single such syllable (which may, however, be a word consisting of more than one generator). Normal edges are labelled by syllables, and are denoted by double arrows $\overset{N}{\Rightarrow}$. For every non-root vertex s of the Cayley graph, there is a unique normal edge originating at s, and it is given by $s \overset{N}{\Rightarrow} t$, where N is the syllable that Algorithm 4.2.9 produces when $M = s$, and $t = \vec{S}s$.

Note that as a consequence of these definitions, the Cayley graph has a tree structure with respect to the normal edges. Specifically, for every vertex s, there

exists a unique path of normal edges starting at s. Also note that if $s \overset{N}{\Longrightarrow} t$, then $\text{level}(t) < \text{level}(s)$, so the path of normal edges starting at any given vertex s is necessarily finite. Since the root I is the only vertex with no outgoing normal edge, every such path therefore necessarily ends at I, i.e., it is of the form $s = s_0 \overset{N_1}{\Longrightarrow} s_1 \overset{N_2}{\Longrightarrow} \ldots \overset{N_m}{\Longrightarrow} s_m = I$, for $m \geq 0$.

Definition 4.3.1. Given any matrix $U \in U_n(\mathbb{D}[i])$, its *normal word* is the word $w \in \mathcal{G}^*$ defined as follows: Let $U = s_0 \overset{N_1}{\Longrightarrow} s_1 \overset{N_2}{\Longrightarrow} \ldots \overset{N_m}{\Longrightarrow} s_m = I$ be the unique path of normal edges from U to I in the Cayley graph. Then we define w to be the concatenation $N_m \cdots N_1$, where each occurrence of $K^\dagger$ is replaced by K^7. Equivalently, w is the word output by Algorithm 4.2.9 on input U. Note that, by definition of the Cayley graph or equivalently by Theorem 4.2.11, we have $[w]U = I$, or in other words $[w] = U^{-1}$. We write $w = \text{normal}(U)$ when w is the normal word of U.

Moreover, given any word $u \in \mathcal{G}^*$, not necessarily normal, define its *normal form* to be $\text{normal}([u]^{-1})$. Note that by definition, every word u has a unique normal form, and if w is the normal form of u, then $[w] = [u]$. Moreover, if $[u] = [u']$ then u and u' have the same normal form. We write $\text{NF}(u)$ for the normal form of a word u.

Given a path $\vec{G} = s_0 \xrightarrow{G_1} s_1 \xrightarrow{G_2} s_2 \ldots s_{n-1} \xrightarrow{G_n} s_n$ of simple edges in the Cayley graph, we define the *level* of the path by $\text{level}(\vec{G}) = \max\{\text{level}(s_i) \mid i = 0, \ldots, n\}$. Here, the level of a vertex is of course as defined in Definition 4.2.10.

4.4 Basic generators

To simplify the proof of completeness, it will sometimes be useful to consider a smaller set of generators for $U_n(\mathbb{D}[i])$, which we call the *basic generators*. They are the following:

- $X_{[j,j+1]}$, for $0 \leq j < n-1$;
- $K_{[0,1]}$; and
- $i_{[0]}$.

We also call the corresponding edges of the Cayley graph *basic edges*.

Lemma 4.4.1. *Each generator $G \in \mathcal{G}$ can be written as a product of basic generators by means of repeated applications of the following relations:*

$$\begin{aligned} i_{[j]} &\sim_\Delta X_{[0,j]} i_{[0]} X_{[0,j]} && \text{for } j > 1, \\ K_{[j,\ell]} &\sim_\Delta X_{[0,j]} K_{[0,\ell]} X_{[0,j]} && \text{for } j > 0, \\ K_{[0,\ell]} &\sim_\Delta X_{[1,\ell]} K_{[0,1]} X_{[1,\ell]} && \text{for } \ell > 1, \\ X_{[j,\ell]} &\sim_\Delta X_{[j,j+1]} X_{[j+1,\ell]} X_{[j,j+1]} && \text{for } \ell > j+1. \end{aligned}$$

Proof. By case distinction. For example, by repeated application of these relations, we have:

$$K_{[2,4]} \sim_\Delta X_{[0,1]}X_{[1,2]}X_{[0,1]}X_{[1,2]}X_{[2,3]}X_{[3,4]}X_{[2,3]}X_{[1,2]}K_{[0,1]}X_{[1,2]}X_{[2,3]}X_{[3,4]}X_{[2,3]}X_{[1,2]}X_{[0,1]}X_{[1,2]}X_{[0,1]}.$$

□

Moreover, we will use the following fact: if $s \xrightarrow{G} r$ is a simple edge of the Cayley graph and $s \xrightarrow{G_1} s_1 \xrightarrow{G_2} \dots \xrightarrow{G_m} r$ is the corresponding sequence of basic edges obtained from the above relations, then $\text{level}(s_1), \dots, \text{level}(s_{m-1}) \leq \max(\text{level}(s), \text{level}(r))$. In other words, the conversion to basic generators does not increase the level.

4.5 Reduction of completeness to the Main Lemma

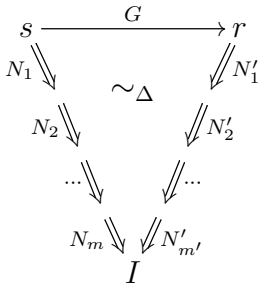
We now outline our strategy for proving completeness. For pedagogical reasons, the following lemmas are stated in the opposite order in which they are proved, i.e., each lemma implies the one before it. Completeness is a direct consequence of the following lemma.

Lemma 4.5.1. *Every word is relationally equivalent to its normal form.*

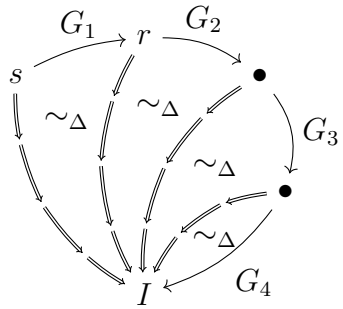
To see why the lemma implies completeness, let v, u be any two words such that $[v] = [u]$. Then v and u have the same normal form, say w. By the lemma, $v \sim_\Delta w \sim_\Delta u$, from which completeness follows by transitivity.

We will prove Lemma 4.5.1 by induction on the length of the word u. Since by Lemma 4.4.1, each generator is relationally equivalent to a word of basic generators, we can assume without loss of generality that u consists of basic generators. For the induction step, we use the following lemma:

Lemma 4.5.2. *Consider any basic edge $s \xrightarrow{G} r$ of the Cayley graph. Let $w = \text{normal}(s)$ and $u = \text{normal}(r)$. Then $w \sim_\Delta uG$. Or in pictures, the following diagram commutes relationally:*



To see why Lemma 4.5.2 implies Lemma 4.5.1, consider any word u composed of basic generators. If $u = \varepsilon$, then u is already normal so there is nothing to show. Otherwise, $u = u'G$ for some generator G. Let $s = [u]^{-1}$ and $r = [u']^{-1}$; then $s \xrightarrow{G} r$ is a basic edge of the Cayley graph. By Lemma 4.5.2, we have $\text{normal}(r)\, G \sim_\Delta \text{normal}(s)$. Also, by the induction hypothesis, since u' is a shorter word than u, we have $u' \sim_\Delta \text{NF}(u')$. Then the claim follows because $u = u'G \sim_\Delta \text{NF}(u')\, G = \text{normal}(r)\, G \sim_\Delta \text{normal}(s) = \text{NF}(u)$. The following diagram illustrates the proof in case $u = G_4G_3G_2G_1$ is a word of length 4.



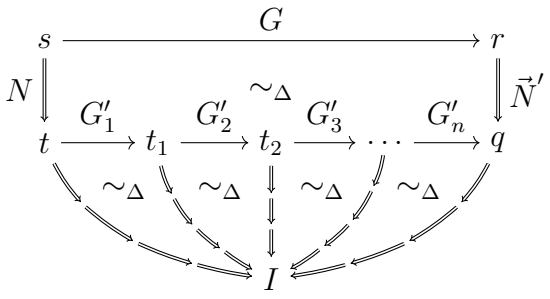
So now, what is left to do is to prove Lemma 4.5.2. We will prove this by induction on the level of s. The induction step uses the following lemma.

Lemma 4.5.3 (Main Lemma)**.** *Assume $s \xrightarrow{G} r$ is a basic edge of the Cayley graph, and $s \overset{N}{\Longrightarrow} t$ is a normal edge. Then there exists a sequence of normal edges $r \overset{\vec{N}'}{\Longrightarrow} q$ and a sequence of basic edges $t \xrightarrow{\vec{G}'} q$ such that $\vec{N}'G \sim_\Delta \vec{G}'N$ and level$(\vec{G}') <$ level(s).*

Here is the relation $\vec{N}'G \sim_\Delta \vec{G}'N$ *as a diagram:*

$$\begin{array}{ccc} s & \xrightarrow{G} & r \\ {\scriptstyle N}\Downarrow & \sim_\Delta & \Downarrow{\scriptstyle \vec{N}'} \\ t & \xrightarrow{\vec{G}'} & q. \end{array} \tag{4.1}$$

The proof that Lemma 4.5.3 implies Lemma 4.5.2 proceeds by induction on the level of s. The base case arises when $s = I$. In that case, an easy case distinction shows that the claim holds for all generators G. For the induction step, we have $s \neq I$, so there exists a unique normal edge $s \overset{N}{\Longrightarrow} t$. By Lemma 4.5.3, there exists a sequence of normal edges $r \overset{\vec{N}'}{\Longrightarrow} q$ and a sequence of basic edges $t \xrightarrow{\vec{G}'} q$ such that (4.1) holds and $\text{level}(\vec{G}') < \text{level}(s)$. Since the level of each vertex occurring in the path $t \xrightarrow{\vec{G}'} q$ is strictly less than the level of s, by the induction hypothesis, the claim of Lemma 4.5.2 is already true for each edge in this path. Then the lemma follows by the following diagram; note that each part commutes relationally and therefore so does the whole diagram.



Given the above sequence of lemmas, to finish the completeness proof, all that is now left to do is to prove Lemma 4.5.3.

4.6 Proof of the Main Lemma

Before we prove Lemma 4.5.3, we collect a number of useful consequences of the relations from Figure 4.1. These are shown in Figure 4.2. The proofs of these relations are straightforward.

To keep the proof of Lemma 4.5.3 as readable as possible, we make the following simplification. Each time we complete the diagram (4.1), we will permit $\vec{G}'$ to be a sequence of simple edges, rather than basic edges as required by the lemma. This is

justified because each such sequence of simple edges can be expanded into a (usually much longer) sequence of basic edges whose level is no higher than that of the original sequence.

With that in mind, we now proceed to prove Lemma 4.5.3 by case distinction. Assume $s \xrightarrow{G} r$ is a basic edge of the Cayley graph, and $s \overset{N}{\Longrightarrow} t$ is a normal edge. Let $L = (p, k, m)$ be the level of s; specifically, p is the index of the pivot column, k is the least denominator exponent of the pivot column, m is the number of odd entries in $\gamma^k v$, where v is the pivot column. Also let $w = \gamma^k v$.

Case 1. $G = i_{[0]}$. Let j be the index of the first odd entry of w.

Case 1.1. $j > 0$. Note that $0 < j \leq p$. Then the normal edge N does not act on row 0, and N is still the normal edge for state r. We complete the diagram as follows.

$$\begin{array}{ccc} s & \xrightarrow{i_{[0]}} & r \\ {\scriptstyle N}\Big\Vert & & \Big\Vert{\scriptstyle N} \\ t & \xrightarrow{i_{[0]}} & q \end{array}$$

The diagram commutes relationally by (C4)–(C9), which ensure that disjoint generators commute. We will encounter additional cases in which the states s and r generate the same syllable N and the indices that N acts on are disjoint from those of G. We will refer to such cases as "disjoint" cases.

Case 1.2. $j = 0$, $k = 0$, and $p = 0$. Since the jth entry of v is odd, by Lemma 4.2.2, it must be of the form i^e for some $e \in \{0, \ldots, 3\}$. Note that $e = 0$ is not possible, because then v would be e_p and would not be a pivot column. Therefore $e > 0$. In that case, the normal edge from s is $i_{[0]}^{4-e}$ and we complete the diagram as follows.

$$\begin{array}{ccc} s & \xrightarrow{i_{[0]}} & r \\ {\scriptstyle i_{[0]}^{4-e}}\Big\Vert & & \Big\Vert{\scriptstyle i_{[0]}^{3-e}} \\ t & \xrightarrow[\varepsilon]{} & t \end{array}$$

Note that here, $\xrightarrow{\varepsilon}$ denotes a path of length 0. Also, $\overset{i_{[0]}^{3-e}}{\Longrightarrow}$ denotes a path of length 0 if $e = 3$ and a path of length 1 otherwise. The diagram commutes relationally by reflexivity.

Case 1.3. $j = 0$, $k = 0$, and $p > 0$. In this case, the exact synthesis algorithm specifies that the normal edge from s is $X_{[0,p]} i_{[0]}^{-e}$, and the normal edge from r is

$$K^\dagger_{[j,\ell]} i_{[j]} \sim_\Delta i_{[j]} i_{[\ell]} X_{[j,\ell]} K^\dagger_{[j,\ell]} i_{[\ell]} \tag{4.2}$$

$$K_{[j,\ell]} \sim_\Delta i^3_{[j]} i^3_{[\ell]} K^\dagger_{[j,\ell]} \tag{4.3}$$

$$K^\dagger_{[j,\ell]} i_{[\ell]} K_{[j,\ell]} \sim_\Delta i^3_{[\ell]} X_{[j,\ell]} K^\dagger_{[j,\ell]} i_{[\ell]} \tag{4.4}$$

$$X_{[j,\ell]} i^q_{[j]} X_{[k,\ell]} \sim_\Delta X_{[j,k]} X_{[j,\ell]} i^q_{[j]} \tag{4.5}$$

$$X_{[k,\ell]} i^q_{[k]} X_{[j,k]} \sim_\Delta X_{[j,k]} X_{[j,\ell]} i^q_{[j]} \tag{4.6}$$

$$K_{[j,\ell]} i^q_{[\ell]} X_{[k,\ell]} \sim_\Delta X_{[k,\ell]} K_{[j,k]} i^q_{[k]} \tag{4.7}$$

$$K^\dagger_{[\ell,\ell']} K^\dagger_{[j,j']} K^\dagger_{[j',\ell']} K^\dagger_{[j,\ell]} X_{[\ell,j']} \sim_\Delta X_{[\ell,j']} K^\dagger_{[\ell,\ell']} K^\dagger_{[j,j']} K^\dagger_{[j',\ell']} K^\dagger_{[j,\ell]} \tag{4.8}$$

$$K^\dagger_{[j,\ell]} i_{[\ell]} X_{[j,\ell]} \sim_\Delta X_{[j,\ell]} i^3_{[j]} i_{[\ell]} K^\dagger_{[j,\ell]} i_{[\ell]} \tag{4.9}$$

Figure 4.2: Some useful relations in $\sim_\Delta$. All of these are consequences of the relations in Figure 4.1. As before, in each relation, the indices are assumed to be distinct, and when a generator $X_{[a,b]}$ or $K_{[a,b]}$ is mentioned, we assume $a < b$. $K^\dagger$ abbreviates K^7.

$X_{[0,p]} i^{-e-1}_{[0]}$. Here and from now on, we will tacitly understand all exponents of $i_{[j]}$ to be taken modulo 4, which is justified by relation (C1). Similarly, from now on we will also tacitly use relations (C2) and (C3) to invert the X and K generators when appropriate. We complete the diagram as follows.

$$\begin{array}{ccc} s & \xrightarrow{i_{[0]}} & r \\ {\scriptstyle X_{[0,p]} i^{-e}_{[0]}}\Big\Downarrow & & \Big\Downarrow{\scriptstyle X_{[0,p]} i^{-e-1}_{[0]}} \\ t & \xrightarrow[\varepsilon]{} & q \end{array}$$

Case 1.4. $j = 0$ and $k > 0$. By Lemma 4.2.4, w has an even number of odd entries. Let ℓ be the index of the second odd entry of w. In this case, the exact synthesis algorithm specifies that the normal edge from s is $K^\dagger_{[0,\ell]} i^q_{[\ell]}$ and the normal edge from r is $K^\dagger_{[0,\ell]} i^{q'}_{[\ell]}$, for $q, q' \in \{0, 1\}$ and $q' \neq q$. We complete the diagram as follows:

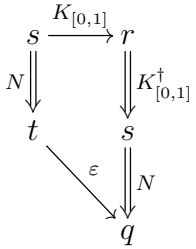
$$\begin{array}{ccc} s & \xrightarrow{i_{[0]}} & r \\ {\scriptstyle K^\dagger_{[0,\ell]} i^q_{[\ell]}}\Big\Downarrow & & \Big\Downarrow{\scriptstyle K^\dagger_{[0,\ell]} i^{q'}_{[\ell]}} \\ t & \xrightarrow{i_{[0]} i_{[\ell]} X^q_{[0,\ell]}} & q \end{array}$$

This diagram commutes relationally by (C15) when $q = 0$ and by (4.2) when $q = 1$.

Case 2. $G = K_{[0,1]}$.

Case 2.1. $k = 0$. Let j be the index of the first odd entry of w.

Case 2.1.1. $j < 2$. In this case, the exact synthesis algorithm specifies that the normal edge from r is $K^\dagger_{[0,1]}$. We complete the diagram as follows, and it commutes relationally by (C3).

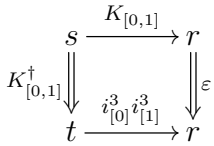


We will encounter additional cases in which the normal edge from r is relationally the inverse of G. In such cases, the diagram can always be completed in the same way. We refer to these cases as "retrograde".

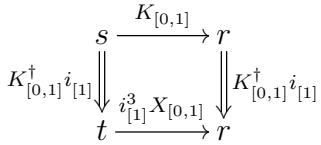
Case 2.1.2. $j \geq 2$. Note that $v_0 = v_1 = 0$. Here, and from now on, we write v_j for the jth component of a vector v. Since $j \leq p$, the normal edges from both s and r are $X_{[j,p]}i^{-e}_{[j]}$, which are disjoint from $K_{[0,1]}$. This is a disjoint case.

Case 2.2. $k > 0$. By Lemma 4.2.4, w has an even number of odd entries. Let j and ℓ be the indices of the first two odd entries of w.

Case 2.2.1. $j = 0$ and $\ell = 1$. In this case, the exact synthesis algorithm specifies that the normal edge from s is $K^\dagger_{[0,1]}i^q_{[1]}$. If $q = 0$, then level$(r) <$ level(s), and we complete the diagram as follows. It commutes relationally by (4.3).



If $q = 1$, then the exact synthesis algorithm specifies that the normal edge from r is also $K^\dagger_{[0,1]}i_{[1]}$. In this case, we complete the diagram as follows. It commutes relationally by (4.4).



Case 2.2.2. $j = 0$ and $\ell > 1$. Note that $j < \ell \leq p$, so the first two entries in each column after the pivot column are 0, hence $K_{[0,1]}$ does not change p. But it will increase the least denominator exponent of the pivot column from k to $k + 1$. The exact synthesis algorithm then specifies that the normal edge from r is $K^\dagger_{[0,1]}$, so this case is retrograde.

Case 2.2.3. $j = 1$. This is similar to the previous case. Again, $K_{[0,1]}$ increases the denominator exponent of the pivot column, the normal edge is $K^{\dagger}_{[0,1]}$, and so the case is retrograde.

Case 2.2.4. $j > 1$. In this case, $\mathrm{lde}(v_0, v_1) < k$. Note that $j < \ell \leq p$, so the first two entries in each column after the pivot column are 0, hence $K_{[0,1]}$ does not change p. The exact synthesis algorithm specifies that the normal edge from s is $K^{\dagger}_{[j,\ell]} i^{q}_{[\ell]}$. Let v' be the pivot column of r.

If $\mathrm{lde}(v'_0, v'_1) < k$, then the normal edge from r is also $K^{\dagger}_{[j,\ell]} i^{q}_{[\ell]}$ and the case is disjoint. On the other hand, if $\mathrm{lde}(v'_0, v'_1) = k$. Let $w' = \gamma^k v'$. One can show that in this case, $w'_0 \equiv w'_1 \pmod{\gamma^2}$, and therefore the normal edge from r is $K^{\dagger}_{[0,1]}$. Then this case is retrograde.

Case 3. $G = X_{[\alpha,\alpha+1]}$.

Case 3.1. $k = 0$. Let j be the index of the first odd entry of v. Note that $j \leq p$. Also, by Lemma 4.2.2, the jth entry of v is of the form i^e for some $e \in \{0, \ldots, 3\}$.

Case 3.1.1. $\alpha \geq p$. Applying $X_{[\alpha,\alpha+1]}$ increases p, and the exact synthesis algorithm specifies that the normal edge from r is $X_{[\alpha,\alpha+1]}$. Hence this case is retrograde using (C2).

Case 3.1.2. $\alpha = p - 1$.

Case 3.1.2.1. $j = \alpha + 1$. Note that $e = 0$ is not possible, because then v would be e_p and would not be a pivot column. Therefore $e > 0$. The exact synthesis algorithm specifies that the normal edge from s is $i^{-e}_{[\alpha+1]}$ and the normal edge from r is $X_{[\alpha,\alpha+1]} i^{-e}_{[\alpha]}$. We complete the diagram as follows, and it commutes relationally by (C10) and (C2).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle i^{-e}_{[\alpha+1]}}\Big\Vert & & \Big\Vert{\scriptstyle X_{[\alpha,\alpha+1]} i^{-e}_{[\alpha]}} \\ t & \xrightarrow{\varepsilon} & q \end{array}$$

Case 3.1.2.2. $j = \alpha$. The exact synthesis algorithm specifies $X_{[\alpha,\alpha+1]} i^{-e}_{[\alpha]}$ and $i^{-e}_{[\alpha+1]}$ as the normal edges from s and r, respectively. We complete the diagram as follows, and it commutes relationally by (C10).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle X_{[\alpha,\alpha+1]} i^{-e}_{[\alpha]}}\Big\Vert & & \Big\Vert{\scriptstyle i^{-e}_{[\alpha+1]}} \\ t & \xrightarrow{\varepsilon} & q \end{array}$$

Case 3.1.2.3. $j \leq \alpha - 1$. The exact synthesis algorithm specifies that the normal edge from both s and r is $X_{[j,\alpha+1]} i_{[j]}^{-e}$. We complete the diagram as follows. It commutes relationally by (4.5).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle X_{[j,\alpha+1]} i_{[j]}^{-e}} \Downarrow & & \Downarrow {\scriptstyle X_{[j,\alpha+1]} i_{[j]}^{-e}} \\ t & \xrightarrow{X_{[j,\alpha]}} & q \end{array}$$

Case 3.1.3. $\alpha \leq p - 2$.

Case 3.1.3.1. $j = \alpha$. The exact synthesis algorithm specifies that the normal edge from s is $X_{[\alpha,p]} i_{[\alpha]}^{-e}$ and the normal edge from r is $X_{[\alpha+1,p]} i_{[\alpha+1]}^{-e}$ We complete the diagram as follows. It commutes relationally by (4.6).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle X_{[\alpha,p]} i_{[\alpha]}^{-e}} \Downarrow & & \Downarrow {\scriptstyle X_{[\alpha+1,p]} i_{[\alpha+1]}^{-e}} \\ t & \xrightarrow{X_{[\alpha,\alpha+1]}} & q \end{array}$$

Case 3.1.3.2. $j = \alpha + 1$. The exact synthesis algorithm specifies that the normal edge from s is $X_{[\alpha+1,p]} i_{[\alpha+1]}^{-e}$ and the normal edge from r is $X_{[\alpha,p]} i_{[\alpha]}^{-e}$ We complete the diagram as follows. It commutes relationally by (4.6) and (C2).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle X_{[\alpha+1,p]} i_{[\alpha+1]}^{-e}} \Downarrow & & \Downarrow {\scriptstyle X_{[\alpha,p]} i_{[\alpha]}^{-e}} \\ t & \xrightarrow{X_{[\alpha,\alpha+1]}} & q \end{array}$$

Case 3.1.3.3. $j \neq \alpha$ and $j \neq \alpha + 1$. In this case, both normal edges are $X_{[j,p]} i_{[j]}^{-e}$. This case is disjoint.

Case 3.2. $k > 0$. By Lemma 4.2.4, w has an even number of odd entries. Let j and ℓ be the indices of the first two odd entries of w.

Case 3.2.1. $\alpha \geq p$. Applying $X_{[\alpha,\alpha+1]}$ increases p, and the normal edge from r is $X_{[\alpha,\alpha+1]}$. Therefore this case is retrograde.

Case 3.2.2. $\alpha < p$. We will do a case distinction on how $j < \ell$ overlaps with $\alpha < \alpha + 1$.

Case 3.2.2.1. $\ell < \alpha$. The normal edge from both s and r is $K_{[j,\ell]}^{\dagger} i_{[\ell]}^{q}$, so this case is disjoint.

Case 3.2.2.2. $\ell = \alpha$. The exact synthesis algorithm specifies that the normal edge from s is $K_{[j,\alpha]}^{\dagger} i_{[\alpha]}^{q}$, for some $q \in \{0, 1\}$.

If $w_{\alpha+1}$ is even, then the normal edge from r will be $K^{\dagger}_{[j,\alpha+1]} i^{q}_{[\alpha+1]}$. In this case, we can complete the diagram as follows. It commutes relationally by (4.7).

$$\begin{array}{ccc} s & \xrightarrow{X_{[\alpha,\alpha+1]}} & r \\ {\scriptstyle K_{[j,\alpha]} i^{q}_{[\alpha]}}\Big\Downarrow & & \Big\Downarrow{\scriptstyle K_{[j,\alpha+1]} i^{q}_{[\alpha+1]}} \\ t & \xrightarrow{X_{[\alpha,\alpha+1]}} & q \end{array}$$

Now assume that $w_{\alpha+1}$ is odd. By Lemma 4.2.4, we have a fourth odd entry. Let $j' = \alpha + 1$ and ℓ' be the index of the fourth odd entry. By Lemma 2.3.3, we have $w_j = i^e + a\gamma^3$, $w_\ell = i^f + b\gamma^3$, $w_{j'} = i^g + c\gamma^3$, and $w_{\ell'} = i^h + d\gamma^3$, for some $e, f, g, h \in \{0, \ldots, 3\}$ and $a, b, c, d \in \mathbb{Z}[i]$.

Case 3.2.2.2.1. $e = f = g = h = 0$. In this special case, the normal edges from s and r are both $K^{\dagger}_{[j,\ell]}$. We complete the diagram as follows.

$$\begin{array}{ccccccccccc} s & & & & & & & & & & r \\ & \searrow^{X_{[\ell,j']}} & & & & & & & & \swarrow & \\ {\scriptstyle K^{\dagger}_{[j,\ell]}}\Big\Downarrow & & & & & & & & & & \Big\Downarrow{\scriptstyle K^{\dagger}_{[j,\ell]}} \\ t & \xrightarrow{K^{\dagger}_{[j',\ell']}} & \xrightarrow{K^{\dagger}_{[j,j']}} & \xrightarrow{K^{\dagger}_{[\ell,\ell']}} & \xrightarrow{X_{[\ell,j']}} & \xrightarrow{K_{[\ell,\ell']}} & \xrightarrow{K_{[j,j']}} & \xrightarrow{K_{[j',\ell']}} & q \end{array}$$

The fact that this diagram commutes relationally follows from (4.8). We verify that it satisfies the level condition in Figure 4.3. The figure shows only entries j, ℓ, j', ℓ' of γ^k times the pth column of each state. Since in all 8 states below the top row, at least two entries are even, the level of all of these states is strictly less than that of s.

Case 3.2.2.2.2. Otherwise. In the general case, the normal edge from s is $K^{\dagger}_{[j,\ell]} i^{q}_{[\ell]}$, where $q = 0$ if $e - f$ is even and $q = 1$ if $e - f$ is odd. Similarly, the normal edge from r is $K^{\dagger}_{[j,\ell]} i^{q'}_{[\ell]}$, where $q' = 0$ if $e - g$ is even and $q' = 1$ if $e - g$ is odd. Define $q'' = 0$ when $-q - f + e \equiv 0 \pmod 4$ and $q'' = 1$ when $e - f - q \equiv 2 \pmod 4$. Similarly, define $q''' = 0$ when $e - g - q' \equiv 0 \pmod 4$ and $q''' = 1$ when $-q - g + e \equiv 2 \pmod 4$. We complete the diagram using the outer perimeter of Figure 4.4. Here, the bottom edge is given as in the previous case. To see that the outer perimeter relationally commutes, it suffices to show that the four inner faces relationally commutes. The bottom face does so by the previous case. The middle face commutes by repeated applications of (C5) and (C10). To see why the left face commutes, we first note that

$$K^{\dagger}_{[j,\ell]}\, i^{e-f-q}_{[\ell]} \sim_{\Delta} X^{q''}_{[j,\ell]}\, K^{\dagger}_{[j,\ell]}. \tag{4.10}$$

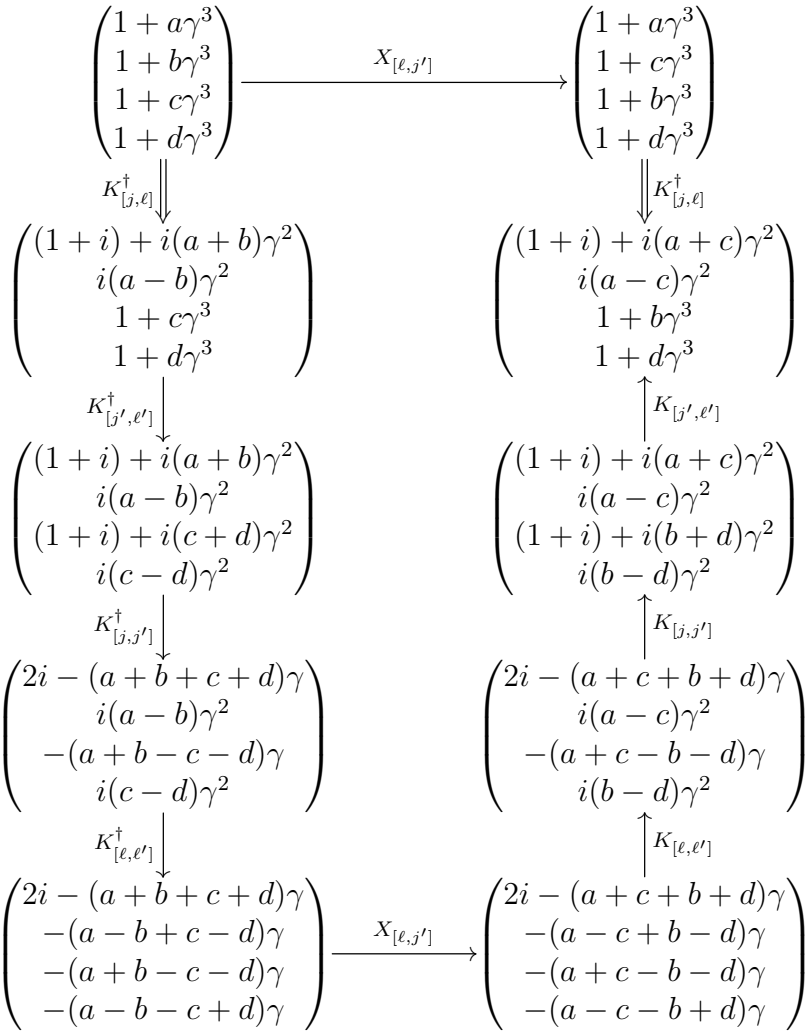


Figure 4.3: Level condition verification.

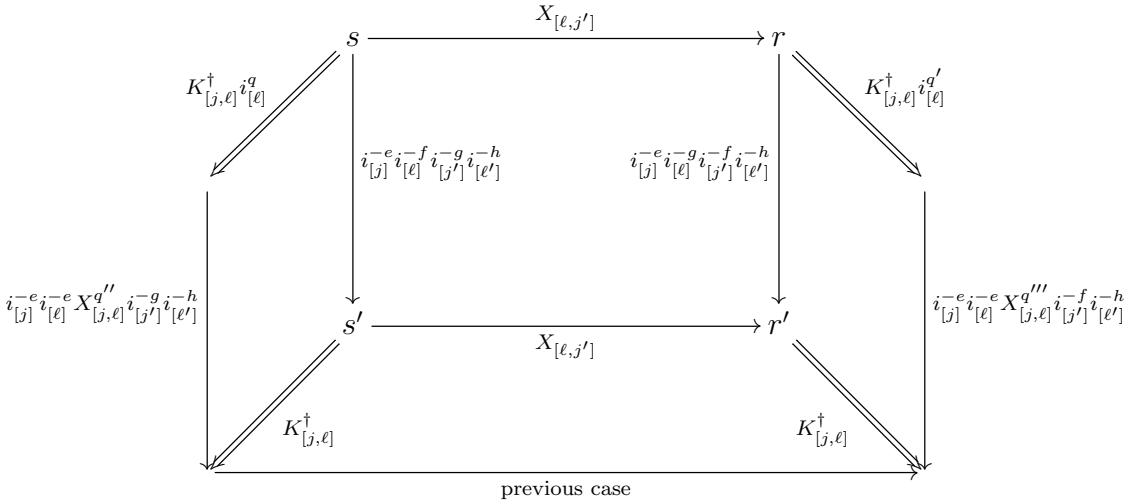


Figure 4.4: A diagram which reduces case 3.2.2.2.2 to a previous case.

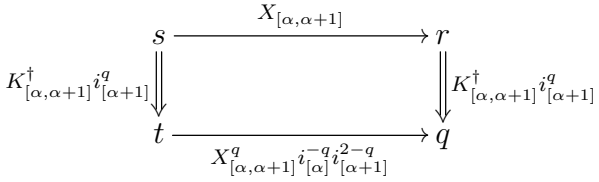
Namely, this holds by (C13) in case $e - f - q \equiv 2 \pmod 4$ and $q'' = 1$, and it holds trivially in case $e - f - q \equiv 0 \pmod 4$ and $q'' = 0$. Then the left face commutes because

$$
\begin{aligned}
K^\dagger_{[j,\ell]}\, i^{-e}_{[j]}\, i^{-f}_{[\ell]}\, i^{-g}_{[j']}\, i^{-h}_{[\ell']} &\sim_\Delta K^\dagger_{[j,\ell]}\, i^{-e}_{[j]}\, i^{-e}_{[\ell]}\, i^{e-f-q}_{[\ell]}\, i^{q}_{[\ell]} i^{-g}_{[j']}\, i^{-h}_{[\ell']} && \text{by (C1)} \\
&\sim_\Delta i^{-e}_{[j]}\, i^{-e}_{[\ell]}\, K^\dagger_{[j,\ell]}\, i^{e-f-q}_{[\ell]}\, i^{q}_{[\ell]} i^{-g}_{[j']}\, i^{-h}_{[\ell']} && \text{by (C15)} \\
&\sim_\Delta i^{-e}_{[j]}\, i^{-e}_{[\ell]}\, X^{q''}_{[j,\ell]}\, K^\dagger_{[j,\ell]}\, i^{q}_{[\ell]}\, i^{-g}_{[j']}\, i^{-h}_{[\ell']} && \text{by (4.10)} \\
&\sim_\Delta i^{-e}_{[j]}\, i^{-e}_{[\ell]}\, X^{q''}_{[j,\ell]}\, i^{-g}_{[j']}\, i^{-h}_{[\ell']}\, K^\dagger_{[j,\ell]}\, i^{q}_{[\ell]} && \text{by (C4) and (C6).}
\end{aligned}
$$

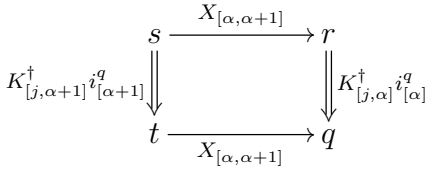
The right face commutes for the same reason, just swapping the rules of f and g. Finally, we need to verify that the diagram satisfies the level condition. To this end, note that s, r, s', and r' all have the same level, because the operations $X_{[\ell,j']}$ and $i^{-e}_{[j]}\, i^{-f}_{[\ell]}\, i^{-g}_{[j']}\, i^{-h}_{[\ell']}$ neither change the pivot column, the denominator exponent, nor the number of odd entries. Together with the fact that normal edges are level decreasing and with what was shown in the previous case, this implies the level condition.

Case 3.2.2.3. $\ell = \alpha + 1$ and $j = \alpha$. In this case, the exact synthesis algorithm prescribes that the normal edges from both s and r are $K^\dagger_{[\alpha,\alpha+1]} i^q_{[\alpha+1]}$, for $q \in \{0, 1\}$. We complete the diagram as follows. It commutes relationally by the inverse of (C13)

when $q = 0$ and by (4.9) when $q = 1$.

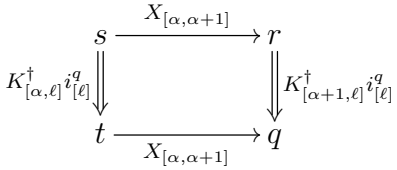


Case 3.2.2.4. $\ell = \alpha + 1$ and $j \neq \alpha$. In this case, the exact synthesis algorithm specifies that the normal edge from s is $K^\dagger_{[j,\alpha+1]} i^q_{[\alpha+1]}$ and the normal edge from r is $K^\dagger_{[j,\alpha]} i^q_{[\alpha]}$. We complete the diagram as follows. It commutes relationally by (C10) and (C12).

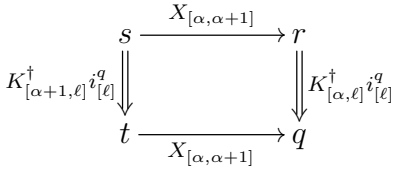


Case 3.2.2.5. $\ell > \alpha + 1$ and $j < \alpha$. This case is disjoint.

Case 3.2.2.6. $\ell > \alpha + 1$ and $j = \alpha$. In this case, the exact synthesis algorithm specifies that the normal edge from s is $K^\dagger_{[\alpha,\ell]} i^q_{[\ell]}$ and the normal edge from r is $K^\dagger_{[\alpha+1,\ell]} i^q_{[\ell]}$. We complete the diagram as follows. It commutes relationally by (C5) and (C12).



Case 3.2.2.7. $\ell > \alpha + 1$ and $j = \alpha + 1$. In this case, the exact synthesis algorithm specifies that the normal edge from s is $K^\dagger_{[\alpha+1,\ell]} i^q_{[\ell]}$ and the normal edge from r is $K^\dagger_{[\alpha,\ell]} i^q_{[\ell]}$. We complete the diagram as follows. It commutes relationally by (C5) and (C12).



Case 3.2.2.8. $\ell > \alpha + 1$ and $j > \alpha + 1$. This case is disjoint.

This finishes the proof of Lemma 4.5.3, and therefore of completeness.

Chapter 5

Presentation for 3-qubit Clifford+CS operators

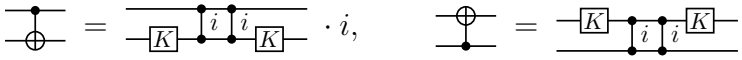
5.1 Statement of the main theorem

Theorem 5.1.1. *The 3-qubit Clifford+CS group is presented by $(\mathcal{X}, \Gamma_X)$, where the set of generators is*

$$\mathcal{X} = \{i, K_0, K_1, K_2, S_0, S_1, S_2, CS_{01}, CS_{12}\},$$

and the set of relations Γ_X is shown in Figure 5.1.

Note that in Figure 5.1, for convenience we have used the following abbreviations:

$X = KSSKi,$  $\cdot\, i,$ $\cdot\, i.$

One interesting feature of the axioms in Figure 5.1 is that the upside-down version of each relation is also a relation, except for (D15). The upside-down version of (D15) is provable, so we do not require it as an axiom.

5.2 Proof outline

Our proof follows a similar general outline as the corresponding proof for 2-qubit Clifford+T operators in Chapter 3. Let $G = U_8(\mathbb{Z}[\frac{1}{2}, i])$ be the group of unitary 8×8-matrices with entries in $\mathbb{Z}[\frac{1}{2}, i]$. In Chapter 4, we gave a presentation of G by generators and relations. It is clear that $\mathcal{CCS}(3)$ is a subgroup of G, because all of its generators belong to G. Moreover, by a result of Amy et al. [2], we know that $\mathcal{CCS}(3)$ is precisely the subgroup of G consisting of matrices whose determinant is ± 1. The only other possible values for the determinant are $\pm i$, and therefore $\mathcal{CCS}(3)$ is a subgroup of G of index 2. We can therefore apply the Reidemeister-Schreier procedure [31, 32] to find generators and relations for $\mathcal{CCS}(3)$, given the known generators and relations for G. Applying this procedure yields a complete set of relations for $\mathcal{CCS}(3)$.

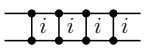
(a) Relations for $n \geq 0$:

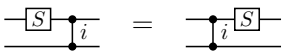
$$i^4 = \epsilon \tag{D1}$$

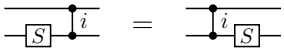
(b) Relations for $n \geq 1$:

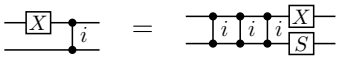
$$K^2 = i^3 \tag{D2}$$
$$S^4 = \epsilon \tag{D3}$$
$$SKSKSK = i^3 \tag{D4}$$

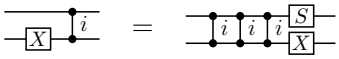
(c) Relations for $n \geq 2$:

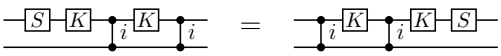

(D5)

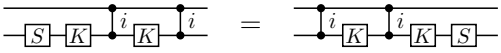

(D6)


(D7)


(D8)

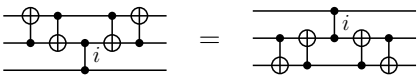

(D9)

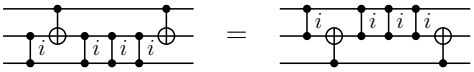

(D10)

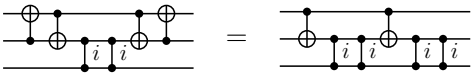

(D11)

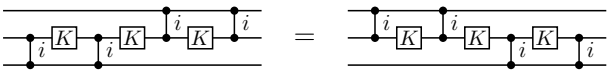
(d) Relations for $n = 3$:

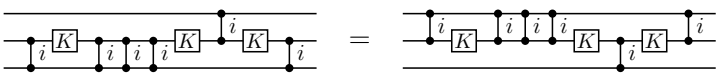

(D12)


(D13)


(D14)


(D15)


(D16)


(D17)

(e) Monoidal relations: the scalar i commutes with everything, and non-overlapping gates commute.

Figure 5.1: Complete relations for $\mathcal{CCS}(3)$. Each relation in (b) denotes three relations (one for each qubit), and each relation in (c) denotes two relations (one for each pair of adjacent qubits).

The application of the Reidemeister-Schreier method produces thousands of relations, compared to the 17 cleaned-up relations in Figure 5.1. Moreover, these relations are very large. In our code, which actually uses a sequence of multiple applications of the Reidemeister-Schreier theorem passing through a number of intermediate representations, some of the longest relations involve more than 50,000 generators. Our main contribution is the simplification of these relations. Due to the sheer magnitude of this task, we must rely on a computer to expedite the computation. However, as we did in Chapter 3, we also require the simplification process to be trustworthy, as it is very easy in a computer program to accidentally use a relation that has not yet been proved. To this end, we have formalized Theorem 5.1.1 and its proof in the proof assistant Agda. This allows the proof to be verified independently and with a high degree of confidence in its correctness, despite the magnitude of the proof.

The main idea of the simplification is to use the 17 relations from Figure 5.1, along with some of their easy consequences, to rewrite the thousands of relations until they are all eliminated. We define several rewrite systems for this task. Some of these rewrite systems are confluent and terminating, and others are just heuristics. All of these rewrite systems are implemented in Agda and the computations are verified within Agda.

5.3 Normal forms and an almost-normal form

5.3.1 Notations

For convenience, we will use the notations in Figure 5.2. The definitions for a Toffoli gate with target on the second, respectively first, qubit are given by $CCX_1 = \mathit{Swap}_{01}\, CCX_0\, \mathit{Swap}_{01}$ and $CCX_2 = \mathit{Swap}_{12}\, CCX_1\, \mathit{Swap}_{12}$. The last notation uses a twice-controlled K' gate. Here $K' = KS^\dagger$ is a variant of the K-gate that has determinant 1. The reason we are not using a twice-controlled K-gate is that it has determinant i and is therefore not an element of $\mathcal{CCS}(3)$.

5.3.2 Normal forms for finite subgroups of Clifford+CS operators

We will define normal forms and discuss the structure of the finite subgroups of Clifford+CS operators that are shown in Figure 5.3. The inclusion relations between

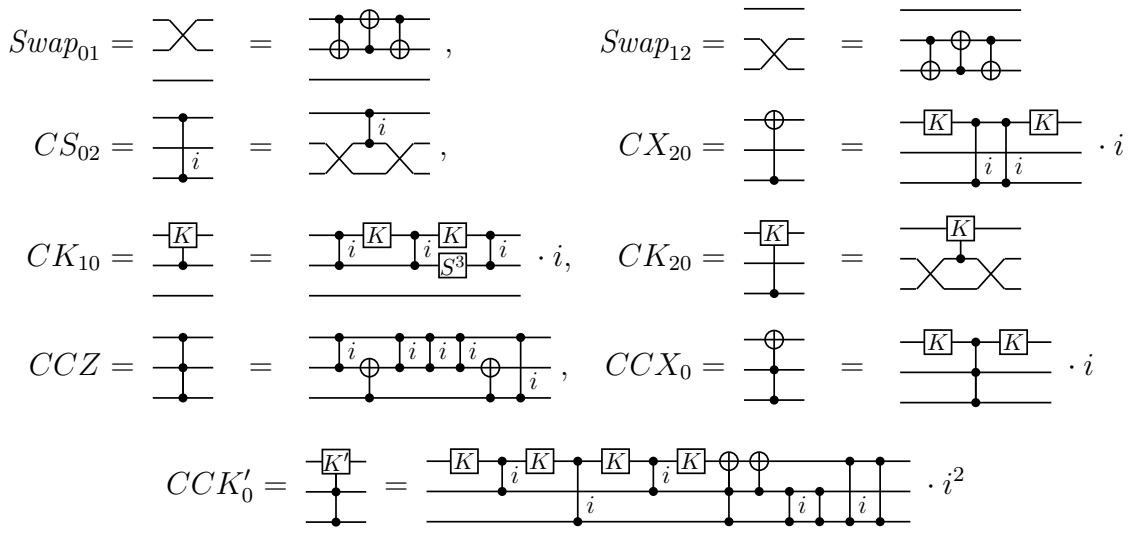


Figure 5.2: Notations for definable gates.

these subgroups are visualized in Figure 5.4.

Note that P is the group of all permutations of the computational basis vectors; we call its members *permutation operators*. Q, C, and CQ are subgroups of P. Similarly, D is the group of all diagonal operators in $\mathcal{CCS}(3)$. The remaining subgroups play a technical role in our proofs.

Given that all claims about finite groups can be proved by just enumerating the elements, we will not give proofs of the following claims about finite subgroups of $\mathcal{CCS}(3)$. Instead, we will illustrate the proofs with examples. Some of the proofs can be found in the Agda code.

The group W is the group of permutations of 3 qubits. The generators of Q all commute with each other and are self-inverse. Therefore, each element of Q can be uniquely written of the form $X_0^a\, CX_{10}^b\, CX_{20}^c\, CCX_0^d$, where $a, b, c, d \in \{0, 1\}$. We say that the subgroup Q has the following normal form:

$$\overline{Q} \quad ::= \quad X_0^a\, CX_{10}^b\, CX_{20}^c\, CCX_0^d, \text{ where } a, b, c, d \in \{0, 1\}. \tag{5.1}$$

We use $\overline{Q}$ to range over normal forms for Q. More generally, given any group G for which normal forms are defined, we use $\overline{G}$ to range over the normal forms of G. The group Q has $2^4 = 16$ distinct normal forms corresponding to 16 distinct elements. It is easy to see that $Swap_{12} \in C$, and therefore also $X_2 \in C$. The group C has the

- W, the subgroup of permutation matrices generated by $\mathcal{X}_W = \{Swap_{01}, Swap_{12}\}$.
- Q, the subgroup of permutation matrices generated by

$$\mathcal{X}_Q = \{X_0, CX_{10}, CX_{20}, CCX_0\}.$$

- C, the subgroup of permutation matrices generated by $\mathcal{X}_C = \{X_1, CX_{12}, CX_{21}\}$.
- CQ, the subgroup generated by $\mathcal{X}_C$ and $\mathcal{X}_Q$.
- P, the subgroup of permutation matrices generated by

$$\mathcal{X}_P = \{CX_{01}, CX_{10}, CX_{12}, CX_{21}, CCX_0, X_0\}.$$

- D, the diagonal subgroup generated by

$$\mathcal{X}_D = \{i, S_0, S_1, S_2, CS_{01}, CS_{12}, CS_{02}, CCZ\}.$$

- PD, the subgroup generated by $\mathcal{X}_P$ and $\mathcal{X}_D$.
- QD, the subgroup generated by $\mathcal{X}_Q$ and $\mathcal{X}_D$.
- CQD, the subgroup generated by $\mathcal{X}_C$, $\mathcal{X}_Q$ and $\mathcal{X}_D$.
- K_0D the subgroup generated by $\{K_0\} \cup \mathcal{X}_D$. Note that this group contains Q, so it can also be denoted by K_0QD.
- K_0CD, the subgroup generated by $\{K_0\} \cup \mathcal{X}_C \cup \mathcal{X}_D$. Since this group contains Q, it can also be denoted by K_0CQD.
- K_0W, the subgroup generated by K_0 and $\mathcal{X}_W$.

Figure 5.3: Finite subgroups of Clifford+CS operators.

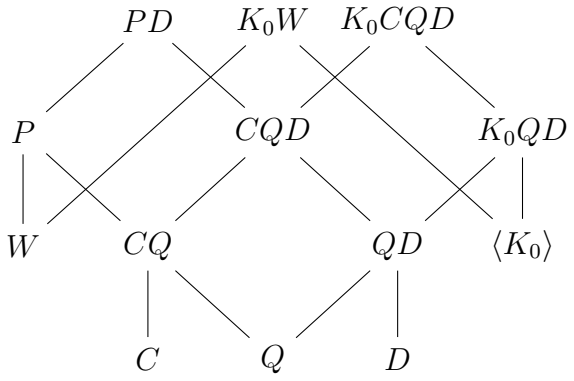


Figure 5.4: The inclusion graph of various finite subgroups of $\mathcal{CCS}(3)$.

following normal form:

$$\overline{C} \quad ::= \quad c_4 c_3 c_2 \tag{5.2}$$

where

$$\begin{aligned} c_2 &\in \{\epsilon, CX_{12}\}, \\ c_3 &\in \{\epsilon, CX_{21}, CX_{12}CX_{21}\}, \\ c_4 &\in \{X_1^a X_2^b \mid a, b \in \{0,1\}\}. \end{aligned}$$

There are $4! = 24$ distinct normal forms in C.

The group CQ is a semidirect product of C and Q with Q being normal. A semidirect product structure means that we have commuting relations of the form $qc = cq'$, or more precisely, for all $q \in Q$ and $c \in C$, there exists a unique $q' \in Q$ such that $qc = cq'$. Consequently, CQ has the following normal form:

$$\overline{CQ} \quad ::= \quad \overline{C}\ \overline{Q}.$$

The group P contains CQ as a subgroup with 105 cosets. We get the following normal form for P:

$$\overline{P} = c\overline{CQ}, \tag{5.3}$$

where c ranges over the set V of 105 left coset representatives. One can easily spot a normal form for D, since all the generators commute with each other, CCZ has order 2, and all of the other generators have order 4. The normal form is:

$$\overline{D} \quad ::= \quad i^{n_0} S_0^{n_1} S_1^{n_2} S_2^{n_3} C S_{01}^{n_4} C S_{12}^{n_5} C S_{02}^{n_6} CCZ^{n_7}, \tag{5.4}$$

where $n_0, \ldots, n_6 \in \{0,1,2,3\}$ and $n_7 \in \{0,1\}$. The group PD is a semidirect product of P and D, with D being normal. It therefore has the following normal form:

$$\overline{PD} \quad ::= \quad \overline{P}\ \overline{D}. \tag{5.5}$$

Since Q is a subgroup of P, it follows that QD is also a semidirect product. It enjoys a similar normal form as (5.5), with P replaced by Q.

It is easy to see that the group $K_0 D$ contains $\mathcal{X}_Q$, hence Q is a subgroup of $K_0 D$. We have the following normal form:

$$\overline{K_0 D} \quad ::= \quad e_4 e_3 e_2 e_1 \overline{D}\ \overline{Q}, \tag{5.6}$$

where

$$\begin{aligned} e_1 &\in \{\epsilon,\, CCK_0',\, CCK_0'CCK_0'\}, \\ e_2 &\in \{\epsilon,\, CK_{10},\, S_0CK_{10}\}, \\ e_3 &\in \{\epsilon,\, CK_{20},\, S_0CK_{20}\}, \\ e_4 &\in \{\epsilon,\, K_0,\, S_0K_0\}. \end{aligned}$$

Note that CK_{10}, CK_{20} and K_0 commute with each other but not with CCK_0'.

Notice that each element of $\mathcal{X}_C$ commutes with K_0. For any element of K_0CD, for example $w = X_1K_0CS_{01}K_0CCZ$, we can commute X_1 all the way to the right using the commuting relations and the semidirect product structure of QD. For example, we get $w = K_0CS_{01}CS_{01}CS_{01}S_0K_0CCZCS_{02}CS_{02}X_1$. We will use the following normal form for K_0CD:

$$\overline{K_0CD} = \overline{(K_0D)}\ \overline{C} = e_4e_3e_2e_1\overline{D}\ \overline{Q}\ \overline{C}. \tag{5.7}$$

Note that this also proves that K_0CD is finite, which perhaps wasn't obvious from its definition.

5.3.3 An almost-normal form for $\mathcal{CCS}(3)$

Consider a Clifford+CS circuit. We can replace the generators K_1 and K_2 by $Swap_{01}\, K_0\, Swap_{01}$ and $Swap_{12}\, Swap_{01}\, K_0\, Swap_{01}\, Swap_{12}$, respectively. The circuit becomes an alternating sequence of elements of PD and K_0:

$$PD\, K_0\, PD\, K_0 \ldots PD\, K_0\, PD.$$

By repeatedly converting subcircuits to normal forms of the form (5.5), (5.3), and (5.7), we can rewrite this circuit as follows:

$$\begin{aligned} & PD\, K_0\, PD\, K_0 \ldots PD\, K_0\, PD \\ \overset{(5.5)(5.3)}{\to}\ & c\overline{CQD}K_0c\overline{CQD}K_0 \ldots c\overline{CQD}K_0c\overline{CQD} \\ \overset{(5.7)}{\to}\ & ce_4e_3e_2e_1\overline{D}\ \overline{Q}\ \overline{C}c\overline{CQD}K_0 \ldots c\overline{CQD}K_0c\overline{CQD} \\ \overset{(5.5)(5.3)}{\to}\ & ce_4e_3e_2e_1c\overline{CQD}K_0 \ldots c\overline{CQD}K_0c\overline{CQD} \\ \overset{repeat}{\to}\ & ce_4e_3e_2e_1\, ce_4e_3e_2e_1\, \ldots\, ce_4e_3e_2e_1c\overline{CQD}. \end{aligned}$$

We can further rewrite the last expression, for example using relations in Figure 5.5. After this step, we might get some new gates that are not in V or of the form e_i. In this case, we continue with the first arrow step. We repeat the whole process until there is no further simplification. We call the resulting word an *almost-normal form*.

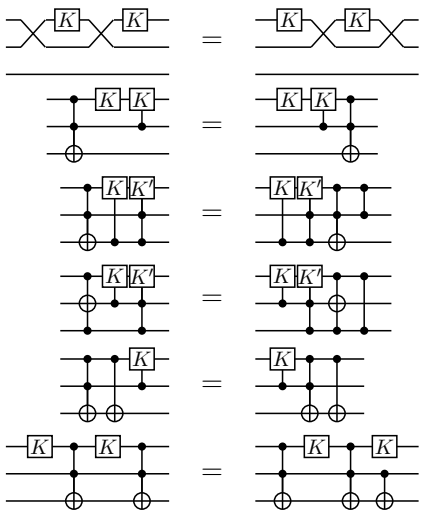


Figure 5.5: Some relations used to rewrite words of the form $ce_4e_3e_2e_1 \ldots ce_4e_3e_2e_1$.

It turns out this almost-normal form is "canonical" enough. It can be used to show that a complete set of thousands of relations hold by rewriting both sides of each relation to almost-normal form. Moreover, all rewriting rules used to get an almost-normal form are consequences of the relations in Figure 5.1. This shows that the relations in Figure 5.1 are complete.

5.3.4 Comparison with Pauli rotation decomposition

Unlike our previous work on generators and relations for 2-qubit Clifford+T operators in Chapter 3, which used a *Pauli rotation decomposition* to guide the rewriting, we found that the analog of the Pauli rotation decomposition, i.e., taking syllables that are conjugates of the CS gate under the action of the Clifford operators, does not work very well. Instead, we were surprised to find that a more useful decomposition was to take conjugates of K_0 (basically a Hadamard gate) under the action of diagonal and permutation operators. We may call this the *Hadamard decomposition* of Clifford+CS.

The fact that the Hadamard decomposition turned out to be more useful than the analog of the Pauli rotation decomposition raises the question whether our earlier work on Clifford+T could benefit from the same insight. By applying these lessons, perhaps one can come up with a simpler complete set of relations. For example, our

Clifford+T axiomatization involved a number of obvious relations and three "non-obvious" ones. We were never able to resolve the question of whether these non-obvious relations actually follow from something simpler.

5.4 Formal proof

The machine-checkable proof of Theorem 5.1.1 can be found in [11]. It has been formalized in the proof assistant Agda [1]. The proof assumes only the result of Chapter 4, i.e., the soundness and completeness of the relations from Figure 4.1 for $U_n(\mathbb{Z}[\frac{1}{2}, i])$. Everything else is proved from first principles, including, for example, a complete proof of the version of the Reidemeister-Schreier theorem that we used.

Verifying the proof. Readers who are interested in verifying the proof only need to know the following: The *statement* of Theorem 5.1.1 is contained in the file `Theorem.agda`, and the final step of the *proof* of Theorem 5.1.1 is contained in the file `Proof.agda`. As in Chapter 3, the reason we separated the statement of the theorem from its proof is to ensure that the statement assumes as little as possible: in fact, the file `Theorem.agda` is almost completely self-contained and only depends on a few definitions concerning generators, words, indices, and two-level relations. On the other hand, the proof requires a large number of auxiliary files with definitions, lemmas, tactics, and more. We checked the proof with Agda 2.6.4, and it took about 120 minutes on our laptop.

Reading the proof. For readers who are interested in inspecting our proof, here are some pointers. The folder `Lib` contains some general-purpose definitions, such as booleans and natural numbers, and some definitions and tactics related to monoids and relations. The main parts of the proof are contained in the folders `Step1` – `Step8`. Each of these steps transforms a set of generators and relations into an equivalent set of generators and relations, gradually simplifying the relations. The file `Gate.agda` provides the definitions for all gates used. The file `CosetNF.agda` contains definitions related to semidirect products and normal forms. The final proof witness is contained in the file `Proof.agda`.

5.5 $\mathcal{CCS}(3)$ is an amalgamated product of three finite groups

Using Theorem 5.1.1, we can show that $\mathcal{CCS}(3)$ is an amalgamated product of three finite groups. We choose the sets of generators as follows:

$$\begin{aligned} X &= \{K_0, i\}, \\ Y &= \{X_0, X_1, X_2, CX_{12}, CX_{21}, CX_{10}, CX_{20}, CCX_0, S_0, S_1, S_2, CS_{01}, CS_{12}, CS_{02}, CCZ, i\}, \\ Z &= \{Swap_{01}, Swap_{12}\}. \end{aligned}$$

One can check that $\langle X \cup Y \rangle = K_0 CQD$, $\langle X \cup Z \rangle = K_0 W$, and $\langle Y \cup Z \rangle = PD$. Since $X \cup Y$, $X \cup Z$, and $Y \cup Z$ each generate a finite subgroup of $\mathcal{CCS}(3)$, all that is left to show is that each relation of $\mathcal{CCS}(3)$ is a consequence of relations in one of these three subgroups.

Before we prove this, we must adjust the relations of Figure 5.1 to fit the new set of generators $X \cup Y \cup Z$. This requires two adjustments. First, compared to the set of generators from Theorem 5.1.1, a number of new generators have been added, namely X_0, X_1, X_2, CX_{12}, CX_{21}, CX_{10}, CX_{20}, CCX_0, CS_{02}, CCZ, $Swap_{01}$, and $Swap_{12}$. For each of these, we must add a defining relation in terms of the old generators. These relations are as in Section 5.3.1. Second, the two generators K_1 and K_2 are no longer used, so where they appear in the relations, they must now be regarded as abbreviations for the words $Swap_{01}\, K_0\, Swap_{01}$ and $Swap_{12}\, Swap_{01}\, K_0\, Swap_{01}\, Swap_{12}$, respectively. With these adjustments, we still have a sound and complete presentation of $\mathcal{CCS}(3)$ using the generators $X \cup Y \cup Z$.

Now we must show that each of the relations follows from relations that hold in $\langle X \cup Y \rangle$, $\langle X \cup Z \rangle$, or $\langle Y \cup Z \rangle$. Many of the relations, such as (D1), (D3), (D5)–(D9), and (D12) are already in one of the three subgroups, so there is nothing else to show for them. The remaining relations must be proved individually; here, we give a proof of (D16) as a representative example in Figure 5.6. In Figure 5.6, steps (1) and (5) use the definition of K_1, which is at this point merely an abbreviation for $Swap_{01}\, K_0\, Swap_{01}$. Steps (2) and (4) uses the relations $Swap_{01}^2 = \epsilon$ and $Swap_{01}\, CS_{12}\, Swap_{01} = CS_{02}$ and $Swap_{01}\, CS_{01}\, Swap_{01} = CS_{01}$. All three of these relations come from $\langle Y \cup Z \rangle$. Step (3) uses the relation $CS_{02}\, K_0\, CS_{02}\, K_0\, CS_{01}\, K_0\, CS_{01} = CS_{01}\, K_0\, CS_{01}\, K_0\, CS_{02}\, K_0\, CS_{02}$, which comes from $\langle X \cup Y \rangle$.

In addition to (D16), there are a number of other relations to be proved, but they all follow a similar pattern.

$$
\begin{aligned}
& CS_{12}\, K_1\, CS_{12}\, K_1\, CS_{01}\, K_1\, CS_{01} \\
&= CS_{12}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{12}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{01}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{01} && (1)\\
&= Swap_{01}\, CS_{02}\, K_0\, CS_{02}\, K_0\, CS_{01}\, K_0\, CS_{01}\, Swap_{01} && (2)\\
&= Swap_{01}\, CS_{01}\, K_0\, CS_{01}\, K_0\, CS_{02}\, K_0\, CS_{02}\, Swap_{01} && (3)\\
&= CS_{01}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{01}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{12}\, Swap_{01}\, K_0\, Swap_{01}\, CS_{12} && (4)\\
&= CS_{01}\, K_1\, CS_{01}\, K_1\, CS_{12}\, K_1\, CS_{12} && (5)
\end{aligned}
$$

Figure 5.6: An example: a relation is derivable using relations in three finite submonoids.

Chapter 6

Conclusion and future work

The first contribution of this thesis is a presentation of the 2-qubit Clifford+T group by generators and relations. We did this by applying the Reidemeister-Schreier theorem to Greylyn's presentation of the group of unitary 4×4-matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. Since there is a very large number of relations to check and simplify, and checking them by hand or by an unverified computer program would be error-prone, we used the proof assistant Agda to formalize our proof. This result was first announced in [7], and was published in [10].

The second contribution is a presentation by generators and relations of the group of unitary $n \times n$-matrices with entries in the ring $\mathbb{D}[i] = \mathbb{Z}[\frac{1}{2}, i]$. This matrix group has some applications in quantum computing because it arises as the group of unitary operations that are exactly representable by a certain gate set that is a subset of the Clifford+T circuits; namely, the Clifford+CS gates. We have described this group in terms of generators that are 1- and 2-level matrices. This work was published in [8].

The third contribution is a presentation of the group of 3-qubit Clifford+CS operators by just 17 relatively simple relations. We prove this by a combination of a the result from Chapter 4, the Reidemeister-Schreier method, and an Agda program that simplified several thousand large relations into the aforementioned 17 simple ones. Doing this simplification by brute force would not have been feasible. Instead, we proceeded by identifying a number of finite subgroups of the Clifford+CS operators, defining normal forms for these, and then combining them into carefully chosen rewrite rules. These rules eventually reduced the relations to a manageable size. In the process, we learned many interesting facts about finite subgroups of Clifford+CS. One of these facts is that the 3-qubits Clifford+CS group is an amalgamated product of three of its finite subgroups. Concretely, this means that every relation that holds in this group follows from relations that already hold in some finite subgroup of Clifford+CS.

One candidate for future work would be to find a complete set of relations for the Clifford+T group with 3 or more qubits. This is currently out of reach for two reasons: first, the computations required to simplify any potential set of relations will be even more labor-intensive than in the 2-qubit case. Second, and more seriously, there is no known presentation of the group of unitary $n \times n$-matrices over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ for $n > 4$.

Complete relations for 4-qubit Clifford+CS operators might be a more feasible project for future work, since we have given a finite presentation of $U_n(\mathbb{Z}[\frac{1}{2}, i])$ for all n, and the group of 4-qubit Clifford+CS operators is an index 4 subgroup. Also, many of our results about finite subgroups of Clifford+CS are valid for n qubits. The only problem remaining is that compared to the 3-qubit case, we probably need at least one thousand times more computation power for applying the Reidemeister-Schreier method to a set of 2-level relations for 16×16-matrices and then simplifying a massive set of relations. Right now, our Agda program runs for about 2 hours, and we don't think Agda can handle such a large computation.

Another intriguing question is whether one can find a unique normal form for 3-qubit Clifford+CS circuits, like the Matsumoto-Amano normal form for 1-qubit Clifford+T circuits. We currently only have an "almost-normal" form, but the fact that it efficiently reduced all of our relations is encouraging. If we succeed, it also might help solve the 4-qubit Clifford+CS presentation problem.

There is also some possible future work about two-level presentations. Other subgroups of the Clifford+T group have been described by generators and relations [28], but this has not yet been done for the Clifford+T group itself, except in the case of matrices of size 4×4 [21]. Doing so would require extending our results from the ring $\mathbb{D}[i]$ to the ring $\mathbb{D}[\omega]$. This problem turns out to be harder than one would expect, because as the complexity of the ring increases, it becomes more and more difficult to complete all the cases of the Lemma 4.5.3.

Bibliography

[1] Agda documentation. `https://agda.readthedocs.io/`. Accessed: 2023-06-13.

[2] Matthew Amy, Andrew N. Glaudell, and Neil J. Ross. Number-theoretic characterizations of some restricted Clifford+T circuits. *Quantum*, 4:252, Apr 2020. Also available from arXiv:1908.06076.

[3] Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, 2014. Also available from arXiv:1303.2042.

[4] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. Available from arXiv:1206.0758v2, August 2012.

[5] Matthew Amy and Michele Mosca. T-count optimization and Reed-Muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, 2019. Also available from arXiv:1601.07363.

[6] Michael Beverland, Earl Campbell, Mark Howard, and Vadym Kliuchnikov. Lower bounds on the non-Clifford resources for quantum computations. *Quantum Science and Technology*, 5(3):035009, 2020.

[7] Xiaoning Bian and Peter Selinger. Relations for the 2-qubit Clifford+T operator group. Slides presented at the Workshop on Quantum Programming and Circuits, Waterloo, Canada, June 8–11, 2015. Available from `https://mathstat.dal.ca/~xbian/talks/slide_cliffordt2.pdf`, 2015.

[8] Xiaoning Bian and Peter Selinger. Generators and relations for $U_n(\mathbb{Z}[1/2, i])$. In *Proceedings of the 18th International Conference on Quantum Physics and Logic, QPL 2021, Gdansk, Poland*, volume 343 of *Electronic Proceedings in Theoretical Computer Science*, pages 145–164, 2021. Also available from arXiv:2105.14047.

[9] Xiaoning Bian and Peter Selinger. Agda code for 2-qubit Clifford+T complete relations. Available as ancillary material at arXiv:2204.02217, 2022.

[10] Xiaoning Bian and Peter Selinger. Generators and relations for 2-qubit Clifford+T operators. To appear in *QPL 2022*. Available from arXiv:2204.02217, April 2022.

[11] Xiaoning Bian and Peter Selinger. Agda code for 3-qubit Clifford+CS complete relations. Available from `https://www.mathstat.dal.ca/~selinger/papers/downloads/cliffordcs3/`, 2023.

[12] Harry Buhrman, Richard Cleve, Monique Laurent, Noah Linden, Alexander Schrijver, and Falk Unger. New limits on fault-tolerant quantum computation. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 411–419, 2006. Also available from arXiv:quant-ph/0604141.

[13] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and effective techniques for T-count reduction via spider nest identities. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:23, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. Also available from arXiv:2004.05164.

[14] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to reduce $\pi/4$-parity-phase circuits, motivated by the ZX calculus. *Electronic Proceedings in Theoretical Computer Science*, 318:131–149, May 2020. Also available from arXiv:1911.09039.

[15] Shelly Garion and Andrew W Cross. Synthesis of CNOT-dihedral circuits with optimal number of two qubit gates. *Quantum*, 4:369, 2020.

[16] Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+T circuits. *Physical Review A*, 87(3):032332 (7 pages), 2013. Also available from arXiv:1212.0506.

[17] Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+T circuits. *Physical Review A*, 87:032332 (7 pages), 2013. Also available from arXiv:1212.0506.

[18] Andrew N. Glaudell, Neil J. Ross, and Jacob M. Taylor. Optimal two-qubit circuits for universal fault-tolerant quantum computation. *npj Quantum Information*, 7(1):103, Jun 2021.

[19] Georges Gonthier. Formal proof — the four color theorem. *Notices of the American Mathematical Society*, 55(11):1382–1393, 2008.

[20] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the T-count. *Quantum Information and Computation*, 14(15–16):1261–1276, 2014. Also available from arXiv:1308.4134.

[21] Seth E. M. Greylyn. *Generators and relations for the group* $U_4(\mathbb{Z}[\frac{1}{\sqrt{2}}, i])$. M.Sc. thesis, Dalhousie University, 2014. Available from arXiv:1408.6204.

[22] Jeongwan Haah and Matthew B Hastings. Codes and protocols for distilling T, controlled-S, and Toffoli gates. *Quantum*, 2:71, 2018.

[23] Thomas C. Hales. Formal proof. *Notices of the American Mathematical Society*, 55(11):1370–1380, 2008.

[24] Luke E. Heyfron and Earl T. Campbell. An efficient quantum compiler that reduces T count. *Quantum Science and Technology*, 4(1):015004, 2018. Also available from arXiv:1712.01557.

[25] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics 84. Springer, 1982.

[26] Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Phys. Rev. A*, 102:022406, Aug 2020. Preprint available from arXiv:1903.10477.

[27] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates. *Quantum Information and Computation*, 13:607–630, June 2013. Available from arXiv:1206.5236.

[28] Sarah Meng Li, Neil J. Ross, and Peter Selinger. Generators and relations for the group on(z[1/2]). *Electronic Proceedings in Theoretical Computer Science*, 343:210–264, September 2021.

[29] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. Automated optimization of large quantum circuits with continuous parameters. *NPJ Quantum Information*, 4(1), May 2018. Also available from arXiv:1710.07345.

[30] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2002.

[31] Kurt Reidemeister. Knoten und Gruppen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 5(1):7–23, 1927.

[32] Otto Schreier. Die Untergruppen der freien Gruppen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 5(1):161–183, 1927.

[33] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[34] Peter Selinger. Generators and relations for n-qubit Clifford operators. *Logical Methods in Computer Science*, 11(2:10):1–17, 2015. Also available from arXiv:1310.6813.

[35] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.