Computation in Models Inspired by Near-Term Quantum Devices

by

Luke Schaeffer

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Author ...... Signature redacted ....
Department of Electrical Engineering and Computer Science
August 30, 2019

Certified by.... Signature redacted ....
Scott Aaronson
Professor of Computer Science, University of Texas at Austin
Thesis Supervisor

Accepted by .... Signature redacted ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Computation in Models Inspired by Near-Term Quantum Devices

by

Luke Schaeffer

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The race is on to build the first quantum computer, and although there are many groups working towards this goal, their quantum devices have certain architectural properties in common. First, the devices tend to be built on qubits arranged in a 2D grid, with gates between neighboring qubits. Second, we expect Clifford gates will be an important gate set because of their close connection to stabilizer codes (being both necessary to encode qubits, and easily implemented on encoded logical qubits). Finally, the limited lifespan of qubits (due to various forms of noise) encourages *shallow* circuits, at least until fault tolerance is achieved. It is important to acknowledge these limitations and incorporate them into our models of computation in order to make the most out of near-term quantum devices.

In this thesis, we will explore the three concepts above. First, we see a cellular automaton with a demanding universality property, to illustrate that computation in the grid is possible even under extreme circumstances. Second, we present a classification of subsets of the Clifford gates, furthering our understanding of this important quantum gate set. Finally, recent work of Bravyi, Gosset, and König (2018) shows, unconditionally, that there are problems that can be solved by constant-depth quantum circuits, but not constant-depth classical circuits. We present two follow-up results above low-depth quantum circuits with the goal of strengthening the classical hardness. One result extends the separation AC^0 circuits (constant depth, unbounded fan-in AND/OR gates), and arguably simplifies the Bravyi et al. problem. The other result proves hardness beyond AC^0 (specifically to $\oplus\mathsf{L}$) for the task of interactively simulating certain constant-depth quantum circuits.

Thesis Supervisor: Scott Aaronson
Title: Professor of Computer Science, University of Texas at Austin

Acknowledgments

I would like to thank my advisor, Scott Aaronson, for guiding the direction of my research and kicking me forward whenever my project stalled. I would like to thank my collaborators, especially those who contributed to work in this thesis: Adam Bene Watts, Daniel Grier, Robin Kothari, and Avishay Tal. Finally, to the great friends I have made at MIT, thank you for the past six years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, we have seen an unprecedented new wave of investment in quantum hardware. Established tech giants, ambitious start-ups, and university labs around the world have finally begun the race for quantum computers in earnest. Over the last six years[1], for example, Google has moved from cautiously benchmarking of D-Wave devices, to constructing their own 72-qubit [68] quantum architecture. These devices (and others like them) appear to be the precursors of general-purpose quantum computers, if they can overcome the high error rates and limited number of qubits they have at present. We assume that with time, researchers will find a way to lower the error rate below the critical threshold for quantum computer and then, as with conventional computers decades ago, scale up the size of the devices (i.e., number of qubits) to the point where full-scale quantum computation is possible.

An important milestone for these quantum hardware groups is the first demonstration of *quantum advantage* or *quantum supremacy*, i.e., some computational task for which quantum devices outperform classical devices. A quantum advantage is crucial to convince skeptics (and more importantly, investors) that quantum devices are not snake oil, and that some fundamentally different, new form of computation is evident. Any particular demonstration will have detractors, so we expect there will be a sequence of increasingly convincing quantum advantage experiments until only the most die-hard detractors of quantum computing are left unpersuaded. Indeed, we may have already seen the first example: although the verdict seems to be that D-Wave's devices did not live up to their marketing (or indeed, surpass the classical state of the art), there was enough doubt about how their quantum annealing algorithms compared to the best classical algorithms that it spurred further investigation by Google [40], and the computer science community in general [79, 104].

The push for a quantum advantage is not just the purview of hardware engineers; there has been significant progress on the theoretical side too. From the beginning, interest in quantum computing has been motivated by the idea that quantum computers could be more powerful than classical computers. That is, the class of problems solved by quantum computers in polynomial time, BQP, is separate from the class of problems that can be solved by classical computers with randomness in polynomial

[1] Coincidentally, the same period of time the research within this thesis was conducted.

time, BPP.[2] Proving such a separation unconditionally is beyond current techniques in complexity theory,[3] but there are some convincing conditional results on the theoretical side.

Initially, it was assumed there would be a quantum advantage for the task of simulating quantum systems [48].[4] That is, there are lots of special cases where physical quantum systems can be simulated efficiently, even classically, but some cases appear to require exponential time. A universal quantum computer could plausibly simulate the same systems in polynomial time (for a substantial quantum advantage). However, this is bordering on tautological, like saying that simulating an NP machine is an NP-complete problem. The first concrete evidence of a quantum advantage came from Deutsch-Jozsa [41], separating EQP and P in a black-box query model of computation, and Simon's algorithm [107] improved this to a separation of BQP and BPP (again, only in the *query* model). Shor's algorithm [105] solves another problem, factoring, which is thought to be hard classically (at the very least, we do not have even a quasipoly-time algorithm yet), but unlike simulating a quantum system, factoring is a very natural, number-theoretic problem which has been studied for centuries.

Around 2010, Aaronson and Arkhipov [2] introduced boson sampling, showing that there could be an advantage for a non-universal quantum device built using linear optics, and using a *sampling* task rather than a conventional decision problem. Unfortunately, linear optical devices have proved difficult to scale up to the point that the Aaronson-Arkipov proposal would be classically intractable. Independently, Bremner, Jozsa, and Shepherd [29] gave a similar result with similar hardness assumptions, but using a commuting gates model instead of linear optics. The weakest versions of the Aaronson-Arkhipov and Bremner-Jozsa-Shepherd results require the (quite reasonable) conjecture that the polynomial hierarchy does not collapse, and stronger versions (usually to sample within additive error) require domain-specific conjectures about, e.g., the permanents of matrices with random Gaussian entries, or anti-concentration of amplitudes in IQP circuits. Sampling results also include *random circuit sampling*, the problem of measuring a sample from a quantum circuit of randomly chosen gates, with various choices of depth, topology, and gate set of the circuit. Variations of random circuit sampling are the basis of several competing proposals [21] for demonstrating quantum effects in near-term devices, although they often fall somewhat short of a rigorous quantum advantage, one main exception being

[2] Whether *polynomial time* (either P or BPP) captures the efficient computation is a big question on its own, going back to Cobham [32] or Edmonds [45] in 1965. It is even less clear that polynomial time captures efficiency for quantum computations, since the same arguments against the idea still apply (e.g., constant factors could be large, big exponents are not especially efficient) and, for now, we lack real world evidence with quantum computers to support it. We adopt the orthodox view that P, BPP, and BQP capture "efficient" or "tractable" problems, although we will have little use for these classes within this thesis.

[3] In particular, $\mathsf{P} \subseteq \mathsf{BPP}$ and $\mathsf{BQP} \subseteq \mathsf{PostBQP} \subseteq \mathsf{PSPACE}$, so a separation would settle the open question of whether $\mathsf{P} = \mathsf{PSPACE}$ negatively. This would be an enormous breakthrough.

[4] In a single quote: "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."

that of Bouland et al.[23].

Notice that in these quantum supremacy proposals, there is a trend over the years towards what is practically realizable on quantum devices in the near future. Early theoretical results were understandably focused on establishing that quantum computers are a worthwhile goal, potentially useful in some hypothetical sci-fi future. As experimental progress is made, the proposals get closer to might soon be achievable, e.g., using linear optics because the hardware looks promising (photons are in some ways easier to manage than other quantum media). The theoreticians are meeting the experimentalists in the middle, so to speak, and in this thesis we argue that is a good thing. Adapting theoretical results (and especially the models of computation) to the realities of near-term quantum devices can only hasten the advent of quantum supremacy. Moreover, some limitations of current quantum hardware may persist well into the future, so there is value in adapting our models of computation to the devices.

This thesis is a collection of results about models of computation (i.e., mathematical descriptions of computational devices, suitable for proving theorems about). In each case, the model captures some aspect or limitation of near-term quantum devices (NTQD), and thus contributes to our understanding of those devices. There are three properties which appear again and again, which are themes (and therefore chapters) of this thesis.

1. The topology of qubits in NTQDs tends to be a **two-dimensional grid**. Being able to implement quantum gates between arbitrary pairs of qubits is probably *not* realistic in the near-to-intermediate future, so expect local gates.

2. The **Clifford gates** are a special gate set in several areas of quantum computing. While we do not expect Clifford gates to be the *only* gates available on NTQDs, they may be easier, cheaper, or more reliable to implement. If we can make our circuit out of only Clifford gates, so much the better.

3. Depending on the implementation of qubits, coherence time puts an upper bound on the usable lifespan of any qubit. This limits the depth of any circuit we can implement (at least until we have quantum error correction), so **low-depth quantum circuits** are the most realistic for now.

The models of computation we study are not new: quantum low-depth circuits have been studied for twenty years, cellular automata were proposed in the 1940s, and the Clifford group was of mathematical interest in the nineteenth century. Neither are the models of computation especially realistic or fine grained: in a long tradition of theoretical results, we are interested more in asymptotic behavior than constant factors, even when the constant factors are prohibitive. Our goal is merely to explore these three limitations of near-term devices in a mathematically precise way.

We devote Chapter 2 to notation and background. Experts should skim it for our notation for Clifford gates and complexity classes in particular. The meat of the thesis is in the four chapters after that on grid topology (Chapter 3), Clifford gates (Chapter 4), and two chapters on low-depth quantum circuits (Chapter 5 and Chapter 6). Do not be surprised to find these concepts outside their respective chapters, since they are the *themes* of this thesis, not merely chapter headings. Let us briefly discuss each of the three themes before we continue.

First, for the popular quantum architecture of *superconducting qubits*, quantum information resides in superconducting islands, i.e., physical structures on a chip. The fabrication techniques used for quantum chips are not so different from conventional integrated circuits, and in particular they arrange the qubits on a plane. In principle one could arrange the qubits in any planar configuration on the chip, but designers naturally tend toward regular grids, since then any qubit in the middle of the grid has a similar neighborhood and hopefully similar circuitry. With sufficiently many qubits, it is no longer practical to connect every pair of qubits directly, and the architecture shifts towards connecting only nearby qubits (again, making the connections regular may simplify matters). Thus, we expect qubits on a plane, in a grid topology, with local gates between neighbors—and this is exactly what we see in 50+ qubit designs of IBM, Google, and D-Wave [68, 22].

The Clifford gates are interesting enough in their own right to deserve study.[5] In keeping with the theme, however, there are also reasons to believe Clifford gates will be especially important for quantum devices in the future. Some quantum error correcting codes have the feature that some gates can be implemented *transversally* on encoded qubits, i.e., by applying the gate to each physical qubit in all logical qubits. That is, in a code with k physical qubits per logical qubit, transversal operations are performed with k gates. Unfortunately, at most the Clifford gates can be transversal, and they are insufficient for universal quantum computation. There are proposals to achieve universal computation with encoded qubits, but in at least one notable case (using magic states, or $|T\rangle$ states) the non-Clifford gates dominate the cost to the extent that Clifford gates are counted as "free". At any rate, under any quantum code with the transversal property, we can expect the Clifford gates to be the cheapest to implement.

Last, we consider low-depth circuits. Qubits have limited lifetimes, in the sense that the time until an error occurs follows an approximately exponential distribution. On real, current devices with superconducting qubits, the mean time to error is in the tens of microseconds [73]. Even at a clock rate of a few gigahertz (comparable to modern classical devices) the quantum device can perform fewer than 10^5 gates on a particular qubit before it should expect an error. Lowering the error is a major engineering challenge, and mitigating the error with quantum error correcting codes is not yet feasible either, and even once it is possible, error correction will require additional qubits and gates to implement. For now, the best way to minimize this kind of error on a qubit is to minimize the length of time the qubit needs to exist coherently. The time required to execute a circuit can be approximated by its depth (up to constant factors, perhaps), so we wish to study low-depth quantum circuits. Furthermore, low-depth *classical* circuits have enjoyed over four decades of fruitful study, so it is natural to study analogous quantum device (as evidenced by two decades of low-depth quantum circuit study).

Each chapter presents a different result, so let us briefly summarize them before

[5] Indeed, mathematicians studied the Clifford group long before computer scientists.

continuing. Theoretically, grid based computation does not present much difficulty, so Chapter 3 considers an extremely constrained grid-based model of computation, the cellular automaton. We show that a particular cellular automata satisfies a very stringent universality property, and take this as evidence that computation in a grid is possible even under extreme circumstances. Next, Chapter 4 classifies the power of arbitrary subsets of Clifford gates. It turns out a collection of Clifford gates generates one of 57 different subsets of Clifford operations, and we enumerate all of these classes to better understand the structure within the Clifford group from the perspective of building quantum circuits. This brings us to our two low-depth quantum circuit results, which both build off of a recent unconditional separation of shallow circuit classes: constant-depth quantum circuits can do something constant-depth classical circuits cannot [26]. The first result, Chapter 5, extends this result by augmenting the classical circuits with unbounded fan-in AND and OR gates, and proving the unconditional separation persists. The second result, Chapter 6, also makes the classical circuits more powerful, but proves a separation for an interactive task (two rounds of input/output), conditional on standard complexity-theoretic assumptions.

Chapter 2

Background

In this chapter, we briefly introduce the basic concepts and notation for quantum computing with qubits. We also discuss universality, the Pauli group and Clifford group, common gates, and relevant complexity theory.

2.1 Qubits, Measurement, and Quantum Gates

Mathematically, a qubit is the set of unit vectors (under the ℓ_2 norm) in a two-dimensional Hilbert space. The two *classical basis* or *Z basis* vectors are called $|0\rangle$ and $|1\rangle$ where

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The symbol $|\cdot\rangle$ is a "ket", and it denotes a complex vector labeled by the contents of the ket. As you can see, $|0\rangle$ and $|1\rangle$ are just the standard basis vectors e_0 and e_1. We also define the X basis vectors,

$$|+\rangle := \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |-\rangle := \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

On the other hand, when we write, e.g., $|x \otimes y\rangle$ where $x, y \in \{0, 1\}$, we mean either $|0\rangle$ or $|1\rangle$ depending on whether $x \otimes y$ evaluates to 0 or 1, *not* a new vector labeled literally "$x \oplus y$". We hope it is clear from context how each ket is meant to be interpreted.

For this thesis, most of the models of computation are built on a finite array of qubits. Since a bit has two states and a qubit is a two-dimensional Hilbert space, by analogy we use a 2^m-dimensional Hilbert space to represent an array of m qubits. Canonically, we construct this space by taking the *tensor product*, denoted $\otimes$, of m two-dimensional Hilbert spaces (i.e., qubits). The states for an m-qubit system are the unit norm vectors in this 2^m-dimensional Hilbert space.

We can also take the tensor product of vectors, e.g., the m-fold tensor product of $|0\rangle$,

$$|0\rangle^{\otimes m} := |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle,$$

is a state on m qubits. The tensor product symbol is omitted in practice as much as possible, so this would typically be written $|0\rangle |0\rangle \cdots |0\rangle$. We define the *$m$-qubit classical basis vectors* $|0\cdots 0\rangle, |0\cdots 1\rangle, \ldots, |1\cdots 1\rangle$ as

$$|b_1 \cdots b_m\rangle := |b_1\rangle \otimes \cdots \otimes |b_m\rangle$$

where $b_1, \ldots, b_m \in \{0,1\}$ are bits. Thus, all of the following states are equivalent:

$$|0\rangle^{\otimes m} = |0\rangle \otimes \cdots \otimes |0\rangle = |0\rangle \cdots |0\rangle = |0\cdots 0\rangle = |0^m\rangle .$$

Measurement (in the Z-basis) of a single-qubit state

$$|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$$

collapses it to either $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta^2|$, and in the process we learn which outcome has occurred. The probabilities sum to one because $|\phi\rangle$ is a unit vector: $|\alpha|^2 + |\beta|^2 = 1$. Similarly, measurement of an m-qubit state $|\psi\rangle = \sum_{x \in \{0,1\}^m} \alpha_x |x\rangle$ returns $|x\rangle$ with probability $|\alpha_x|^2$, and again the probabilities sum to one because $|\psi\rangle$ is a unit vector. More generally, *orthogonal measurements* divide the Hilbert space into orthogonal subspaces and project the state onto a subspace with probability proportional to the square of its component in that subspace. For example, measuring just the first qubit of $|\psi\rangle$ gives the state

$$|\psi'\rangle = \begin{cases} \frac{\sum_{x \in \{0,1\}^{n-1}} \alpha_{0x} |0x\rangle}{\left|\sum_{x \in \{0,1\}^{n-1}} \alpha_{0x}\right|^2} & \text{with probability } \left|\sum_{x \in \{0,1\}^{n-1}} \alpha_{0x}\right|^2, \\ \frac{\sum_{x \in \{0,1\}^{n-1}} \alpha_{1x} |1x\rangle}{\left|\sum_{x \in \{0,1\}^{n-1}} \alpha_{1x}\right|^2} & \text{with probability } \left|\sum_{x \in \{0,1\}^{n-1}} \alpha_{1x}\right|^2. \end{cases}$$

In this thesis, we are interested in models of quantum computation built on qubits. That is, we start with the qubits in some initial state, manipulate them by some sequence of quantum operations, then measure. Furthermore, we will consider only models of computation where the quantum operations applied in discrete steps (i.e., a gate model), though it is worth mentioning that continuous time models exist where quantum operations can, e.g., continuously rotate $|0\rangle$ to $|1\rangle$ through $|+\rangle$. A *quantum gate* is linear map U from the Hilbert space (however many qubits) to itself, mapping quantum states to quantum states, i.e., preserving ℓ_2 norm. One can show that the ℓ_2 norm is preserved if and only if U is a unitary matrix, i.e.,

$$UU^\dagger = U^\dagger U = I,$$

where $U^\dagger$ is the Hermitian conjugate of U. Clearly the inverse of U is $U^\dagger$, and the product of two unitaries is unitary, since

$$UV(UV)^\dagger = UVV^\dagger U^\dagger = UIU^\dagger = UU^\dagger = I.$$

Thus, the unitary operations on a d-dimensional space form the *d-dimensional unitary* *group*, denoted $\mathrm{U}(d)$.[1]

In both classical and quantum circuit models, we do not assume we can implement *arbitrary* gates (i.e., arbitrary Boolean functions classically, or arbitrary unitaries quantumly). For one, an arbitrary gate has an exponentially long description (assuming an *approximate* unitary quantumly). Instead, one typically assumes a small set of gates with constantly many inputs, and perhaps a uniform family of gates on an unbounded number of inputs. Classically, one takes the AND, OR, and NOT gates, or simply the NAND gate, or all 1- and 2-bit gates. These can be augmented by unbounded fan-in gates for AND and OR, or for PARITY, MOD functions, MAJORITY, and so on. Quantumly, the situation is similiar: we take some subset of 1- and 2-qubit gates which is *universal* in the sense that the gate set can simulate all quantum gates. For example, the CNOT gate defined below, and the set of *all* (n.b., uncountably many) single qubit gates is universal.

Fact 1. *The* CNOT *gate and the set of all single-qubit gates are* universal *for quantum computing in the sense that any unitary $U \in \mathrm{U}(2^m)$ on m qubits may be implemented exactly with a circuit of these gates.*

2.1.1 Quantum Gates

We know in the abstract that quantum gates are unitaries, acting on a small number of qubits. Let us see some examples used in practice, starting with the *reversible gates*, which permute the classical basis states.[2] These correspond to permutation matrices on 2^m dimensions, and every permutation matrix is unitary so they are immediately quantum gates as well, acting on quantum states by linearity.

First among the reversible gates is NOT, which flips the input, exchanging 0 and 1, the only non-trivial 1-bit reversible gate. Next, the SWAP gate is a 2-bit gate which exchanges the two bits, i.e., exchanges 01 and 10. The controlled-NOT or CNOT gate is a 2-bit such that input (x, y) is transformed to $(x, x \oplus y)$ for any $x, y \in \{0, 1\}$. The first bit is called the *control* and the second bit is called the *target*, and the gate is like applying a NOT gate to target if and only if the control bit is 1. In the same vein, the controlled-SWAP gate (or CSWAP or *Fredkin* gate) swaps the second and third (target) bits if and only if the first bit is 1. And finally, we have the controlled-controlled-NOT gate (or CCNOT or *Toffoli* gate), where a (singly)

[1] There is also a question of *phase*. If we recall the definition of quantum measurement, notice that nothing changes if we multiply the entire state by $e^{i\theta}$, in that the measurement outcomes occur with the same probabilities, and the states afterward are multiplied by $e^{i\theta}$. For this reason, people say the *global phase* of a quantum state is irrelevant. It follows that we wish to treat all unitaries of the form $e^{i\theta}I$ as equivalent, so we are actually interested in the *projective unitary group*, $\mathrm{PU}(d)$, defined as the quotient of $\mathrm{U}(d)$ by its center, the set of unitaries of the form $e^{i\theta}I$.

[2] As background, reversible computing was proposed (before quantum computing) as a remedy to Landauer's principle, the idea that destructive operations (e.g., erasing a bit) must consume energy as a consequence of thermodynamics. To avoid this cost, the computation needs to be *reversible* physically, and therefore also reversible logically, and remarkable work of Bennett [15] shows this is possible. Reversible gates are logically reversible and therefore useful in reversible computation, hence the name.

controlled-NOT gate is applied to the second and third bit if the first bit is 1, i.e., $(x, y, z) \mapsto (x, y, xy \oplus z)$.

In addition, we define some common quantum gates. The controlled-sign gate (or CSIGN or CZ) flips the global sign if and only if both inputs are 1, i.e., the diagonal transformation

$$\text{CSIGN} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Similarly, the *phase gate*, traditionally either S or P (we opt for S), is diagonal: $S := \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$. Last but not least, the *Hadamard gate* is $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$.

2.2 Pauli Group and Clifford Group

An entire chapter of this thesis (Chapter 4) is devoted to the Clifford gates on qubits, and they are ubiquitous in quantum computing. We have already seen many Clifford gates above, but let us formally define the Clifford group, and before that, the closely related Pauli group.

2.2.1 Pauli Group

First, recall the *Pauli matrices*, a collection of four 2×2 unitary matrices:

$$\mathtt{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathtt{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \mathtt{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \mathtt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Since the Pauli matrices satisfy the relations

$$\begin{gathered} \mathtt{X} \cdot \mathtt{Y} = i\mathtt{Z}, \qquad \mathtt{Y} \cdot \mathtt{Z} = i\mathtt{X}, \qquad \mathtt{Z} \cdot \mathtt{X} = i\mathtt{Y}, \\ \mathtt{Y} \cdot \mathtt{X} = -i\mathtt{Z}, \qquad \mathtt{Z} \cdot \mathtt{Y} = -i\mathtt{X}, \qquad \mathtt{X} \cdot \mathtt{Z} = -i\mathtt{Y}, \\ \mathtt{X}^2 = \mathtt{Y}^2 = \mathtt{Z}^2 = \mathtt{I} \end{gathered}$$

and $\mathtt{I}$ is an identity element, the set $\mathcal{P}_1 := \{\pm 1, \pm i\} \times \{\mathtt{I}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$ is a group under multiplication. This is the *one-qubit Pauli group*, and it generalizes to the *m-qubit Pauli group* $\mathcal{P}_m := \{\pm 1, \pm i\} \times \{\mathtt{I}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}^{\otimes m}$. A typical element is therefore a m-fold tensor product of the Pauli matrices, which we call a *Pauli string*, and a $\{\pm 1, \pm i\}$ component which we call the *phase*. We typically omit the tensor products in the Pauli string when possible, e.g., writing $\mathtt{IXYZ}$ for $\mathtt{I} \otimes \mathtt{X} \otimes \mathtt{Y} \otimes \mathtt{Z}$.

Let us name the group of phases $\mathcal{Z}_m := \{\pm 1, \pm i\} \times \mathtt{I}^{\otimes m}$, and note that $\mathcal{Z}_m$ is a normal subgroup of $\mathcal{P}_m$. This means the quotient $\mathcal{P}_m / \mathcal{Z}_m$ is well-defined. Each element of $\mathcal{P}_m / \mathcal{Z}_m$ is a coset $\{+P, -P, +iP, -iP\}$ for some $P \in \{\mathtt{I}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}^{\otimes m}$, but we identify each such element with P, its *positive* representative.

A useful property of the Pauli group is that it is a basis for all matrices.

Fact 2. *Any matrix $A \in \mathbb{C}^{2^m \times 2^m}$ can be written as a complex linear combination of $\{I, X, Y, Z\}^{\otimes m}$.*

Define the notation $G^{(b_1,\dots,b_m)} := G^{b_1} \otimes \cdots \otimes G^{b_m}$ where G is a unitary gate and $(b_1,\dots,b_m) \in \mathbb{Z}$ is an integer vector. Another useful fact is that there are subsets of $2n$ Pauli elements which generate the whole group up to phase.

Fact 3. *Any $P \in \mathcal{P}_m$ can be written in the form $P := \alpha X^a Z^b$ where $\alpha \in \{\pm 1, \pm i\}$ and $a, b \in \mathbb{F}_2^m$. Since $X^{e_1}, \dots, X^{e_m}$ generates (by multiplication in the group) all X^a and $Z^{e_1}, \dots, Z^{e_m}$ generates all Z^b, together they generate all of $\mathcal{P}_m$ up to phase.*

Finally, although the Pauli group is not abelian, it has the property that any two elements either commute or anti-commute.

Fact 4. *For all $P, Q \in \mathcal{P}_m$, the commutator $[P, Q] = PQP^\dagger Q^\dagger$ is either $+I^{\otimes m}$ or $-I^{\otimes m}$.*

2.2.2 Clifford Group

Define the *m-qubit Clifford group*, $\mathcal{C}_m$, as the set of m-qubit unitaries (under multiplication) normalizing the m-qubit Pauli group,

$$\mathcal{C}_m := \{U \in \mathrm{U}(2^m) : U\mathcal{P}_m U^\dagger = \mathcal{P}_m\}.$$

That is, a unitary $U \in \mathrm{U}(2^m)$ is *Clifford* if for any $P \in \mathcal{P}_m$, conjugation by U gives an element of $\mathcal{P}_m$. It is immediate that this is a group, and by construction $\mathcal{P}_m$ is a normal subgroup of $\mathcal{C}_m$, i.e.,

$$\mathcal{Z}_m \trianglelefteq \mathcal{P}_m \trianglelefteq \mathcal{C}_m.$$

Since conjugation of a Pauli by a Clifford operation is so common, we define the notation $\bullet : \mathcal{C}_m \times \mathcal{P}_m \to \mathcal{P}_m$ where $U \bullet P := UPU^\dagger$ for any $U \in \mathcal{C}_m$ and $P \in \mathcal{P}_m$.

This definition is a quite abstract, so let us give some examples. First, perhaps obviously, the SWAP gate is a Clifford gate since

$$\begin{aligned} &\mathrm{SWAP}(\mathtt{XI})\mathrm{SWAP}^\dagger = \mathtt{IX}, && \mathrm{SWAP}(\mathtt{IX})\mathrm{SWAP}^\dagger = \mathtt{XI}, \\ &\mathrm{SWAP}(\mathtt{ZI})\mathrm{SWAP}^\dagger = \mathtt{IZ}, && \mathrm{SWAP}(\mathtt{IZ})\mathrm{SWAP}^\dagger = \mathtt{ZI}. \end{aligned}$$

It suffices to check that $\mathtt{XI}, \mathtt{IX}, \mathtt{ZI}, \mathtt{IZ}$ are mapped to Pauli strings because of Fact 3. Any other Pauli string can be written as a product of these elements, and since $U(PQ)U^\dagger = (UPU^\dagger)(UQU^\dagger)$, will be transformed to a product of Pauli strings.

Similarly, CNOT is a Clifford gate:

$$\begin{aligned} &\mathrm{CNOT}(\mathtt{XI})\mathrm{CNOT}^\dagger = \mathtt{XI}, && \mathrm{CNOT}(\mathtt{IX})\mathrm{CNOT}^\dagger = \mathtt{IX}, \\ &\mathrm{CNOT}(\mathtt{ZI})\mathrm{CNOT}^\dagger = \mathtt{ZI}, && \mathrm{CNOT}(\mathtt{IZ})\mathrm{CNOT}^\dagger = \mathtt{IZ}. \end{aligned}$$

All Pauli operations are immediately Clifford gates by Fact 4, which implies $PQP^\dagger =$

$\pm P$ for all $P, Q \in \mathcal{P}_m$. The Hadamard gate (H) and phase gate (R_Z) are also Clifford:

$$H\mathtt{X}H^\dagger = \mathtt{Z}, \qquad S\mathtt{X}S^\dagger = \mathtt{Y},$$
$$H\mathtt{Z}H^\dagger = \mathtt{X}, \qquad S\mathtt{Z}S^\dagger = \mathtt{Z}.$$

The surprising fact, however, is that CNOT, Hadamard, and Phase generate the *entire* Clifford group.

Theorem 5. *Every element of $U \in \mathcal{C}_m$ can be constructed from a circuit of* CNOT, *H, and* R_Z *gates.*

We define a *Clifford circuit* as any quantum circuit built from a basis of Clifford gates, with $\{\mathrm{CNOT}, H, \mathrm{R}_Z\}$ being one possible basis (for the entire group, by Theorem 5). A *Clifford state* or *stabilizer state* is any quantum state of the form $U|0\rangle^{\otimes m}$, where $U \in \mathrm{U}(2^m)$ is Clifford. A equivalent way to define a Clifford state is by its *stabilizer group*,

$$\mathrm{Stab}_{|\psi\rangle} := \{P \in \mathcal{P}_m : P|\psi\rangle = |\psi\rangle\},$$

the set of Pauli elements which fix the state. The X-basis ($|+\rangle$, $|-\rangle$), Y-basis ($|i\rangle$, $|-i\rangle$), and Z-basis ($|0\rangle$, $|1\rangle$) states are all trivially stabilizer states, where

$$|i\rangle = \frac{|0\rangle + i|1\rangle}{\sqrt{2}} \qquad |-i\rangle = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}.$$

The cat state $\frac{|0^m\rangle + |1^m\rangle}{\sqrt{2}}$ is also an example of a Clifford state.

The Clifford gates are *not* universal for quantum computing, which is hardly surprising given that $\mathcal{C}_m$ is a discrete, finite group for any particular m. In fact, the Clifford gates are classically simulable, not only in P, but in the complexity class $\oplus\mathsf{L}$ [4]. We discuss complexity classes in Section 2.3.

2.2.3 Clifford Tableaux

An important representation for Clifford operations (and a good way to prove a simulation result) is the *tableaux representation*. In this section, we explain how tableaux can be derived from first principles, going as far as we need for general background and Chapter 6. The Clifford chapter, Chapter 4, will require a more extensive treatment (see Section 4.2).

Observe that any unitary can be defined (up to phase) by how it conjugates density matrices, i.e., by the set $\{(\rho, U\rho U^\dagger) : \rho \geq 0\}$. Since the Pauli group forms a basis for all matrices, this can be reduced to the set $\{(P, UPU^\dagger) : P \in \mathcal{P}_m\}$. In fact, since $\mathtt{X}^{e_1}, \ldots, \mathtt{X}^{e_m}, \mathtt{Z}^{e_1}, \ldots, \mathtt{Z}^{e_m}$ generate $\mathcal{P}_m$, and conjugation preserves produces (i.e., $UPU^\dagger UQU^\dagger = UPQU^\dagger$) it suffices to write down $U\mathtt{X}^{e_i}U^\dagger$ and $U\mathtt{Z}^{e_i}U^\dagger$ for all i. When U is a Clifford unitary, we also get that $U\mathtt{X}^{e_i}U^\dagger$ and $U\mathtt{Z}^{e_i}U^\dagger$ are in $\mathcal{P}_m$. For example, $S\mathtt{X}S^\dagger = \mathtt{Y}$ and $S\mathtt{Z}S^\dagger = \mathtt{Z}$, from which we can derive that

$$S\mathtt{Y}S^\dagger = iS\mathtt{X}S^\dagger S\mathtt{Z}S^\dagger = i\mathtt{YZ} = -\mathtt{X}.$$

Clearly $SIS^\dagger = \mathtt{I}$, and thus we can derive $S\rho S^\dagger$ for any ρ by linearity. Similarly, $H\mathtt{X}H^\dagger = \mathtt{Z}$, $H\mathtt{Z}H^\dagger = \mathtt{X}$ and

$$\begin{aligned} \mathrm{CNOT}(\mathtt{XI})\mathrm{CNOT}^\dagger &= \mathtt{XX}, & \mathrm{CNOT}(\mathtt{IX})\mathrm{CNOT}^\dagger &= \mathtt{IX}, \\ \mathrm{CNOT}(\mathtt{ZI})\mathrm{CNOT}^\dagger &= \mathtt{ZI}, & \mathrm{CNOT}(\mathtt{IZ})\mathrm{CNOT}^\dagger &= \mathtt{ZZ}. \end{aligned}$$

In general, a Pauli operator $P = \alpha\mathtt{X}^a\mathtt{Z}^b \in \mathcal{P}_m$ can be represented with two m-bit vectors $a, b \in \mathbb{F}_2^m$ and a pair of bits for α, and any Clifford operation is defined by $2m$ Pauli operators $U\mathtt{X}^{e_1}U^\dagger, \ldots, U\mathtt{X}^{e_m}U^\dagger, U\mathtt{Z}^{e_1}U^\dagger, \ldots, U\mathtt{Z}^{e_m}U^\dagger$, the entire operation can be described by a matrix of $2n$ rows with $2m+2$ bits per row. The phase information can be reduced to one bit per row (instead of two) to give a tableau, but since we will not need that part of the tableau for most of thesis (only in Chapter 4, and we have a separate discussion of Clifford tableau), we will skip it and focus on the remaining $2m \times 2m$ binary matrix.[3] We divide the matrix into four blocks,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where $A, B, C, D \in \mathbb{F}_2^{m\times m}$. That is, the ith row of A records the X component of the Pauli string $U\mathtt{X}^{e_i}U^\dagger$, and the ith row of B represents the Z component of the same Pauli string. Similarly, the ith row of C and D represent the X and Z components of the Pauli string $U\mathtt{Z}^{e_i}U^\dagger$. Then we have the following facts.

Fact 6. *The tableau* $[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}]$ *corresponds to a Clifford operation if and only if it is* symplectic. *The matrix is symplectic if and only if $AD^T + BC^T = I$ and both AB^T and CD^T are symmetric.*

Fact 7. *Let $U \in \mathcal{C}_m$ be a Clifford unitary with tableau $[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}]$. Then $U(X^r Z^s)U^\dagger = \alpha X^u Z^v$ for some $\alpha \in \{\pm 1, \pm i\}$ if and only if*

$$\begin{pmatrix} r & s \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} u & v \end{pmatrix}.$$

It follows that, ignoring phase, the tableau for a unitary VU is the standard matrix product of the tableau for U and the tableau for V.

We have not seen complexity classes yet, but matrix multiplication over $\mathbb{F}_2$ is in $\oplus\mathsf{L}$. Hence, we can simulate a sequence of Clifford gates in $\oplus\mathsf{L}$. This is essentially the Aaronson-Gottesman theorem [4], which we slightly extend in Theorem 120 to get measurements for multiple bits.

2.2.4 Clifford Gates

In our final background subsection on Clifford gates, we make a thorough study of single-qubit Clifford gates (which happen to correspond to symmetries of cube, see

[3] One can check that for all $\mathtt{X}^a$ and $\mathtt{Z}^b$, the phase of $U\mathtt{X}^aU^\dagger$ and $U\mathtt{Z}^bU^\dagger$ is either +1 or −1 for all $U \in \mathcal{C}_m$. We formally justify the sign issue in the next section, and in particular Lemma 106.

Figure 2-1), and then the named multi-qubit Clifford gates.

2.2.5 Single-qubit Gates

We group the gates by the type of rotation to emphasize this geometric intuition.

Face rotations: The Pauli matrices $\mathtt{X}$, $\mathtt{Y}$, and $\mathtt{Z}$ (as gates) correspond to 180° rotations about the X, Y, and Z axes respectively. Similarly, we define R_X, R_Y, and R_Z to be 90° rotations (in the counterclockwise direction) about their respective axes. Formally,

$$\mathrm{R}_X = \frac{\mathtt{I} - i\mathtt{X}}{\sqrt{2}}, \qquad \mathrm{R}_Y = \frac{\mathtt{I} - i\mathtt{Y}}{\sqrt{2}}, \qquad \mathrm{R}_Z = \frac{\mathtt{I} - i\mathtt{Z}}{\sqrt{2}},$$

although in the case of R_Z (also known as the *phase gate* and often denoted by S or P), a different choice of phase is more conventional. The clockwise rotations are then $\mathrm{R}_X^\dagger$, $\mathrm{R}_Y^\dagger$ and $\mathrm{R}_Z^\dagger$.

Edge rotations: Another symmetry of the cube is to rotate one of the edges 180°. Opposing edges produce the same rotation, so we have six gates: θ_{X+Y}, θ_{X-Y}, θ_{X+Z}, θ_{X-Z}, θ_{Y+Z}, θ_{Y-Z}. We define

$$\theta_{P+Q} = \frac{P+Q}{\sqrt{2}}, \qquad \theta_{P-Q} = \frac{P-Q}{\sqrt{2}},$$

for all Pauli matrices $P \neq Q$. Note that θ_{X+Z} is the well-known *Hadamard gate*, usually denoted by H.

Vertex rotations: The final symmetry is a 120° counterclockwise rotation around one of the diagonals passing through opposite vertices of the cube. The cube has eight vertices, $(\pm1, \pm1, \pm1)$, and we denote the corresponding single-qubit gates Γ_{+++}, Γ_{++-}, $\ldots$, Γ_{---}. Algebraically, we define

$$\begin{aligned}\Gamma_{+++} &= \frac{\mathtt{I} - i\mathtt{X} - i\mathtt{Y} - i\mathtt{Z}}{2}, \\ \Gamma_{++-} &= \frac{\mathtt{I} - i\mathtt{X} - i\mathtt{Y} + i\mathtt{Z}}{2}, \\ &\vdots \\ \Gamma_{---} &= \frac{\mathtt{I} - i\mathtt{X} + i\mathtt{Y} + i\mathtt{Z}}{2},.\end{aligned}$$

We also define Γ (without subscripts) to be the first gate, Γ_{+++}, since it is the most convenient; conjugation by Γ maps $\mathtt{X}$ to $\mathtt{Y}$, $\mathtt{Y}$ to $\mathtt{Z}$, and $\mathtt{Z}$ to $\mathtt{X}$.

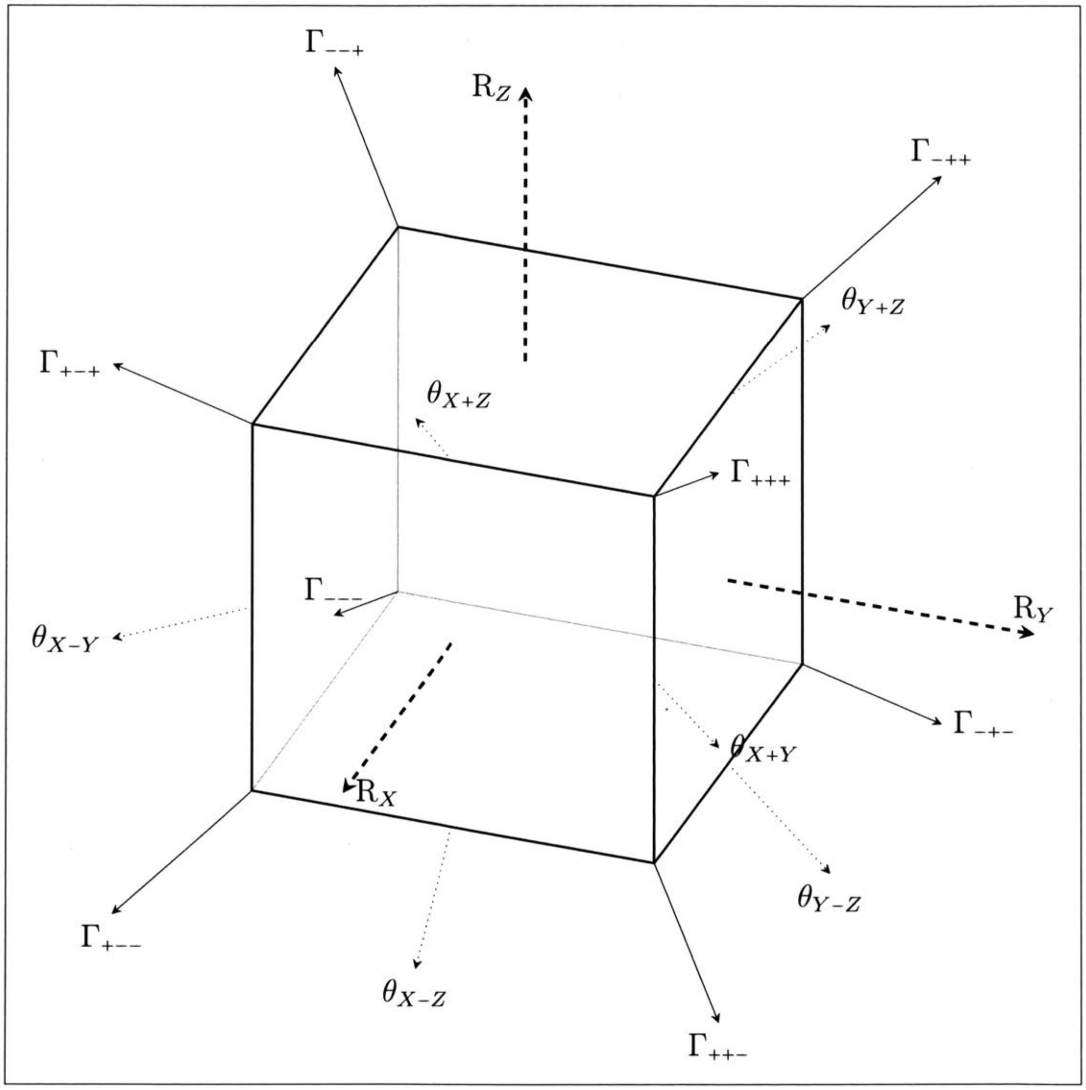


Figure 2-1: Single-qubit gates as symmetries of the cube

Gate	Tableau	Unitary Matrix
X	$\left(\begin{array}{cc\|c} 1 & 0 & 0 \\ 0 & 1 & 1 \end{array}\right)$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Y	$\left(\begin{array}{cc\|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right)$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Z	$\left(\begin{array}{cc\|c} 1 & 0 & 1 \\ 0 & 1 & 0 \end{array}\right)$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
R_X	$\left(\begin{array}{cc\|c} 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}\right)$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$
R_Y	$\left(\begin{array}{cc\|c} 0 & 1 & 1 \\ 1 & 0 & 0 \end{array}\right)$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$
$\mathrm{R}_Z = S$	$\left(\begin{array}{cc\|c} 1 & 1 & 0 \\ 0 & 1 & 0 \end{array}\right)$	$\frac{1-i}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
$\theta_{XZ} = \theta_{X+Z} = H$	$\left(\begin{array}{cc\|c} 0 & 1 & 0 \\ 1 & 0 & 0 \end{array}\right)$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
θ_{X-Z}	$\left(\begin{array}{cc\|c} 0 & 1 & 1 \\ 1 & 0 & 1 \end{array}\right)$	$\frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$
$\Gamma_{+++} = \Gamma$	$\left(\begin{array}{cc\|c} 1 & 1 & 0 \\ 1 & 0 & 0 \end{array}\right)$	$\frac{1-i}{2}\begin{pmatrix} 1 & -i \\ 1 & i \end{pmatrix}$

Table 2.1: Single-qubit gates

2.2.6 Multi-qubit Gates

We now introduce some multi-qubit Clifford gates. We have already seen the SWAP gate and the CNOT gate. A *generalized CNOT gate* is a two-qubit Clifford gate of the form

$$C(P,Q) := \frac{I \otimes I + P \otimes I + I \otimes Q - P \otimes Q}{2},$$

where P and Q are Pauli matrices. If the first qubit is in the $+1$ eigenspace of P then $C(P,Q)$ does nothing, but if it is in the -1 eigenspace of P then $C(P,Q)$ applies Q to the second qubit. Of course, the definition is completely symmetric, so you can also view it as applying P to the first qubit when the second qubit is in the -1 eigenspace of Q.

Observe that $C(\mathtt{Z},\mathtt{X})$ is actually the CNOT gate; Figure 2-2 shows this equivalence, and illustrates our circuit diagram notation for generalized CNOT gates. Also note that $C(\mathtt{X},\mathtt{Z})$ is a CNOT in the opposite orientation. The rest of the *heterogeneous generalized CNOT gates* (i.e., $C(P,Q)$ where $P \neq Q$) are the natural equivalents of CNOT in different bases.

Similarly, $C(\mathtt{Z},\mathtt{Z})$ is better known as the *controlled-sign gate* (denoted CSIGN or CZ), which flips the sign on input $|11\rangle$, but does nothing otherwise. The *homogeneous generalized CNOT gates* (i.e., $C(P,P)$ for some P) are quite different from heterogeneous CNOT gates. For instance, when one CNOT targets the control qubit of

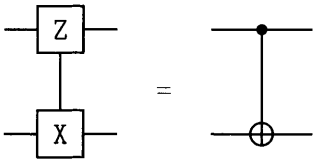


Figure 2-2: CNOT expressed as a $C(\mathtt{Z},\mathtt{X})$ gate

another CNOT then it matters which gate is applied first. On the other hand, two CSIGN gates will always commute, whether or not they have a qubit in common.

It turns out that every two-qubit Clifford gate is equivalent (up to a SWAP) to some combination of one qubit Clifford gates and at most one generalized CNOT gate (see Section 4.9.4). Most classes of Clifford gates are of this form, but there are a handful of classes which require larger generators such as the T_4 gate [5, 53]. For all $k \geq 1$, we definc T_{2k} as the $2k$-bit classical gate that flips all bits of the input when the parity of the input is odd and does nothing when the parity of the input is even. In particular, T_2 is the lowly SWAP gate. Notice that T_{2k} is an orthogonal linear function of the input bits. The matrix of the T_4 gate (over $\mathbb{F}_2$) is

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

In the quantum setting, define the T_{2k} gate as apply this linear transformation to the Z basis vectors.

2.3 Computational Complexity

We assume a familiarity with Turing machines and the basic complexity classes:

- P: decision problems solvable in polynomial time,
- L: decision problems solvable in logarithmic space,
- NP and NL: decision problems solved by non-deterministic versions of the above,
- PSPACE: decision problems solvable in polynomial space.

We will also need $\oplus$L, the class of decision problems solved by an NL machine where we accept if the machine has an odd number of accepting paths and reject if the machine has an even number of accepting paths (i.e., perfectly analogous to $\otimes$P, but logarithmic space).

Now let us define some standard polynomial-size circuit families. For each of the families below, we adopt the convention that $\mathcal{C}^i$ denotes the subfamily of circuits in $\mathcal{C}$ with depth $O(\log^i n)$.

Gate	Tableau	Unitary Matrix on computational basis
$C(\mathtt{X},\mathtt{X})$	$\left(\begin{array}{cc\|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right)$	$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{pmatrix}$
$C(\mathtt{Y},\mathtt{Y})$	$\left(\begin{array}{cc\|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right)$	$\frac{1}{2}\begin{pmatrix} 1 & -i & -i & 1 \\ i & 1 & -1 & -i \\ i & -1 & 1 & -i \\ 1 & i & i & 1 \end{pmatrix}$
$C(\mathtt{Z},\mathtt{Z})$ = CSIGN	$\left(\begin{array}{cc\|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
$C(\mathtt{Y},\mathtt{X})$	$\left(\begin{array}{cc\|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right)$	$\frac{1}{2}\begin{pmatrix} 1 & 1 & -i & i \\ 1 & 1 & i & -i \\ i & -i & 1 & 1 \\ -i & i & 1 & 1 \end{pmatrix}$
$C(\mathtt{Z},\mathtt{Y})$	$\left(\begin{array}{cc\|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right)$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix}$
$C(\mathtt{X},\mathtt{Z})$ = CNOT	$\left(\begin{array}{cc\|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Table 2.2: Two qubit gates. The sign bits are all 0 in the above tableaux so they are omitted. In this table, we adopt the convention (from Chapter 4) that the Pauli basis elements are `XI`, `ZI`, `IX`, `IZ` *in that order.*

- NC: classical circuits of bounded fan-in AND, OR, and NOT gates.
- AC: classical circuits of unbounded fan-in AND, OR, and NOT gates.
- QNC: quantum circuits of CNOT gates and arbitrary single-qubit gates.
- Clifford: quantum circuits of CNOT, Hadamard, and $\frac{\pi}{4}$-Phase gates.

Let us now define another quantum circuit family, CCC, the classically-controlled Clifford circuits.

Definition 8. A classically-controlled Clifford circuit is a quantum circuit of Clifford gates where some gates are labeled by a individual bits of the classical input. To execute a classically-controlled Clifford circuit on an input, we remove Clifford gates labeled by a 0 input bit and keep the unlabeled gates or those labeled by a 1 bit. Then run the quantum circuit as usual.

The name comes from the concept of a controlled gate. Given any n-qubit gate G, we can define an $(n+1)$-qubit controlled gate

$$\chi(G) := \frac{(\mathtt{I}+\mathtt{X})\otimes I_n + (\mathtt{I}-\mathtt{X})\otimes G}{2},$$

which applies G to the last n qubits (the *target* qubits) if and only if the first qubit (the *control* qubit) is 1. For example, CNOT is literally a controlled NOT gate, $\chi(\mathtt{X})$. An alternative definition of CCC is a quantum circuit with the input partitioned into control bits and target bits, where controlled Clifford gates (e.g., CCNOT, C-Hadamard, C-Phase, CZ) are allowed from the control bits to the target bits and arbitrary Clifford gates are allowed on the target bits. The classical input appears as the initial setting of the control bits. See Figure 2-3 for an example circuit.

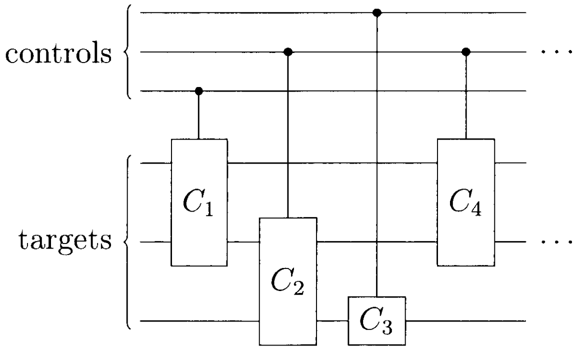


Figure 2-3: CCC circuit for Clifford gates $C_1, C_2, \ldots \in \{\text{CNOT}, H, S\}$.

We also note two important subclasses of the classically controlled Clifford gates.

- L/poly: the family of classically controlled Clifford circuits where the only gates are controlled or not controlled versions of SWAP and X (i.e., NOT),

- $\oplus$L/poly: the family of classically controlled Clifford circuits where the only gates are controlled or not controlled versions of CNOT and X (i.e., NOT).

The names of the classes (deliberately) suggest that they are related to the Turing machine classes L and $\oplus$L. The notation /poly means they are given *polynomial-length classical advice.* In other words, the class is augmented with an (by default, classical) *advice string* of polynomial length (indicated by the poly) which depends on the input length, but not the input. This string is passed to the computational device with the input. For example, P/poly is equivalent to polynomial-size circuits since, in one direction the advice could encode a circuit to simulate (circuits $\subseteq$ P/poly), and in the other direction, any $\mathcal{P}$ computation can be written out as a circuit and then the advice could be hard-coded (P/poly $\subseteq$ circuits).

As the names suggest, these two classically controlled Clifford circuit classes indeed align with advice classes.

Theorem 9. *The class of decision problems solved by a family of* L/poly *classically controlled Clifford circuits is the advice class* L/poly. *Similarly, the class of decision problems solved by a family of* $\oplus$L/poly *classically controlled Clifford circuits is the advice class* $\oplus$L/poly. *Note that the solution to a decision problem is encoded (for this kind of computational device) by the value of the top bit at the end of the computation.*

Proof. A result of Cobham [33] says that L/poly (the advice class) is equivalent to polynomial-length, polynomial-width branching programs. For the unfamiliar, a deterministic branching program is a computational device consisting of a layered directed graph, where the edges of the graph are labeled by input bits and there is only one outgoing edge for each label. A result of [74] says that log space is equivalent to reversible log space, which improves the Cobham result by asserting the layers of the branching program are *permutations.* We leave it as an exercise to verify that a polynomial-length, polynomial-width branching program of permutations is equivalent to a network of swaps (i.e., the classically controlled Clifford circuit class) and thus to L/poly.

Computing the product of a network of CNOT gates is complete for $\oplus$L according to a survey of Damm [39]. It follows that a $\oplus$L/poly family of classically controlled Clifford circuits can be simulated in $\oplus$L/poly, where the advice is to specify the circuit. Conversely, given a $\oplus$L machine that uses advice, translate it to a network of CNOT gates and then hardcode the ones that depend on the advice. □

It will be handy to have circuit characterizations of these Turing machine classes, since we are about to define several kinds of computational problems from circuit families.

Traditionally, a *complexity class* is a set of computational problems, often the set of problems solved by a particular device under certain resource constraints (e.g., limited time/space or depth/size) and acceptance conditions (e.g., the device counts as answering "yes" if it says yes on at least $\frac{2}{3}$ of random inputs). There are many kinds of computational problems:

Decision problems: A decision problem is a subset of strings $L \subseteq \{0,1\}^*$, and a computational device solves that problem if it *accepts* precisely the input strings in L. In other words, standard, deterministic, yes-or-no problems on classical inputs. By default, complexity classes are sets of decision problems. We abuse notation and use, e.g., NC^1 as both family of circuits, and the collection of decision problems solved by NC^1 circuits $\{C_n\}_{n\geq 0}$ where C_n has n input bits and one output bit, and the circuit accepts if and only if the output bit is 1.

Relation problems: A relation problem is defined by a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ on strings. A computational device which takes an input string and returns a string solves a relation problem if the input-output pair satisfy the relation. In the interest of clarity, classes of relation problems will begin with Rel. For example, RelNC^1 is the class of relation problems such that there exists a family of NC^1 circuits which, for any input string, output some string such that the pair satisfies the relation.

Sampling problems: A sampling problem is like a relation problem, but each input defines a distribution over output strings, and the computational device is required to output strings at random from this distribution (sometimes *exactly*, but by default with multiplicative error). Some computational devices are inherently deterministic and thus require a stream of random bits to be provided with the input. Classes of sampling problems will begin with Samp. For example, $\mathsf{SampQNC}^0$ is the collection of sampling problems where the goal is to sample from distributions obtained by measurements on a family of QNC^0 circuits.

Interactive problems: An interactive problem is a generalization from a single round of interaction, e.g., the device gets an input and returns an output, to multiple rounds of interaction. Our main results are about interactive problems that can be solved by shallow depth quantum circuits, but probably not by certain weak classical devices. An interactive problem is distinct from a series of relation problems (or sampling problems) in two ways: the computational device is allowed to keep some state from one round to the next (by explicitly passing bits/qubits for circuits, or by not erasing the work tape of a Turing machine between rounds), and the device is evaluated based on the entire transcript (i.e., the sequence of inputs and outputs). We introduce the prefix Inter to put in front of circuit families to denote an interactive problem. This kind of problem is used in Chapter 6 and the details of that interactive model are described in Section 6.3.

When we augment a circuit family $\mathcal{C}$ with a gate G, we write $\mathcal{C}[G]$ for the new circuit family (and decision problem class). When the gate is a MOD_k gate (i.e., MOD_k outputs 1 if the sum of the input bits is 0 modulo k) then it is traditional to write just k, e.g., $\mathsf{AC}^0[2] := \mathsf{AC}^0[\mathsf{MOD}_2]$. With that, we have the following hierarchy of non-uniform[4] complexity classes:

$$\mathsf{NC}^0 \subset \mathsf{AC}^0 \subset \mathsf{AC}^0[2] \subset \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L/poly} \subseteq \oplus\mathsf{L/poly} = \mathsf{Clifford} \subseteq \mathsf{AC}^1[2]$$

where all containments are presumed strict, but only the ones written $\subset$ are *known* to be strict.

2.4 $\oplus$L-Complete Problems

Finally, we give some more background on $\oplus$L, and justify that simulating a sequence of CNOT and NOT gates is $\oplus$L-hard. We are particularly interested in the following promise problem variant:

Problem 10 (CNOT Multiplication with Cycle Promise (CNOTMult*)). *Given a sequence of* CNOT *gates $g_1, \dots, g_n$, decide whether the product $g_1 \cdots g_n$ is equal to the three-cycle on the first three bits or the identity transformation, promised that one is the case. Each gate is represented by an $m \times m$ binary matrix such that the only non-zero entry (i,j) denotes the* CNOT *gate from bit i to bit j.*

The reduction will go through the following two problems, both of which were already known to be $\oplus$L-complete (e.g., see Damm [39]). We include this material for both completeness and to verify the efficiency of the reductions.

Problem 11 (Layered DAG Path Parity). *A* layered DAG *is a directed acyclic graph where the vertices are divided into an ordered sequence of* layers *such that there are only edges from each layer to the next (no loops, no backwards edges, and no skipping layers).*

An instance of LDAGParity *consists of a binary matrix describing a layered DAG with $n+1$ layers having n nodes each. The goal is to compute the parity of the number of paths from some source node to some target node, where the source is in the first layer and the target is in the last layer.*

Problem 12 (CNOT Multiplication (CNOTMult)). *Given* CNOT *gates $g_1, \dots, g_n$, compute the top right entry of the binary matrix $g_1 \cdots g_n$.*

2.4.1 Reductions

We note that the reductions can be computed in DLOGTIME, that is, with a RAM Turing machine using logarithmic time.

Lemma 13. *For any language $A \in \oplus\mathsf{L}$, $A \leq_{\mathsf{DLOGTIME}}$ LDAGParity.*

Proof. Consider the non-deterministic log-space machine M for the language A. The main idea is to construct a layered DAG where there is a node for each configuration

[4] *Non-uniform* describes a computational device defined on a per input length basis, e.g., most circuit classes or uniform models with advice.

of M (where the *configuration* includes the contents of the work tape, the position of all tape heads, and the internal state) at each time step. There is a connection from a node in layer to the next if M could transition between the corresponding configurations in one time step. We leave it as an exercise to check that, in general, paths in the DAG correspond to computation paths.

All that remains is to check a few details:

- The number of configurations and the running time of the computation are both $\mathsf{poly}(n)$, so by adding dummy configurations or extra steps we can make the DAG square (i.e., m layers of m nodes each).
- Connectivity in the DAG can be decided in $\mathsf{DLOGTIME}$. That is, given two suitably encoded configurations and the input, there is a $\mathsf{DLOGTIME}$ algorithm to decide whether there is a transition from one to the other.
- We assume there is a single accepting configuration, so the number of accepting computation paths is exactly the number of DAG paths between a single source and target.

□

Lemma 14. *Let $A = \{a_{i,j}\}$ be an $n \times n$ upper triangular binary matrix with ones along the diagonal. Then,*

$$A = \prod_{i=n-1}^{1} \prod_{j=i+1}^{n} \mathrm{CNOT}(j,i)^{a_{i,j}}.$$

(assume $\prod_{i=1}^{n} x_i = x_1 x_2 \dots x_n$ and $\prod_{i=n}^{1} x_i = x_n x_{n-1} \dots x_1$ for non-commutative variables x_i.)

Proof. The proof follows from a straightforward calculation using the fact that the matrix for $\mathrm{CNOT}(j,i)$ is the identity matrix except that the (i,j)th entry is 1. □

Lemma 15. $\mathsf{LDAGParity} \leq_{\mathsf{DLOGTIME}} \mathsf{CNOTMult}$.

Proof. The transitions between a pair of layers in a layered DAG can be expressed as an adjacency matrix. Thus, an instance of $\mathsf{LDAGParity}$ is equivalent to binary matrices $A_1, \dots, A_n \in \mathbb{F}_2^{n \times n}$, where the number of paths from a source s_i in the first layer to a target t_j in the last layer is the (i,j)th entry of the product $A_1 \cdots A_n$ over $\mathbb{F}_2$.

Consider the matrix $\mathbf{A} \in \mathbb{F}_2^{n(n+1) \times n(n+1)}$ given below:

$$\mathbf{A} = \begin{pmatrix} I & A_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & A_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & I & A_3 & \cdots & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & I & A_n \\ 0 & 0 & 0 & 0 & \cdots & 0 & I \end{pmatrix}.$$

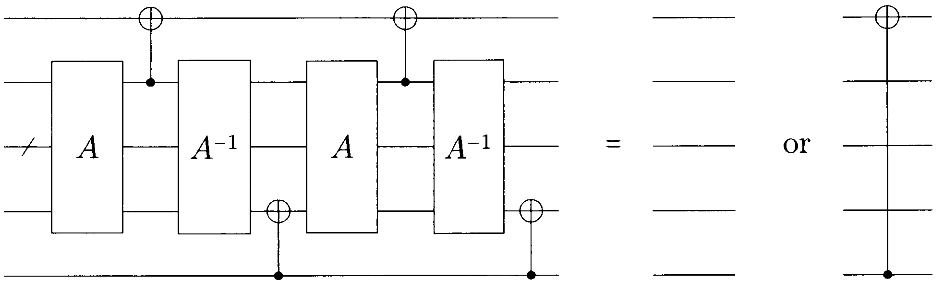


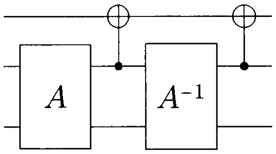
Figure 2-4: Construction of CNOT if top right entry of A is equal to 1.

The inverse, $\mathbf{A}^{-1}$, is upper triangular, and we leave it as an exercise to show that for $i \leq j$, the (i,j)th block of $\mathbf{A}^{-1}$ is $(-1)^{j-i} A_i \cdots A_{j-1}$. In particular, the upper right block is (ignoring sign, since we are working in $\mathbb{F}_2$) the product $A_1 \cdots A_n$. In other words, we can assume without loss of generality that the top right entry of $\mathbf{A}^{-1}$ reports the parity of the number of paths from the source to the sink in the LDAGParity problem. Given the decomposition of $\mathbf{A}$ into CNOT gates by Lemma 14, we can also easily construct the inverse of $\mathbf{A}$ in DLOGTIME, which completes the proof. □

Theorem 16. CNOTMult $\leq_{\mathsf{DLOGTIME}}$ CNOTMult*.

Proof. Let $A = \{a_{i,j}\}$ be the matrix from the reduction in Lemma 15 whose top right entry is ⊕L-hard to compute. We now give a procedure to create a new matrix from A and additional CNOT gates, which isolates the effect of the top right entry—namely, if the top right entry is 1, the product of the gates involved is the 3-cycle on the first three bits; otherwise, the product is the identity gate. We show the entire construction in Figure 2-4.

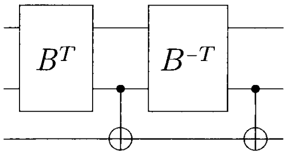
For the analysis, we break the construction into a simpler units. First, we extract information about the first row of A using the following construction:



Call this matrix B. We can determine B by considering its affect on each standard basis vector $e_i \in \{0,1\}^{m+1}$ (where e_i is the binary vector whose only non-zero entry is at location i). We have

$$\begin{array}{ccccccccc}
(1,0,\ldots,0) & \xrightarrow{A} & (1,0,\ldots,0) & \xrightarrow{\text{CNOT}} & (1,0,\ldots,0) & \xrightarrow{A^{-1}} & (1,0,\ldots,0) & \xrightarrow{\text{CNOT}} & (1,0,\ldots,0) \\
(0,1,\ldots,0) & \xrightarrow{A} & (0,1,\ldots,0) & \xrightarrow{\text{CNOT}} & (1,1,\ldots,0) & \xrightarrow{A^{-1}} & (1,1,\ldots,0) & \xrightarrow{\text{CNOT}} & (0,1,\ldots,0) \\
e_{i+1} & \xrightarrow{A} & (0,a_{1,i},\ldots,a_{m,i}) & \xrightarrow{\text{CNOT}} & (a_{1,i},\ldots,a_{m,i}) & \xrightarrow{A^{-1}} & (a_{1,i},1,0,\ldots,0) & \xrightarrow{\text{CNOT}} & (a_{1,i},1,0,\ldots,0)
\end{array}$$

where the last row is for $i > 2$. That is, B is the identity matrix except the first row of A is now (more or less) the first row of B. It is now clear that if we repeat the construction except with the matrix B^T, we can extract the information from the last column:



Notice that this will produce a CNOT gate if the top right entry of A is 1; otherwise, it is the identity. From here, we can simply use this CNOT gate to construct the 3-cycle on the first three qubits. The entire construction of the CNOT gate is shown in Figure 2-4, where some CNOT gates have been eliminated by noticing that the construction of the circuit B is the same if you apply the CNOT gate first or last.

Up to this point, we have been rather agnostic as to the encoding of the CNOT gates. We simply point out that we can convert between most reasonable encodings in DLOGTIME. □

Chapter 3

Grid-Based Computation

As discussed in the introduction, there is reason to believe near-term quantum devices will gravitate towards a grid of qubits. It is the logical conclusion of three points:

1. Current techniques for building quantum chips arrange the qubits on a flat surface, i.e., a two-dimensional plane. For most kinds of quantum hardware, qubits exist in physical constructs on the (planar) quantum chip.[1]

2. Any arrangement of qubits on the plane is *possible*, but there are advantages to a structured grid configuration. It is simpler to design and fabricate a chip if the neighbors of a qubit are always in the same relative position, for example.

3. As the number of qubits increases, there is not enough space on the chip to directly connect all quadratically many pairs of qubits. A configuration with $O(1)$ connections per qubit, e.g., nearest neighbor gates in the grid, *does* scale and is therefore preferable for large numbers of qubits.

We conclude that near-term quantum devices will place qubits on a plane, most likely in a grid with local gates between neighboring qubits. This chapter is about computation within a grid topology.

Surprisingly, much is already known about computation on the grid. Let us start with the obvious: if our model allows arbitrary 2-qubit quantum (or reversible) gates, but only between adjacent qubits in some connectivity graph, then we can simulate gates between arbitrary qubits as long as the graph is connected. That is, to apply a gate to two distant qubits, use swaps to bring the qubits next to each other, apply the desired gate, then swap the two qubits back to their original positions. In the worst case, this incurs overhead proportional to the diameter of the graph, i.e., a factor of $O(\sqrt{n})$ for the grid. From a coarse-grained complexity theory perspective, this polynomial factor overhead is irrelevant since, e.g., it is not enough to move a problem outside BQP. And in some cases the slowdown is unavoidable, in the sense that we cannot hope to do anything non-trivial on a $\sqrt{n} \times \sqrt{n}$ grid in less than $\sqrt{n}$

[0] This chapter is based on work in [100].

[1] A notable exception is in bosonic quantum computing, which bounces photons through optics.

depth, simply because of "light cone"[2] arguments. Much of the remaining work is focused on mitigating the slowdown in practice, or for special algorithms. Let us see a few examples.

For example, early work of Aaronson and Ambainis [1] shows that the quadratic speedup of Grover search *almost* carries over to the grid via a quantum walk algorithm, with only a polylog slowdown. Their work is physically motivated by the limitations of the physical universe in general (rather than near-term quantum devices). They note that in one dimension, search necessarily requires $\Omega(N)$ time since the diameter of the connectivity graph is $\Omega(N)$, but in two or more dimensions, they recover $\tilde{O}(\sqrt{N})$ Grover search by a quantum walk algorithm. On the complexity side, Harrow and Mehraban [60] consider random circuits in the D-dimensional grid, and show that they entangle qubits (under certain important measures) in time proportional to $n^{1/D}$, essentially as fast as the speed of light allows since the diameter of the grid is $n^{1/D}$. This implies a number of promising quantum advantage proposals based on random circuits could be made to work on a @D grid with $O(\sqrt{n})$ depth. In a more practical vein, there is work (e.g., [75]) on automatically compiling circuits under qubit connectivity constraints, in the hope of beating the worst-case slowdown for actual quantum algorithms. They explicitly target existing quantum devices, but offer some flexibility for devices with different connectivity.

The message we get from this work is that computation in a grid is straightforward except for the slowdown, and this can often be reduced in practice, or with more careful algorithms and analysis. How can we add to this picture of grid computation, aside from picking a problem and showing the slowdown can (or cannot) be avoided for that special case? One idea is to further restrict the model of computation to, e.g., cellular automata. Briefly, a cellular automaton is a grid of cells, with each cell having a finite state, which update according to an extremely uniform circuit in both space (i.e., the same gate is applied to each cell) and time (i.e., layers of the circuit are the same). Nevertheless, it is known that classical cellular automata [118, 18, 36], reversible cellular automata [78, 85], and quantum cellular automata [119, 115, 92] are computationally universal in the sense that they can be made to simulate classical, reversible, or quantum circuits respectively. That is, simulating a polynomially large region of a cellular automaton for polynomial time can be P- or BQP-complete for certain CAs.

In the rest of the chapter, we study a particularly demanding property of cellular automata called *physical universality*, defined by Janzing [65] in 2010, with no know examples at time of definition. Physical universality extends the computational universality described above (i.e., P-completeness or BQP-completeness) by demanding the computation take place on cells chosen arbitrarily from the grid. In computational universality results, the simulation/reduction is free to choose how the input bits/qubits are represented (as long as the representation is easy to compute for the

[2] The light cone for some qubit at a particular time within a circuit is the region of influence of that point: the collection of all points in the past that could affect the qubit, and all points in the future that could be affected by it. With local gates on an $\sqrt{n} \times \sqrt{n}$ grid, the influence of a qubit takes $\Omega(\sqrt{n})$ time steps to propagate across the grid, and if some circuit depends on all the bits then $\Omega(\sqrt{n})$ time is necessary.

reduction), and they often appear as *spaceships*, i.e., multi-cellular patterns which repeat in the same orientation but a different position (that is, they move across the cells). *Physical universality* further requires the reduction to work when the input is an arbitrary region of cells in an arbitrary initial state. We will see a formal definition of cellular automata and physical universality below, followed by examples of physically universal CAs in both classical and quantum settings.

We argue that the existence of these physically universal CAs provides new evidence that computation, whether classical or quantum, is robust to restriction to the grid. For example, this result shows that computation on a grid is possible even if we only have one quantum gate, and we are forced to apply that gate locally on the grid in a very uniform way. In that sense, this result is similar to work showing that translationally-invariant Hamiltonians on a grid are universal [72], but in a gate model instead of a Hamiltonian model.

3.1 Cellular Automata

Computation with cellular automata (CA) has a long, interesting history. Early work by von Neumann focused on self-reproducing structures and universal construction in cellular automata [118], although many of the proposed CAs were also computationally universal. In 1982, Berlekamp, Conway and Guy [18] showed that Conway's Game of Life (see also Gardner's article [50]), a particularly simple binary cellular automaton, is Turing-complete. Shortly after, Margolus [78] gave a reversible cellular automaton analogue for Fredkin and Toffoli's (reversible) billiard ball model of computation. More recently, Cook [36] proved rule 110 (a binary, one-dimensional CA) Turing-complete.

A cellular automaton is Turing-complete if it can simulate a Turing machine, but there is a fair amount of latitude in the details of the simulation. For instance, in Life, the simulation uses multi-cellular patterns called *gliders* to transmit and manipulate information. Likewise, Margolus's machine is based on "billiard balls" composed of pairs of particles moving along the same trajectory. Rule 110 builds its simulation on multi-cellular patterns called *spaceships*. In each case, the simulation requires the input tape to be specially formatted (as gliders, spaceships, or pairs of particles) and produces output in a similar format.

Janzing [65] proposed an alternative notion of universality specific to cellular automata. Informally, a CA is *physically universal* if it is possible to implement any transformation on a finite set of cells by initializing the surrounding cells and letting the system evolve (we will see a formal definition in the next section). In other words, computations in a physically universal CA operate directly on cells, without requiring any special input encoding.

For instance, in a physically universal CA, there is some way to configure the cells around a 5-by-5 square region such that after some number of steps, t, the contents of the region will be rotated 90 degrees. Of course, we are not limited to geometric transformations. Given two disjoint 1-by-n regions, we can treat them as n-bit integers, compute their product, and write back the high-order bits of the product in one region and the low-order bits in the other. We can transform the input region by *any* function.

As motivation for physical universality, consider Deutsch's constructor theory [42], which "seeks to express all fundamental scientific theories in terms of a dichotomy between possible and impossible physical transformations". A physically universal CA is a toy universe in which all transformations (on a finite region) are possible. As a special case of physical universality, we can construct any desired pattern, and although this does not imply the existence of self-reproducing machines, it is closely related to von Neumann's work on universal construction. Janzing proposed physically universal CAs as a model of a world in which the boundary between a controller and the physical system it controls can be shifted. He discusses the thermodynamic and Kolmogorov complexity implications of physical universality at length in [65].

Physical universality imposes fairly stringent conditions on a CA. In particular, the CA must be reversible (i.e., it is possible to run the CA backwards), because otherwise information in the input may be lost before we can reach it. For instance, there are exponentially many configurations in Life which evolve to the empty configuration in one step. If one of these configurations occurs in the middle of the input region, there is no way to distinguish it from the others, because we cannot interact with it before it collapses. Reversibility rules out all of the examples above except for Margolus's CA.

Despite being reversible and capable of simulating circuits, Margolus's CA is not physically universal. It is possible to construct immovable, impenetrable walls in Margolus's CA, which contradict physical universality in two different ways. First, information cannot pass through the wall, so input cells within the wall cannot interact with the cells encoding the transformation, and it is impossible to implement an arbitrary transformation. Second, there is no way to move or destroy a wall, so it is impossible to construct certain output configurations. Since these well-known CAs (and many others) are not physically universal, Janzing asked whether physically universal CAs exist, which we answer affirmatively below.

3.2 Physical Universality

Consider a cellular automaton where each cell has a state in Σ, a finite set. We will assume the cells are positioned at integer points $\mathbb{Z}^d$ in d dimensions (and eventually specialize to $d = 2$), but many of the following definitions could be generalized to other topologies. Let $\Omega := \mathbb{Z}^d$ be the set of all cells. Given a set $X \subseteq \Omega$, we say $\overline{X} := \Omega \backslash X$ is the *complement of* X in Ω.

A *configuration* of some subset of cells $X \subseteq \Omega$ is a map assigning each cell in X a state in Σ, and we let Σ^X denote the set of all configurations of X. The notation $\gamma \to \gamma'$ indicates that a configuration $\gamma \in \Sigma^\Omega$ changes into $\gamma' \in \Sigma^\Omega$ after one timestep. Similarly, $\gamma \xrightarrow{n} \gamma'$ means that after n timesteps the configuration goes from γ to γ'.

Suppose $Z \subseteq \Omega$ is a set of cells and we partition Z into disjoint subsets X and Y. A configuration $\gamma \in \Sigma^Z$ naturally splits into configurations $\gamma|_X \in \Sigma^X$ and $\gamma|_Y \in \Sigma^Y$ called the *restriction of* γ *to* X *(resp.* Y*).* Conversely, given configurations $\alpha \in \Sigma^X$

and $\beta \in \Sigma^Y$, we let $\alpha \oplus \beta$ denote the combined configuration of Σ^Z.

Definition 17. Let $X, Y \subseteq \Omega$ be finite sets. Let $f: \Sigma^X \to \Sigma^Y$ be an arbitrary function. Then we say a configuration $\phi \in \Sigma^{\overline{X}}$ *implements the transformation f in time T* if for any configuration $x \in \Sigma^X$ there exists a configuration $\psi \in \Sigma^{\overline{Y}}$ such that

$$x \oplus \phi \xrightarrow{T} \psi \oplus f(x).$$

In other words, a configuration ϕ of $\overline{X}$ implements a transformation f if any initial configuration of x, combined with ϕ, evolves (in T timesteps) to a configuration where Y contains $f(x)$. We do not care about the contents of $\overline{Y}$ after T timesteps.

Definition 18. We say a cellular automaton is *physically universal* if for any input region $X \subseteq \Omega$, output region $Y \subseteq \Omega$, and transformation $f: \Sigma^X \to \Sigma^Y$, there exists a configuration ϕ of $\overline{X}$ and constant t_0 such that ϕ implements f in time t_0.

We say the CA is *efficiently* physically universal if the implementation runs in time t_0, where t_0 is polynomial in

- the *diameter* of X (i.e., the width of the smallest hypercube containing the set) and diameter of Y,
- the distance between X and Y, and
- the computational complexity of f under some appropriate model of computation (e.g., the number of logical gates in a circuit for f).

Janzing's definition differs from our definition in two ways, but it turns out the definitions are equivalent. First, he assumes the input and output region are the same, i.e., $X = Y$. Clearly we can take any region Z containing $X \cup Y$ and extend f to a function $\hat{f}$ on the region Z, which does not depend on cells outside X, agrees with f on Y, and takes arbitrary values everywhere else. Janzing also assumes the transformation is bijective, but by a well-known result of Bennett [15], we can embed an arbitrary function f into a reversible function on twice as many bits, such that for some initial configuration of half the bits, f is a restriction of the reversible function to the other half of the bits.

Let us make a few observations about efficient physical universality. First, the time to implement an arbitrary transformation must grow linearly with the diameter of the input, since information can only travel $O(1)$ cells per timestep. For the same reason, information must (in general) travel from X to Y, so the number of steps is at least linear in the distance between them. Finally, there is an enormous number of transformations from X to Y. Even if Y is a single cell, we need at least $|\Sigma|^{|X|}$ bits specify an arbitrary transformation. We do not have that many bits in polynomial space, so we need a third parameter, e.g., computational complexity, to give us more space for some transformations.

3.3 Cellular Automaton

In this section, we define and motivate a cellular automaton; in Section 3.4 we will prove the CA is physically universal. We use two states, $\Sigma = \{0, 1\}$, in our CA. In figures, we will represent 0 cells as white and 1 cells as black. Our CA is a kind of *block cellular automaton*, meaning that we partition the grid into blocks of cells and update each block according to a *local update rule*. In this case, we use a partition of the grid into 2-by-2 blocks. On the next timestep, we shift the partition one step north and one step east, then back again for the timestep after that. This is known as a *Margolus neighborhood*.

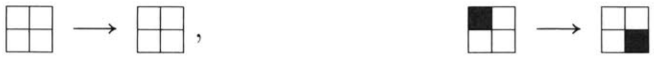
Let the symbol $\boxplus$ denote a set of four cells in a 2-by-2 square. The state of a block is a function from $\boxplus$ to Σ, so we write $\Sigma^{\boxplus}$ for the set of all configurations of the block. In a Margolus neighborhood cellular automaton, the local update rule takes the form $\rho\colon \Sigma^{\boxplus} \to \Sigma^{\boxplus}$, which expresses the evolution of a 2-by-2 block over one timestep.

There are some advantages to using a Margolus neighborhood over the more common *Moore neighborhood*, where each cell is updated based on its current state and the states of its eight nearest neighbors. It is much easier to express CAs with reversibility and conservation properties in a Margolus neighborhood. For instance, a Margolus neighborhood CA is reversible if and only if the update rule ρ is reversible, and the reverse CA has a Margolus neighborhood. Compare this to a Moore neighborhood CA, where it is undecidable whether an arbitrary CA is reversible, and the reverse may not be a Moore neighborhood CA.

Margolus and Toffoli [114] give a very general conversion from block cellular automata to Moore neighborhood CAs. Their conversion augments the set of states with extra information, so each cells knows its current position within the block (e.g., NE, NW, SE, SW). Then the Moore neighborhood rule can tell which neighbors of a cell are in the same block, and update accordingly. This is suitable for simulating a block CA, but note that the extra information must have the correct initial configuration (e.g., a “northeast corner" cell should not be next to another cell of the same type) or the CA will fail. The resulting Moore neighborhood CA is therefore not physically universal, since some configurations of the input region (or output region) are forbidden.

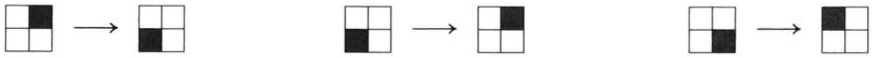
Another way to convert a Maroglus neighborhood CA to a Moore neighborhood is to collapse each 2-by-2 block to a cell (with state $\Sigma^{\boxplus}$). The configuration of a block after two timesteps is a function of its current state and the eight surrounding blocks, so we can make one timestep in the Moore neighborhood CA simulate *two* timesteps of the Margolus neighborhood CA. It will turn out that our (Margolus neighborhood) CA takes an even number of steps to implement any transformation, so the corresponding Moore neighborhood CA is also physically universal.

Let us introduce the local update rules for our CA. We begin with the following two rules:



The rule on the left says that empty (white) space remains empty. If we start with a single black cell in the northwest corner of a block, the rule on the right moves it to the southeast corner. Since the partition also moves one step southeast (or equivalently northeast/northwest/southwest), the black cell is again in the northwest corner and therefore it takes another step southeast. Every step, the black cell will move one unit southeast.

We also have the following three similar rules.



It follows that a lone black cell will move in some direction (northeast, northwest, southeast or southwest as dictated by the corner it starts in) until it encounters another black cell. We think of black cells as *particles* and white cells as empty space, so the remaining update rules describe the interactions between two or more particles. In keeping with the particle metaphor, our CA satisfies the following "physical" laws.

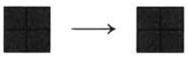
Symmetry: Particle interaction is symmetric. To be precise, given a symmetry $\pi\colon \boxplus \to \boxplus$ of $\boxplus$ (e.g., rotation or reflection), $\rho(x \circ \pi) = \rho(x) \circ \pi$ for all $x \in \Sigma^{\boxplus}$, where $\circ$ denotes function composition.

Conservation: The number of particles is conserved. The number of black cells within a block does not change after one update step, and therefore the number of black cells across the whole grid is constant over time.

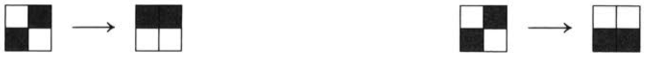
Reversibility: The CA is reversible. A Margolus neighborhood CA is reversible if and only if the update rule, ρ, is bijective.

We argued that reversibility is necessary in a physically universal CA. There is no reason to believe symmetry or conservation are necessary for physical universality, but they are nice properties to have.

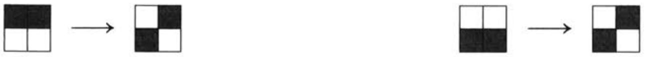
Up to symmetry, there are only four kinds of collisions: two opposing particles, two particles at right angles, three particles, and four particles. Particle conservation forces the four-particle rule to be

,

and severely restricts the two- and three-particle collisions. Symmetry narrows down the set of possible rules even further. For instance, if we have one of these two rules,



then we have both by symmetry, since they are 180 degree rotations of each other. Both rules cannot hold simultaneously because the local update rule is deterministic, therefore neither rule holds. Due to the reversibility of the CA, the reverse situation is similar. That is, we cannot have the rules



since having one would imply the other (by symmetry) and having both contradicts reversibility.

There are eight cellular automata which meet our self-imposed constraints. We

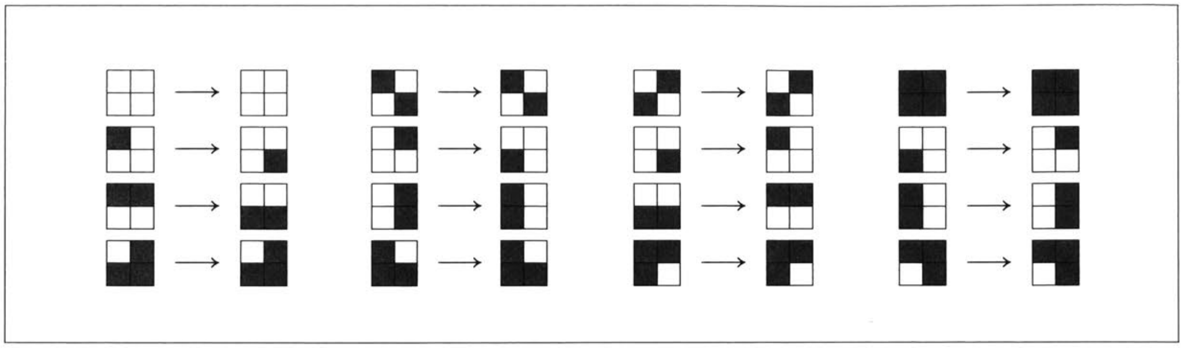
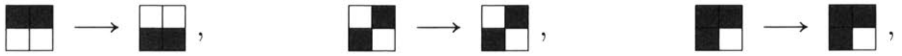


Figure 3-1: A full description of ρ, the local update rule.

will take the CA such that



but at least five of the alternatives have been previously studied, including Margolus's billiard ball CA [78] and Hardy-Pazzis-Pomeau (HPP) gas [59]. The HPP gas is particularly interesting, because it is likely physical universal by a similar proof.

For reference, Figure 3-1 presents the complete update rule for our CA.

3.4 Proof of Physical Universality

As discussed earlier, we may assume the input region X and output region Y are the same, and take X to be a $2n \times 2n$ square of cells aligned to the block boundaries. We assume our boolean function is given as a circuit of m NOR gates. Our goal is to produce an initial configuration $\phi \in \Sigma^{\overline{X}}$ which implements f in time polynomial in n and m.

We divide the computation into stages chronologically as follows.

1. It turns out that particles in our CA behave somewhat like a gas, in that particles diffuse out of any bounded region. We will show that if all particles are initially in a square box (e.g., X), then after some number of timesteps, all particles will be outside the box, moving away in one of four groups of particles.

2. We place particles in the initial configuration of $\overline{X}$ to intercept the particles as they escape from the box, preserving the information represented by the presence or absence of a particle. We rearrange this information into a column formation for easier computation.

3. Next, we show how to implement a NOR operation on the column of bits. Since the NOR gate is universal, we can implement arbitrary circuits, which we use to perform the following transformations.

    - Simulate the CA in reverse to determine the initial configuration of the input, based on the particles that escaped X.

- Apply the function f to the initial contents of X.
- Simulate the CA in reverse again, so we know where to send particles to achieve the desired output configuration.

4. Finally, we carefully position particles such that they converge on X and interact to produce the desired result.

We structure the proof to reflect these stages. For each stage, we place a few more particles in the initial configuration at positions designed to cause collisions between particles at precise locations. By setting up collisions in appropriate locations (and avoiding all other collisions), we can accomplish the objectives for each stage.

3.4.1 Particle Diffusion

Particles in our CA will tend to diffuse out of any finite region into the empty space surrounding it. The CA is deterministic, so all particles escape after finite time, and we can give an explicit bound on that time. Furthermore, there are exactly $4n^2$ places where the $4n^2$ particles in the input region can escape, so we know where to intercept the particles. We express this more precisely with the following theorem.

Theorem 19 (Diffusion Theorem). *Let $X \subseteq \Omega$ be a $2n$-by-$2n$ square region of cells. Partition X into four groups of cells, C_{NE}, C_{NW}, C_{SE} and C_{SW}. Let C_{NE} consist of the southwest corner of every block in X, because particles in those cells will move northeast in the next timestep (barring interference from other particles). Similarly, C_{NW}, C_{SE} and C_{SW} consist of the southeast, northwest and northeast corners of the blocks.*

Let each group evolve as a function of time (i.e., $C_{NE} = C_{NE}(t)$) such that after t timesteps, group C_x is translated t cells in the direction x from its initial position. Let X evolve over time as well by defining

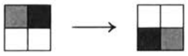
$$X(t) = C_{NE}(t) \cup C_{NW}(t) \cup C_{SE}(t) \cup C_{SW}(t).$$

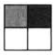
If we start in a configuration with all particles inside $X(0)$ then all particles will be inside $X(t)$ after t timesteps. For $t \geq n$, no block contains more than one cell in $X(t)$, so it follows that there are no particle interactions after timestep n.

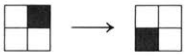
The difficulty in the proof is reasoning about the evolution of the CA under all possible configurations of the region. Our strategy will be to keep track of all positions a particle could reach. To do this, we augment our original CA (the *concrete CA*) with a third state, drawn gray, to represent an unknown, uncertain, or "wildcard" state. We call the new three-state CA the *abstract CA*. The idea is that the abstract CA agrees with the concrete CA on all black-and-white configurations, and when there are gray cells it evolves to propagate the uncertainty as best it can. Note that the new CA will not be physically universal, or even reversible; it is purely a proof tool.

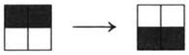
More formally, the abstract CA is over a set of three states, $\Gamma = \{\{0\}, \{1\}, \{0, 1\}\}$, drawn as white, black and gray in figures. We say a concrete configuration $\phi \in \Sigma^X$

is *consistent* with an abstract configuration $\Phi \in \Gamma^X$, and write $\phi \vDash \Phi$, if for all cells $x \in X$, the state ϕ_x is an element of Φ_x. Then the update rule $\rho' : \Gamma^{\boxplus} \to \Gamma^{\boxplus}$ for the abstract CA can be defined as follows: for each $\Phi \in \Gamma^{\boxplus}$, let $\rho'(\Phi)$ be the element of $\Gamma^{\boxplus}$ with the fewest gray cells such that if $\phi \vDash \Phi$ then $\rho(\phi) \vDash \rho'(\Phi)$. That is, if a cell of $\rho'(\Phi)$ is gray (i.e., uncertain) then there are concrete (i.e., black and white) configurations ϕ, ϕ' consistent with Φ which evolve to configurations $\rho(\phi), \rho(\phi') \in \Sigma^{\boxplus}$ having different states for that cell. For example, ρ' contains the following rule

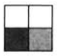


because the two configurations consistent with  evolve as follows,





,

and  is the minimal configuration consistent with both outcomes.

Recall that the abstract CA is a tool to describe about the evolution of many concrete configurations. The precise result we need is stated below, without proof.

Theorem 20. *Let $\Phi \in \Gamma^{\Omega}$ be an abstract configuration. Suppose $\phi \in \Sigma^{\Omega}$ is a concrete configuration such that $\phi \vDash \Phi$. Then clearly $\rho^n(\phi) \vDash (\rho')^n(\Phi)$ for all $n \geq 0$.*

The abstract CA and the theorem above clearly generalize to other CAs. We could easily take an arbitrary binary cellular automaton and construct a corresponding abstract CA as above. In most cases, the result will be uninformative; the "uncertain" gray states tend to proliferate, which tells us nothing about the concrete state. For some CAs, Theorem 20 may tell us little more than "non-blank cells may spread up to t steps away in t timesteps". In the abstract version of Conway's Life, for example, aside from two simple patterns[3], it is challenging to construct any configuration where the uncertain cells neither go extinct, nor overwhelm everything else.

Our specific abstract CA does not suffer from the problems described above because interaction between cells is rare in the concrete CA. It just swaps opposite corners within each block unless exactly three of the cells are black. As a result, ρ' swaps opposite corners within each block unless it is consistent with some three-particle concrete block. This allows us to compute most of the rules on white and gray blocks, shown in Figure 3-2. Observe that on white and gray blocks, ρ' is identical to ρ, except for the three-particle interactions. It follows that gray cells behave like particles, except when exactly three of them collide.

We will now argue that a square of gray cells will diffuse into four groups without generating any new gray cells. By Theorem 20, any concrete configuration of the square must also diffuse, and the diffusion theorem from the main text follows.

Theorem 21. *Let $X \subseteq \Omega$ be a $2n$-by-$2n$ square region of cells. Let $\phi \in \Gamma^{\Omega}$ be the*

[3] Specifically, a blinker and a block, identical to their counterparts in concrete Life, but with uncertain cells.

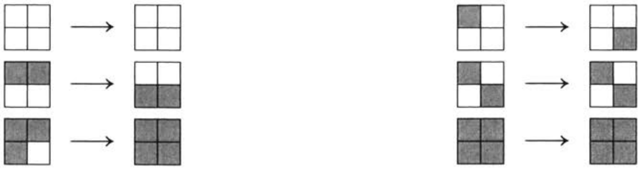


Figure 3-2: The update rule ρ' (suppressing symmetries) on white and gray blocks.

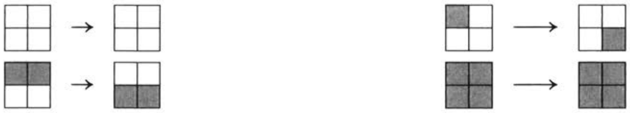
configuration where $\overline{X}$ is white and X is gray. Then we can partition the gray cells into four groups, C_{NE}, C_{NW}, C_{SE} and C_{SW} such that

- *each group contains n^2 gray particles,*
- *the gray particles in C_x all move in the direction x each step, and*
- *the cells outside $C_{NE} \cup C_{NW} \cup C_{SE} \cup C_{SW}$ are all white.*

It follows (by Theorem 20) that for any concrete configuration ψ consistent with ϕ (i.e., with all particles confined to the box X), the region X will be empty after $2n$ timesteps.

Proof. The initial abstract configuration ϕ contains only white and gray cells. Since ρ' maps white and gray blocks to white and gray blocks (see Figure 3-2), the configuration will always be white and gray. Divide the gray particles into groups C_{NE}, C_{NW}, C_{SE} and C_{SW}. We let C_{NE} contain particles in the southwest corners of their respective blocks because a particle in the southwest corner will move to the northeast corner (i.e., in the northeast direction) if there are no other particles. Similarly for C_{NW}, C_{SE} and C_{SW}.

We claim that only the following four rules (and rotations thereof) from Figure 3-2 are used in the evolution of the square.[4] ‘

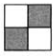
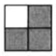


This rule swaps opposite corners, so gray particles move diagonally (as described in the theorem statement) as long as the claim is true. Clearly the claim is true initially, since all of the blocks are full or empty.

Next, we argue that if the blocks move diagonally in their respective directions, then the claim is true. Suppose there is some block where the northeast corner of

[4] Interestingly, since the proof only depends on the behavior of the abstract CA on these four blocks, it generalizes to other abstract CAs (and corresponding concrete CAs). In particular, we can change the two-opposing-particle, three-particle and four-particle interactions in the concrete CA, and it will only affect the corresponding interactions (between gray particles) in the abstract CA. As an example, the HPP gas changes only the two-opposing particle and three-particle interactions, so it satisfies the theorem too. Perhaps this is not surprising for a CA designed as a crude model of gas particles.

the block is white. That is, C_{SW} does not overlap with the block. Since C_{SW} is (the upper left corners of) a square, our block of interest does not overlap C_{SW} vertically or horizontally. If the block is too far north or south relative to C_{SW}, then the same is true of C_{SE}, so the northwest corner of the block will be empty. If the block is too far east or west relative to C_{SW}, then the same is true of C_{NW}, so the southeast corner of the block will be empty. Together, this implies we will never see the blocks

 , 

or their rotations.

A simple induction argument completes the proof. The claim about blocks is true for $t = 0$, so the cells evolve as described, which implies there are no forbidden blocks in the next timestep, and so on. □

The theorem tells us that the information in a square will escape from the region on its own, saving us the trouble of probing into the input region to its contents. Most reversible CAs would scatter that information across a square expanding outwards at the speed of light, mixing the bits with each timestep, creating an urgency to probe the bits to learn them before they spread any further. By the diffusion theorem, this is unnecessary for our CA because all the information from the input region will remain concentrated in $4n^2$ cells forever, or at least until we begin probing with outside particles. Furthermore, after n steps, the contents of those $4n^2$ cells is essentially fixed, so there is no need to probe the cells simultaneously. Recovering the information from the input region is substantially easier because the diffusion theorem applies.

The diffusion theorem has other applications. First, we can avoid unintended collisions between particles. For example, suppose we design the first stage of the computation (say, probing the diffusing clouds from the input region and reorganizing the information into a line of particles). If we put a box around this entire stage, the diffusion theorem tells us when the last collision between these particles can occur. Any subsequent collisions necessarily involve at least two particles from a later stage. Second, the CA is the same running forwards in time as backwards, so the diffusion theorem tells us we can create an arbitrary configuration (momentarily) in the output region by firing four groups of particles into it. This is, in fact, exactly how we will construct the final configuration. Finally, the diffusion theorem upper bounds the running time of any finite configuration, ruling out the idea of unbounded computation in this CA (at least with finitely many particles).

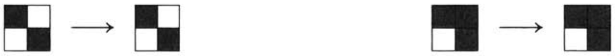
3.4.2 Redirecting Signals

After $2n$ steps, the input particles are in the four groups, as described above. The presence or absence of particles in the groups encodes $4n^2$ bits of information. Every timestep, each particle in C_{NE} moves one step northeast. Similarly, if there is no particle in some cell of C_{NE} then in the next timestep there will be no particle in the cell one step northeast. Either way, a bit of information moves northeast each timestep. We call this moving piece of information a *signal*.

We need to preserve the signals in order to reconstruct the original input bits, and eventually perform the transformation. We cannot hope to perform any computation while the signals are moving in four different directions at the speed of light. Therefore, our first task is to redirect the signals to all move in the same direction, say, northeast. Then we manipulate the signals so they are in the same column and densely packed. We call this arrangement *column formation.*

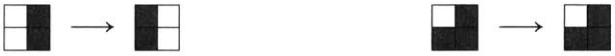
We use carefully placed particles (*control particles*) to control the signals. Since there are no interactions between pairs of particles, we need to intercept a signal with at least two other particles to affect it. Nothing useful happens when three particles meet a signal, so we consider the two ways (up to symmetry) for a signal and two particles to meet. We call these cases *reflection* and *deflection.*

Reflection: Suppose two particles meet head-on, with the signal at right angles. The signal will reflect off of the two particles and move in the opposite direction, as shown by the following rules.



We use reflection to reverse the direction of a particle.

Deflection: Suppose the signal meets one particle head-on and another particle at right angles. The signal will continue on, but the particle at right angles will be reflected or not depending on the signal, effectively copying the signal.



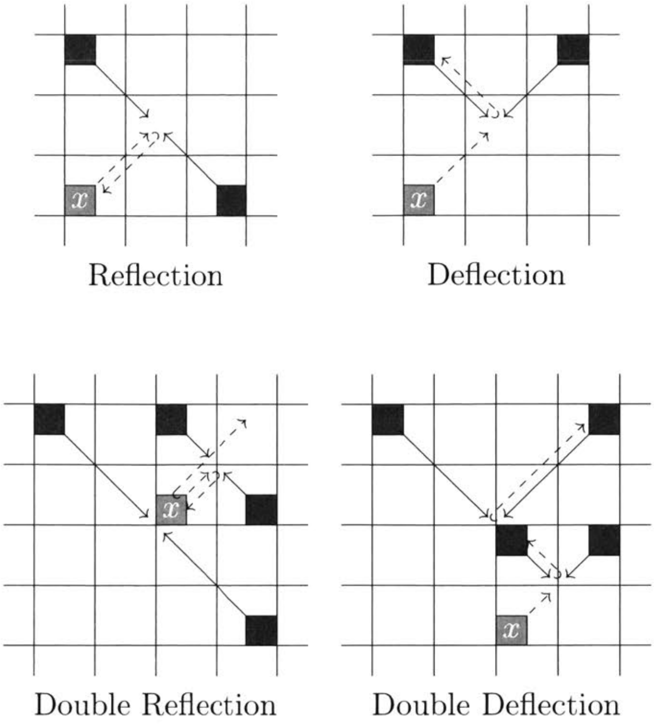
Deflection is useful for creating a duplicate (or inverted) signal at right angles. A deflection can produce particles or signals in all four directions, so we usually follow a deflection with one or more reflections to clean up.

These two operations are the building blocks for all the signal redirection. We also think of the following two operations as primitives, although they are built from deflections and reflections.

Double Deflection: In this maneuver, we deflect the signal 90 degrees, and then deflect it back along its original heading. This changes the diagonal along which the signal is moving, but not the direction. The deflections do not change the path of the original signal, so unless stated otherwise, a double deflection operation also includes a reflection to divert the original signal.

Double Reflection: When we reflect a particle twice, it will end up moving the same direction, along the same diagonal, but further back than before. The time between the two reflections controls how far the particle moves backwards.

Figure 3-3 illustrates all four basic operations, and shows how a signal x is manipulated by the other particles.



Reflection Deflection

Double Reflection Double Deflection

Figure 3-3: Basic operations for manipulating a signal (x) representing a bit of data. The flow of information is indicated by dashed lines.

Notice that collisions only happen with three particles, so reflections and deflections require us to intercept a signal with two other control particles. The control particles intersect at a unique position and time, evident from their initial positions, so we can be sure they introduce no other collisions. However, when there are multiple redirection operations, we must be careful to avoid collisions between control particles implementing separate operations, since these collisions could unintentionally redirect signals. In fact, we use this in the following section to redirect a group of $O(n^2)$ signals with $O(n)$ control particles, but otherwise we wish to have each control particle participate in exactly one operation.

3.4.3 Redirecting Input Signals into a Column

As discussed, it is not feasible to compute on four groups (C_{NE}, C_{NW}, C_{SE}, C_{SW}, as in the diffusion theorem) of signals moving apart at the speed of light, so our first goal is to redirect the groups into an orderly column. The highest priority is actually to get the signals (or copies of the signals) moving all in the same direction, say, northeast. Only then will we worry about constructing a column.

Theorem 22. *We can arrange for the signals (or copies of the signals) in C_{NE}, C_{NW}, C_{SE}, C_{SW} to be directed northeast.*

Proof. Consider each group separately:

Northeast The signals in C_{NE} are already moving northeast. We do not need to do anything with them until we have all the other signals moving northeast too.

Southwest The group C_{SW} is moving southwest, so we can use reflections to reverse the signals. One could reflect each particle individually, but it is easier to reflect the entire group by having a line of $O(n)$ particles moving southeast meet a line of $O(n)$ particles moving northwest. The two lines form a "mirror" which reflects the group.

Southeast The signals in C_{SE} are moving southeast, so we need to deflect them northeast. There is no shortcut to deflect the entire group at once (as we did for C_{SW}), but a single particle moving northwest may intercept many signals in C_{SE}, and we can reuse it for deflection of all those signals in quick succession.

Northwest The procedure for deflecting C_{NW} is essentially the mirror image of the procedure for C_{SE}. However, particles passing through C_{NW} (going southeast) may meet the particles passing through C_{SE} (going northwest). These intersections are harmless unless they meet a signal from C_{SW} moving northeast (having been redirected by our mirror) and reflect it back southwest. We can avoid this kind of interaction by changing the precise timing of C_{SW}, C_{SE} or C_{NW} redirections, so we will assume it does not happen.

□

Now suppose the four groups have been redirected and all the information from the input is contained in a collection of $4n^2$ signals moving northeast. We can use reflections to turn around any other particles or signals moving northeast, so let us assume only the $4n^2$ signals are moving northeast. Assume we also wait (by the diffusion theorem) until there can be no more collisions involving control particles and/or signals defined up to this point. For the sake of avoiding unwanted collisions, we will try to maintain the above points as an invariant. That is, we adopt the following convention until Section 3.4.5, where we must abandon some of it to redirect the particles to the output.

Definition 23 (Hygiene Convention). We assume the following properties:

- All relevant information is contained in a group of signals C moving northeast.
- No control particles are moving northeast.
- Control particles and/or signals from previous stages never intersect.
- There are diagonal planes separating the control particles and/or signals of previous stages from C:
  - Particles moving northwest are above the NE/SW diagonals of all signals in C,
  - Particles moving southeast are below the NE/SW diagonals of all signals in C,

  - Particles moving southwest are below the NW/SE diagonals of all signals in C.

Next, we want to organize the signals into an orderly column.

Theorem 24. *Given a collection of m signals moving northeast, there is a sequence of collisions which reorganizes them into a column formation, i.e., all signals moving northeast in a line with the same x-coordinate.*

Proof. We break this task into two components: move all signals to separate northeast/southwest diagonals, then move signals backwards until their x-coordinate lines up. To change the diagonal of a signal, we use a double deflection operation to copy the signal to a different diagonal, then reflect the original signal backwards, since it is no longer needed. Once the signals are in separate diagonals, we align them in a column by finding the signal furthest west and using double reflection to move the other particles backwards (i.e., southwest) along their diagonals until they are in the same column as the westernmost signal.

The plan above defines a sequence of operations which will redirect the signals into a column. What we need to justify is that we can schedule the operations, i.e., choose a time and position for each one, in such a way that unwanted collisions are avoided. Once we choose the time and position, we can work backwards to determine the initial configuration of the control particles.

We claim that the first double deflection, double reflection, or reflection operation in the plan can be scheduled as soon as the hygiene convention holds. By construction, the control particles in *one* of these operations do not intersect except at the desired collision points. If there were an unwanted collision, it would have to involve two particles from a previous stage, but the hygiene convention rules this out. In fact, the new control particles cannot intersect even one particle from a previous stage because of the plane condition. For example, if our new control particle is moving northwest, it cannot intersect an old particle moving southwest because it touched some element of C, so at that time its NW/SE diagonal was above all old particles moving southwest. Similarly for the other cases.

To be safe, we should wait to schedule the next operation until the hygiene convention is satisfied again. We have only created signals moving northeast, none of the control particles we used move northeast, and we just argued that there will be no intersections between any new control particle and anything else. All that remains is to wait for the new control particles to be separated from C by diagonal planes. This is guaranteed by the diffusion theorem, for example, in time proportional to a bounding box around the signals. At that point the hygiene convention applies again, and we can schedule another operation.

Assuming the signals are originally in a box of size $B \times B$, we can implement each operation in $O(B)$ timesteps. The whole plan involves $O(m)$ operations, so it can be executed in at most $O(mB)$ timesteps. However, we claim (without proof) an optimal schedule can reduce this to $O(m + B)$ timesteps. This is optimal since we may be required to transverse the entire box at least once, and $O(m)$ operations are required in general. The intuition is that each operation puts control particles on

$O(1)$ NE/SW diagonals and $O(1)$ NW/SE diagonals. We can avoid scheduling more than one operation to a NW/SE diagonal, and we can implement the first deflection in several double deflections along a NE/SW diagonal using one control particle from the NE.

Also note that our "plan" is extremely flexible, both in the order of the operations (we can do the double deflections in whatever order is most convenient, then the double reflections in any order, etc.) and the operations themselves (we can choose the order of the signals in the column, or put arbitrary gaps in the column), it seems virtually guaranteed that there is some way to schedule (the first half of) all double deflections in $O(m)$ NE/SW diagonals, and thus $O(m + B)$ time total, and likewise for the double reflections and reflections. □

3.4.4 Logical Operations and Computation

Observe that the local update rule $\rho: \Sigma^{\boxplus} \to \Sigma^{\boxplus}$ is a function from four bits to four bits. Define a four-input/four-output logical gate, the ρ-gate, based on ρ. The ρ-gate is universal,[5] so we could write our circuit in terms of ρ-gates. If we believe we can essentially arbitrarily redirect signals (using the primitives from the previous section), then we can arrange for four arbitrary signals to meet at the same time at some block, where they are transformed (in a single step) according to ρ, thereby implementing a ρ-gate. The outputs of this interaction can then be redirected as well, and so on, to build up an arbitrary ρ-gate circuit.

For the sake of computational efficiency, however, we will now give a different approach, starting from the column of signals we just constructed. We show how to compute the NOR of a set of signals and append it to the (south) end of the column. Since NOR is universal, we can repeat this procedure to implement any logical circuit. When all gates are complete, we have a column with signals representing the input bits, intermediate computations, and output bits. The cost of the computation is proportional to the length of the column.

Theorem 25. *Let $X = \{x_1, \dots, x_{n+m}\}$ be a group of signals moving northeast in column formation, where signals $x_1, \dots, x_n$ encode bits $b_1, \dots, b_n \in \{0, 1\}$ respectively, and $x_{n+1}, \dots, x_{n+m}$ are empty.*

Suppose we are given sets $S_1, \dots, S_m$ where S_i is a subset of $\{1, \dots, n + i - 1\}$, for all i. Define $b_{n+1}, \dots, b_{n+m}$ such that

$$b_{n+i} = \neg \bigvee_{j \in S_i} b_j.$$

Then we can initialize $\overline{X}$ such that after t timesteps, the signal x_j encodes bit b_j, for all $1 \le j \le m + n$. In other words, we can modify the last m signals so each one is the NOR of a given set of earlier signals.

[5] One can implement a controlled-controlled-NOT (i.e., a Toffoli gate), for instance, where two of the inputs control a "dual rail" representation of a bit in the other two inputs.

Furthermore, t is in $O(n+m)$ and the number of particles in $\overline{C}$ is

$$3m + \sum_{i=1}^{m} |S_i|.$$

Proof. Consider an arbitrary signal x_{n+i} where $1 \le i \le m$. We construct particles p_i, q_i and r_i, moving southeast, southwest, and northwest respectively. We position the particles so they all meet (at the same time) with x_{i+n}, assuming they are not reflected earlier. Furthermore, we position p_i north of all signals in X and north of $p_1, \dots, p_{i-1}$. The idea is that if p_i is not reflected back northwest before it meets r_i, then p_i and r_i will reflect q_i into x_{i+n} (i.e., signal x_{i+n} will contain a particle). Otherwise q_i will continue moving southwest and x_{i+n} will continue to be empty.

We also set up a particle s_{ij} for each $1 \le i \le m$ and $j \in S_i$. Let s_{ij} move southwest and intercept p_i and x_j at the same time. The result is that p_i will be reflected if $b_j = 1$, unless p_i has already been reflected. Hence, p_i only reaches x_{i+n} if $b_j = 0$ for all $j \in S_i$. It follows that

$$b_{n+i} = \overline{\bigvee_{j \in S_i} b_j},$$

as desired.

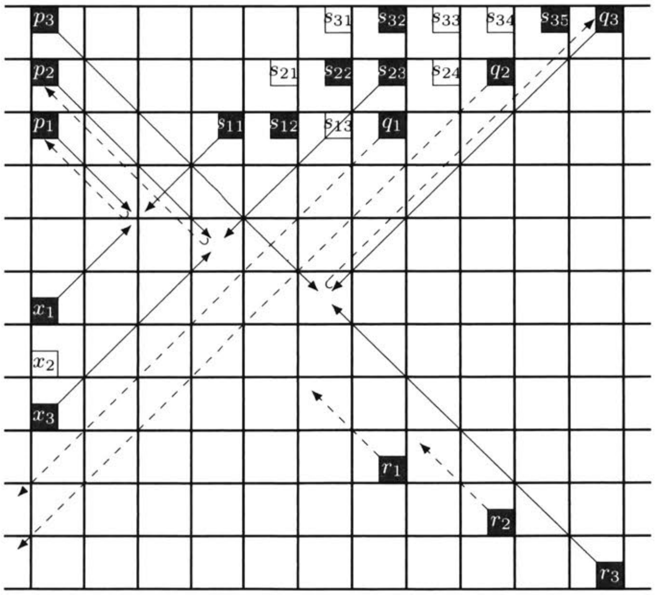


Figure 3-4: An example of a small computation on bits x_1, x_2, x_3.

Observe that only $x_1, \dots, x_{n+m}$ and $p_1, \dots, p_m$ are moving eastwards (i.e., northeast or southeast). Each p_i is positioned to collide with the corresponding x_{i+n}, so they must be in the same column. It follows that all eastbound particles are in the same column. Collisions require at least three particles, at least one of which must be eastbound, so all collisions must occur in the same column as the x_js. We positioned the particles (p_i, q_i, r_i and s_{ij}) based on where and when they meet the x_js, so the only collisions are the ones we have already discussed.

Suppose we place p_1 immediately above x_1 and $p_2, \ldots, p_m$ in order above that. Then clearly p_m reaches x_{m+n} in $t = O(m+n)$ time. It is also clear that we use only $3 + |S_i|$ particles to modify x_{i+n}, and hence only $3m + \sum_{i=1}^{m} |S_i|$ particles altogether. □

Figure 3-4 shows the construction on three bits, $x_1 = 1$, $x_2 = 0$ and $x_3 = 1$. We add $x_4 = \overline{x_1 \vee x_2} = 0$, $x_5 = \overline{x_2 \vee x_3} = 0$ and $x_6 = \overline{x_2 \vee x_5} = 1$ over the course of the computation. The black cells are particles, and white cells are locations where particles are conspicuously absent. For instance, we could place particles at s_{13}, s_{21}, and so on, to specify a different circuit. We use arrows to indicate the motion of key particles, and we use a dashed arrow if the particle does not interact with any other particles. We can see that p_1 and p_2 are deflected (by x_1 and x_3 respectively), but p_3 makes it all the way to q_3 and r_3, so it can reflect q_3 into position as x_6.

We use our ability to perform a sequence of computations, where the original transformation, $f : \Sigma^X \to \Sigma^X$, is somewhere in the middle. Before we can apply f to the initial configuration, $x \in \Sigma^X$, we must decode x from the $4n^2$ signals captured from the four square groups. Similarly, after we have computed the final configuration $f(x) \in \Sigma^X$, we need to encode it as a set of $4n^2$ signals. We arrange the signals into four groups, which converge on X to produce $f(x)$. This last step is the subject of the following section.

3.4.5 Output Stage

The final step of the construction is to insert the desired configuration, $f(x)$, into X, the output region. The diffusion theorem tells us we can insert $f(x)$ by making four groups of carefully designed signals meet in the output. We can compute whatever we want, so assume the necessary signals were computed in the previous section, so all that remains is to redirect a column of $4n^2$ signals into four groups, converging on X. This is essentially the reverse of an earlier step, where we redirected the four groups of signals into a single column.

Unfortunately, literally reversing our steps (in an attempt to exploit the reversibility of the CA) is not possible. To illustrate, suppose we implement our procedure and let the system evolve up to the point where the contents of the input region are laid out as a column of signals (i.e., just before we begin computation). Then we change some of the signals in the column, and run CA backwards for the same number of timesteps. In this situation, the contents of the input region will not reflect the changes we made to the column. The problem is that many cells, not just the the ones in the column, depend on a given cell of the input. Our construction copies information every time we perform a deflection operation, and we leave intermediate steps around in our computation. We have to change all of these cells to change the value of the input cell. Since there are too many cells to practically change, we must find another way to populate the output. See our discussion of *reversible physical universality* in Section 3.5.

Instead, we describe a procedure to redirect a column of signals four groups C_{NE}, C_{NW}, C_{NW}, C_{SE} converging on the output. We use the same primitive operations as we did to redirect the input signals. The added challenge for output signals is that they must arrive simultaneously, which we ensure by designing "slack" into the paths of the signals, so we can adjust the length of the path to make the signal arrive at the correct time.

Let X be the square region, and divide the output signal into four groups, C_{NE}, C_{NW}, C_{SW}, and C_{SE}, where the subscript indicates the direction of motion of the group when it eventually converges on X. Initially, the signals are in column formation, moving northeast, along with signals leftover from the intermediate computations. We make a few preliminary adjustments to the signals as follows.

1. Reflect all signals in the column except the $4n^2$ designated signals. We do not want any interference.

2. Use a series of double deflections to move the $4n^2$ designated signals so they are approximately northeast of X. That is, they are not too far from the northeast/southwest diagonal passing through the center of X.

3. Use further double deflections and double reflections to rearrange the signals into four square formations corresponding to the four groups. Within a group, each signal should have the same offset from its intended destination, except within C_{SW}. We will move C_{SW} into final position by reflection, so we expect each signal to have the same offset to the *reflection* (over a northwest/southeast diagonal) of its destination. For C_{SW} and C_{NE}, we expect the offsets to be strictly northeast (no steps northwest or southeast).

This should all be familiar from Theorem 24, and the last point is essentially Theorem 24 in reverse.

At this point, we wait until the signals are, as a group, some suitably large distance D northeast of X. The plan is to have the signals diverge at this point, and converge on X in approximately $3D$ steps, at some target time t. Conceptually, we redirect the four groups as follows.

Southwest Signals in C_{SW} continue northeast until they are $2D$ units from X, and then we reflect them back towards the origin. No other manipulations are necessary because the signals in C_{SW} are already in the correct square formation.

Northeast We reflect signals in C_{NE} back to the southwest, and let them travel $2D$ units before reflecting them northeast again. The two reflections are sufficient because C_{NE} is already in a square formation.

Southeast We deflect signals in C_{SE} northwest for about D units. Then we deflect the signals southwest for D units, and finally southeast for the final approach. The final deflection (to the southeast) is different from the other deflections; we use the incoming signal from the northeast and a particle from the southwest to reflect a particle coming from the northwest because a particle from the southeast would have to pass through X at an awkward time. In other words, there is a group of particles in square formation moving southeast towards X, and the final deflection whittles away particles to form signals.

Northwest We redirect signals in C_{NW} as the mirror image C_{SE}. We will see that it is not an exact mirror image in practice.

There are two practical issues. First is parity: all of the primitive operations except double deflection do not change the parity of the position. That is, if a particle is x blocks right of its target position and y blocks above, then it would require double deflections to maneuver the particle into position if $x + y$ is odd. Since we plan to use only reflections for two of the groups, it is best if those signals have the correct parity at the end of the computation stage. Second, we cannot redirect all four groups simultaneously. Even if we could, deflecting C_{SE} and C_{NW} would likely create unwanted collisions. As a result, there may be some distance (and time) between the ideal initial redirection of a signal, and the actual initial redirection. Each signal still needs to arrive at the right time, so we make corrections as follows.

- Define D so that the path of group C_{SW} does not require correction. The other groups may be delayed, but we assume D is large enough that the other groups will be deflected long before we need to reflect C_{SW}.

- As long as the initial reflection of C_{NE} is less than D steps late, it will pass through X before the designated time t. As the time remaining (until the groups converge) decreases, the distance between C_{NE} and X increases. At some point the time remaining is equal to the distance, and we schedule the second reflection for that point.

- We adjust signals in C_{SE} using the second deflection, at a location approximately $2D$ cells north of its final location. Recall that we want each signal in C_{SE} to meet a particle moving southeast, so we must deflect the signal southwest when it is in the same row of cells as the particle. Then the signal is to the east of the particle, so the two will eventually collide.

  Observe that if we delay the first deflection, it does not change the time of the second deflection, since the signal is moving north (either northeast or northwest) at the same rate before and after the first deflection. The delay does move the location of the second deflection east.

Finally, we need to argue that this can all be done without any undesired collisions. Since the undesired collision involves at least three particles, those be control particles and/or signals involved in collisions from three different directions. If two or more of the signals come from nearby collisions (say, within $\frac{D}{2}$ timesteps) then we say the collision is *local*. The nearby collisions must involve the same group, so we can eliminate a local collision by rescheduling the collisions for a single group; there is plenty of time and space to do schedule the collisions for a group, so we leave this as an exercise. Assume there are non-local collisions.

There are only two areas where three intentional collisions could occur at the right time to have an unintentional non-local collision. First, there are collisions on all sides of the output region, but on three sides (NW, SW, SE) the only collisions are the ones which produce the output signals as intended. Second, the area D units northeast of the output region is also D units from collisions involving C_{SE}, C_{NW}, and C_{SW}. The concern is that control particles from the C_{SE} and C_{NW} collisions (which deflect each of those groups to the southwest) will cross in this area and interfere with C_{SW}. This can be corrected by scheduling the initial deflections for C_{SE} and C_{NW} on disjoint NW/SE diagonals. Since all the signals are intentional, the four groups of signals converge on X and create the desired configuration at time t. This completes the proof of physical universality.

3.5 Reversible and Quantum notions of Physical Universality

First, there are some intricacies to even define quantum cellular automata, since the update rule must be unitary. Like reversible cellular automata, the easiest way to do this is with a generalization of Margolus neighborhood CAs called *partitioned cellular automata.* A partitioned cellular automaton divides the cells into many congruent blocks and applies the same update rule to each one, then again with a different partition (and possibly different update rule), through constantly many partitions/update rules in a periodic cycle. For example, a special case of this is a *layered cellular automaton*, which has ℓ *layers* (i.e., parallel tracks) of cells arranged along $\mathbb{Z}$. In one timestep, we update the cells in all layers at each position according to some reversible $\rho\colon \Sigma^{\ell} \to \Sigma^{\ell}$, then shift all of the cells of layer i to the right s_i units, where $s_1, \ldots, s_\ell \in \mathbb{Z}$. See follow-up work of Salo and Törmä [99] for an explicit CA; they construct a one-dimensional physically universal CA by flattening the CA above to 4 layers of different speeds.

Clearly a partitioned cellular automaton or layered cellular automaton amounts to an extremely uniform circuit with one kind of gate per layer, and a short cycle of distinct layers (two layers for our purposes; one of the layers being shifts in the case of layered CAs). The quantum analogues of these CAs work the same way, but they are extremely uniform circuits of quantum gates. To avoid trouble with infinite dimensional Hilbert spaces, we insist that there is a *quiescent state*, and if all of the cells are initialized to this state then the update rules leave it as it is. Hence, if all but finitely many of the states are quiescent, computing the next timestep is a finite problem.

Now we turn to the subject of *reversible* physical universality as a warm up to quantum physical universality. When we wish to implement a reversible transformation, i.e., one implemented by a reversible circuit, we can demand a notion stronger than ordinary physical universality: we require that at the end of the computation, the state of the cells *outside* the output region is independent of the state of the input cells at the beginning of the computation. That is, there is a fixed state $P \in \Sigma^{\overline{X}}$ at the beginning, and a fixed state $P' \in \Sigma^{\overline{X}}$ at the end of the computation, and for any $x \in \Sigma^X$, running the CA for t steps maps

$$P \oplus x \xrightarrow{t} P' \oplus f(x).$$

An interesting consequence is that if we replace the contents in the output with some $y \in \Sigma^X$ then we can run the CA backwards to compute the inverse: $P' \oplus y \xrightarrow{t} P \oplus f^{-1}(y)$. Note that this is only possible when f is reversible; otherwise a counting argument will prevent us from running the computation both directions like this. Formally, we define reversible physical universality as follows.

Definition 26. We say a cellular automaton is *reversibly physically universal* if for any input/output region $X \subseteq \Omega$, and bijection $f \colon \Sigma^X \to \Sigma^X$, there exists a configuration ϕ of $\overline{X}$ and constant t_0 such that ϕ implements f in time t_0, and the configuration of $\overline{X}$ at time t_0 does not depend on the input configuration.

Our physically universal CA is *not* reversibly physically universal for a very simple reason: it conserves (among other things) the total number of particles (i.e., 1 states). That is, the total number of particles is the same in the initial configuration as the final configuration, and since the configurations of $\overline{X}$ are fixed, this imposes a relationship between the number of particles in the input and output. In fact, we must have the *same* number of particles in X before and after (and the same number of particles in $\overline{X}$ before and after), otherwise either the empty or full input configuration would be a problem. Since most bijections do not have this property, our CA cannot be physically universal. Ordinarily, we see the fact that the CA conserves particles as natural and aesthetically pleasing, since it feels more like physical system, but in this case it is a drawback. Instead, we define a layered CA with the properties we want.

We wish to define a layered CA that satisfies something like the diffusion theorem. To this end, we define *depleted* configurations.

Definition 27. We say a configuration is *depleted* if there is at most one particle per layer, and the particles are ordered by speed. That is, if $s_i < s_j$ then all particles in layer i are to the left of all particles in layer j.

The idea is that there are no more collisions between particles in a depleted configuration, i.e., no more computation.

Theorem 28 (Layered CA diffusion). *Consider a CA on $2k$ layers with velocities $-k, \ldots, -1, 1, \ldots, k$. Suppose the update rule applied to $(x_{-k}, \ldots, x_{-1}, x_1, \ldots, x_k)$ is such that it does nothing if either $x_{-k} = \cdots = x_{-1} = 0$ or $x_1 = \cdots = x_k = 0$.*

Then any finite configuration (say, confined to a region X of n cells) will be depleted in $O(kn)$ timesteps.

Proof. The crucial fact is that if there are no particles moving right (i.e., $x_1 = \cdots = x_k = 0$) then the CA does nothing. If there are no right-moving particles in the region $(-\infty, j]$ then none are created there by the update rule, and even the slowest right-moving particle moves one step, so after one step the region $(-\infty, j+1]$ is free of right-moving particles. Similarly, if there are no left-moving particles in $[j, \infty)$ then in the next step there will be no left-moving particles in $[j-1, \infty)$.

Note that there are initially no right-moving particles in $(-\infty, 0]$ and no left-moving particles in $[n+1, \infty)$, so in $\lceil \frac{n+1}{2} \rceil$ timesteps these two regions meet. At that point, every location on the line has either no left-moving particles or no right-moving particles, so there can be no interaction. All we need to do is wait for the particles to sort themselves by speed.

At time $n/2$, all the right-moving particles are between position $\frac{n}{2} + O(1)$ and position $\left(1 + \frac{k}{2}\right) n + O(1)$. The worst case for sorting is if a particle of speed k is at the left edge and a particle of speed $k - 1$ is at the right edge. In that case, it will take $(1 + k)n/2 + O(1)$ steps for the particle to catch up, for a total of $t = (1 + \frac{k}{2})n + O(1) = O(kn)$ steps for the configuration to be depleted. □

As long as we define a layered CA satisfying the conditions of Theorem 28, we will have a diffusion theorem. In addition to diffusion, we want a few other properties.

- We want interactions involving fewer than three particles to be trivial, so that it is easy to control exactly where and when non-trivial interactions happen.
- We do not want the update rule to satisfy any conservation laws:
  - The total number of particles cannot be conserved.
  - The number of particles of a given velocity, speed, or direction cannot be conserved.

For example, we propose a CA where $k = 4$ and the update rule is defined as follows, where we implement the first pattern that matches.

- if $x_{-4} = \cdots = x_{-1} = 0$ or $x_1 = \cdots = x_4 = 0$ then do nothing,
- if $x_{-4} = \cdots = x_{-1} = 1$ then cyclically permute $x_1 \to x_2 \to x_3 \to x_4 \to x_1$,
- if $x_1 = \cdots = x_4 = 1$ then cyclically permute $x_{-1} \to x_{-2} \to x_{-3} \to x_{-4} \to x_{-1}$,
- if exactly one of x_4, x_3 and exactly one of x_{-3}, x_{-4} are set to 1, then swap x_2 or x_1 (corresponding to whether x_4 or x_3 is set) with x_{-1} or x_{-2} (corresponding to whether x_{-3} or x_{-4} is set),
- if exactly three of x_4, x_3, x_{-3}, x_{-4} are set, then apply a Toffoli gate to the corresponding three bits of x_2, x_1, x_{-1}, x_{-2},
- otherwise, do nothing.

Claim 1. *The CA above is reversibly physically universal.*

Sketch. It is immediate that CA satisfies the conditions of the layered diffusion theorem (Theorem 28), so in $O(n)$ steps the input region will be depleted. At that point, we use the ability to cyclically permute the layers to adjust the speed of the particles, and we use the ability to swap x_1 and x_{-1} to reverse some of the particles. With a bit of work, we can arrange for all the signals from the input to be moving at velocity 1, for example.

We can arrange for any three of these signals to meet, perform a Toffoli, and then return them to their original positions. This is a substantial gadget to justify, but we have the advantage that at least three particles/signals must meet before anything non-trivial happens, so we leave it as an exercise. Furthermore, unlike the original physically universal CA, when we perform swaps, cyclic permutations, and Toffoli gates, the positions of the outgoing control particles (i.e., after the collision) do not depend on the signals. We leave the details as an exercise.

Finally, we redirect the signals into the output region. This *is* literally the reverse of extracting signals from the input region, so with some care (again, an exercise), we should be able to reverse our procedure for the input. Since any reversible transformation can be constructed from Toffoli gates, we can do the same as for the original physical universality proof: collect the input signals, decode them to determine the initial contents of the input region, compute on those bits, and then encode them for reinsertion into the output region. The only difference is that since the positions of the outgoing control particles do not depend on the signals, at a macroscopic level the final configuration outside the region (i.e., all control particles) does not depend on the signals inside the region. □

A similar argument constructs a *quantum* physically universal CA. First, the definition of physical universality for a quantum CA, although at this point one could probably guess it.

Definition 29. A quantum cellular automaton (QCA) is physically universal if for any region $X \subseteq \Omega$, and any unitary $U \in \mathrm{U}(2^X)$ operating on X, there exist finite configurations $|\phi\rangle, |\phi'\rangle$ on $\overline{X}$ and a time t such that for any state $|\psi\rangle$ on X, the state $|\phi\rangle \otimes |\psi\rangle$ evolves to $|\phi'\rangle \otimes U|\psi\rangle$ in exactly t timesteps.

There are two differences of note:

- Quantum universal gate sets exist with one- and two-qubit gates, so we can get away with $k = 3$, i.e., six layers,
- We must define a quantum configuration to be *depleted* if all the classical basis configurations are depleted. The layered diffusion theorem, Theorem 28, still holds, because requires a new proof.

Otherwise, the proof goes through more or less as with the reversible CA. We omit the details since they can be found in Schaeffer [101].

We conclude that classical, reversible, and quantum variants of physical universality are all achievable by essentially the same argument. In all cases the physical universality is *efficient* in the sense that the time t is polynomial in the size of the input/output region and the depth of the circuit. It remains unclear whether a qualitatively different approach (say, without a diffusion theorem) can achieve physical universality, and whether it can be extended to *unbounded* computation.

It is unlikely that near-term quantum devices will pursue a cellular-automaton-like model where all the gates are the same, and have to be applied uniformly across the entire grid, so the results of this chapter may not be directly applicable to near-term hardware. Nonetheless, the fact that physically universal CAs (both classical and quantum) exist demonstrates that computation on the grid is possible under the most extreme circumstances. And the grid, if not cellular automata in general, *will* be a feature of near-term quantum devices, or more generally, physical systems. For instance, as discussed earlier, these results can also be connected to translationally invariant Hamiltonians [72], which are of interest for physical reasons.

Chapter 4

Clifford Gates

The Clifford gates[1] are extremely well studied and fundamental to quantum computation in many ways. First and foremost, the Clifford gates are connected to error-correcting codes, which may be integral to the development of a general-purpose quantum computer. The important family of *stabilizer codes* [52] are built on the stabilizer formalism, meaning that the encoded states are stabilizer states (or Clifford states). In particular, Clifford circuits suffice to encode or decode each logical qubit, and to compute the syndrome. Since most of the work in these error-correcting codes is to perform Clifford gates, we can only achieve error-corrected general-purpose quantum computers when the error rate is sufficiently low *on the Clifford gates*, and not before then.

Conversely, the Clifford gates are also exactly those operations which can be easily computed *transversally* in many fault-tolerant schemes of quantum computing (e.g., the Shor code [106] or the [[7,1,3]] Steane code [109]). This means a Clifford gate on two logical qubits can be implemented by applying the same gate to corresponding pairs of physical qubits in each logical qubit. Since we expect any logical operation to touch all physical qubits, a transversal implementation for a gate is as inexpensive (in terms of gates, depth, etc.) as one can hope for. In other words, we think Clifford gates may be among the cheapest gates to implement on fault-tolerant quantum computers.

As already noted, Clifford gates are not universal, but adding literally *any* non-Clifford gate makes the gate set universal [87, 88]. As a side note, because of the mathematical elegance of the Clifford group, certain additional gates (such as $T := \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{smallmatrix}\right)$, or Toffoli) produce a universal gate set with a clean number theoretic description [9]. Number theoretic descriptions of unitary gates have led, in more than one case [71, 20, 51], to "exact synthesis" algorithms for compiling unitaries, which produce asymptotically better results than the Solovay-Kitaev algorithm [70]. Better compilation algorithms reduce the depth of the circuit, and minimizing the depth is a good idea for a variety of reasons discussed in the next two chapters (Chapter 5 and

[0] This chapter is based on work in [56], used with the permission of the authors.

[1] Readers are encouraged to refresh their understanding of the Clifford gates and tableaux in Chapter 2, since this chapter will, beyond this introduction, assume some familiarity with Clifford gates.

Chapter 6).

Yet another path to universality is to augment the Clifford gates with access to special quantum states, known (unironically) as *magic states* [28, 94, 106]. Traditional wisdom is that in this universal model of Clifford gates plus magic states, the cost of distilling the magic states will dominate the cost of a typical computation. In some papers, *only* magic states or non-Clifford gates count towards the cost of the computation, further supporting the idea that Clifford gates are cheap to implement. That said, there is some evidence in the other direction [77], so we may have to wait and see.

For all of the reasons above, Clifford gates are relevant to quantum devices now and in the future, and certainly worthy of study. This chapter will focus on a result which *classifies* Clifford gates by the set of unitaries they can construct. That is, for any set of Clifford gates whatsoever, we have a procedure to decide whether they generate *all* Clifford gates or just a subset. We find that there are finitely many sets of Clifford gates one can generate, and catalog all of them within this chapter.

4.1 Why Classify Clifford Gates?

A common thread throughout quantum computing is the manner in which a few elementary gates often suffice for universal computation. This "pervasiveness of universality" is explored in recent work of Aaronson, Grier, and Schaeffer [5]. There, the authors give a complete classification of *classical* reversible gates in terms of the functions over bits they generate and find that a rich structure emerges.

Of course, the ultimate goal would be a complete classification of *quantum* gate sets based on the functions over qubits they generate. Unfortunately, not even a full classification of the subgroups of a three-qubit system is known.[2] Since each class of gates is a subgroup, this suggests that a complete classification remains out of reach. This might be surprising given how well we understand random gate sets, and even those that contain particular gates such as CNOT [12, 103]. However, a full classification begets a complete understanding of *all* possible behaviors, despite their strangeness or rarity (see, for example, the sporadic gate sets in the lattice of Aaronson et al. [5]). Nevertheless, there has been some recent and encouraging progress on *some* classification problems, in particular, on classifying Hamiltonians (which can be applied for any period of time), rather than discrete gate sets. For instance, Bouland, Mančinska, and Zhang [24] recently classified all 2-qubit commuting Hamiltonians, while Childs et al. [31] characterized all 2-qubit Hamiltonians when restricted to circuits over two qubits. Additionally, Aaronson and Bouland [3] completed a classification for linear optics of 2-mode beamsplitters, which relied heavily on the characterization [57] of the finite subgroups of SU(3), underscoring the difficulty of quantum gate classification.

In this chapter, we contribute to the general classification problem by completing

[2] The difficulty in classifying the subgroups of $\mathrm{SU}(N)$ arises not from the infinite classes but from the finite ones. In fact, even the finite subgroups of SU(5) remain unclassified. This motivates our focus on finite, discrete classes such as the Clifford group.

the case of Clifford gates. Our model is motivated in part by the use of Clifford circuits as subroutines of a general quantum computation, much like the transversal gates in a fault-tolerant scheme. We regard the creation of arbitrary quantum states (e.g., magic states) as an inherently difficult task, and therefore require that any ancillary state used during the computation must be returned to its initial state at the end of the computation. These ancilla states are very similar to the "reusable magic states" of Anderson [10]. In fact, our result closes many of the open problems left by that paper.

4.1.1 Results

We present a full inclusion lattice of the 57 different classes of Clifford gates. Refer to Figure 4-2 for a diagram of the 30 single-qubit classes and Figure 4-3 for the remaining lattice of multi-qubit gates. Notation for the gates can be found in Chapter 2. We list some highlights of the classification below:

(1) **Conserved Quantities.** The invariants that dictate the structure of the lattice are most easily described by appealing to the tableau representation of a Clifford circuit (see Section 4.2). Nevertheless, there are some common themes that persist throughout classification. One major theme is that of basis preservation. We say that a gate "preserves" a basis if it maps elements of that basis to elements of that basis. For instance, the CNOT gate preserves the X and Z bases, but not the Y basis. There are classes which correspond to all 8 possible ways to preserve the X, Y, and Z bases. There are also classes in which each entangling operation only changes the sign of a certain basis, such as the CSIGN gate, or classes in which each entangling operation acts like an orthogonal transformation over the computational basis states, such as the T_4 gate (see Section 2.2.6). Finally, a class can be constrained by its ability to rotate a single qubit. See Section 4.4 for a more formal exposition.

(2) **Finite Generation.** Every class can be generated by a single gate on at most four qubits. In fact, given a set of gates generating some class, there always exist three gates from that set that generate the same class. Moreover, the classification implies that the canonical set of Clifford generators—CNOT, Hadamard, and phase—is *not* a minimal set of generators in our model. Indeed, it turns out that with the aid of ancilla qubits, CNOT and Hadamard generate a phase gate. This is well-known [8, 66], but comes as a simple consequence of our classification theorem.

(3) **Ancillas.** More generally, giving a Clifford gate access to ancillary qubits often increases the sorts of functions it can compute. A priori, one might suspect that extracting all functionality from a large entangling Clifford gate would require large highly-entangled ancilla states. Nevertheless, our classification shows that only a constant number of one and two-qubit ancillary states are ever needed. In fact, an even stronger result is true. Namely, our classification holds even when

we allow the ancillas to change in an input-independent manner,[3] as would be natural for a Clifford subroutine applied unconditionally in a quantum circuit. See Section 4.3 for further discussion.

(4) **Algorithms.** Our classification implies a linear time algorithm which, given the tableau of a gate G, decides which class G belongs to. As a consequence, we can identify the class for a list of tableaux (by running the algorithm for each one, then looking up the least upper bound on our lattice diagram), or determine whether one gate generates another (by running the algorithm for both, and checking whether the one class contains the other). In fact, to witness that G generates some class $\mathcal{C}$ (or higher) in the classification, one only needs to view a constant number of bits of the tableau. These details are discussed in Section 4.7 and Section 4.8.

(5) **Enumeration.** For each class $\mathcal{C}$ and for all n, we give explicit formulas for the number of gates in $\mathcal{C}$ on n-qubits. The enumeration usually leads to efficient canonical forms for circuits in the various classes. In fact, every class is exponentially smaller than any class strictly containing it. See Section 4.9.1 for details.

(6) **Sporadic Gates.** There are five classes containing the T_4 gate, four of which require some generator on at least 4 qubits. Nevertheless, the smallest generator for the class $\langle \mathrm{T}_4, \Gamma, \mathcal{P} \rangle$ is a gate on three qubits. We investigate such a gate in Section 4.9.3. Interestingly, there are no affine gate sets in the classification of *classical* reversible gates (i.e., in [5]) which admit a generator over three bits and no smaller.

4.1.2 Proof Outline

We can divide the proof into a few major steps. First, we introduce the notion of a Clifford tableau (or rather, recall it from Chapter 2), a binary matrix representation of the function computed by a Clifford circuit. We then "guess" all the classes in the classification and designate them by their generators. An examination of the tableaux of the gates in these classes reveals candidate invariants based on the elements of those tableaux. We then prove that these candidate invariants are indeed invariant under the rules of our model of computation. That is to say, if we have two gates whose tableaux satisfy the invariant, then the tableau of their composition satisfies the invariant, and so on for all the other ways to build circuits from gates—tensor products, ancillas, swapping.

At this point, we will have shown that each class has a corresponding invariant, from which the distinctness of the classes in the lattice is immediate. That is, for every pair of distinct classes there is a generator of one that fails to satisfy the invariant of the other. Next, we will show that this correspondence is bijective; that is, for each

[3] Aaronson et al. called this the "Loose Ancilla Rule," and it *does* affect their classification of classical reversible circuits.

tableau invariant there is a corresponding class whose generators generate all gates satisfying that tableau invariant. This will have the side effect of producing canonical forms for most of the classes in the lattice.

The challenge remains then to show that our list of classes is exhaustive. Suppose we are given some gate set G. Clearly, the class generated by G is contained in some class in the lattice. Let $\mathcal{C}$ be the smallest class containing all the gates in G. The hope is to show that G generates all of $\mathcal{C}$. To do this, we use the minimality of $\mathcal{C}$. That is, for each class $\mathcal{S} \subset \mathcal{C}$, there is some gate $g \in G$ which is not in $\mathcal{S}$, otherwise $\mathcal{S}$ would be a smaller class containing G. We now wish to use g to generate a simpler gate, also violating some invariant of $\mathcal{S}$. This is accomplished via the "universal construction" on the gate g, which is a particular circuit built from g and SWAP gates. Finally, we combine the simpler gates to construct the canonical generators for the class $\mathcal{C}$ itself.

4.2 Tableaux

Observe that the matrices $\mathtt{I}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}$ are linearly independent, and therefore form a basis for the 2×2 complex matrices. It follows that $\mathcal{P}_n$ spans all $2^n \times 2^n$ complex matrices. Hence, any unitary operation on n qubits can be characterized by its action on $\mathcal{P}_n$. In particular, any gate is characterized by how it acts on

$$p_1 = \mathtt{XI}\cdots\mathtt{I}, p_2 = \mathtt{ZI}\cdots\mathtt{I}, \ldots, p_{2n-1} = \mathtt{I}\cdots\mathtt{IX}, p_{2n} = \mathtt{I}\cdots\mathtt{IZ}.$$

We call this list the *Pauli basis on n qubits*, since one can write any element of $\mathcal{P}_n$ as a product of basis elements times a phase (± 1 or $\pm i$).

Now suppose we are given a Clifford gate, $U \in \mathcal{C}_n$. By definition, Clifford gates map each Pauli basis element to something in $\mathcal{P}_n$, which can be written as a product of basis elements times a phase. That is,

$$U p_j U^\dagger = \alpha_j \prod_{k=1}^{2n} p_k^{M_{jk}}$$

for some bits $M_{j1}, \ldots, M_{j(2n)} \in \{0,1\}$ and some phase[4] $\alpha_j \in \{\pm 1, \pm i\}$. The *tableau* for U is a succinct representation for U consisting of the binary matrix $M = [M_{jk}]$, and some representation of the phases $\alpha_1, \ldots, \alpha_{2n}$.

It turns out that U maps p_j to ± 1 times a Pauli string, never $\pm i$ times a Pauli string. This follows from the fact that the square of any p_j is $\mathtt{I}\cdots\mathtt{I}$, so

$$(U p_j U^\dagger)^2 = U p_j^2 U^\dagger = U(\mathtt{I}\cdots\mathtt{I})U^\dagger = +\mathtt{I}\cdots\mathtt{I}.$$

If the phase in front of the Pauli string in $U p_j U^\dagger$ were $\pm i$, then squaring it would produce a negative phase. Therefore, $U p_j U^\dagger$ is ± 1 times a Pauli string.

Unfortunately, α_j may still be any one of $\{\pm 1, \pm i\}$. This is because the product of $p_{2k-1} p_{2k}$ is $\mathtt{I}\cdots\mathtt{I}(\mathtt{XZ})\mathtt{I}\cdots\mathtt{I} = -i\mathtt{I}\cdots\mathtt{IYI}\cdots\mathtt{I}$, with an awkward $-i$ phase. Once we cancel the extra factors of i from α_j, we are left with

$$(-1)^{v_j} := \alpha_j \prod_{k=1}^{n} (-i)^{M_{j(2k-1)} M_{j(2k)}},$$

where $v_j \in \{0,1\}$ is the *phase bit for row j*. For example, if $U p_1 U^\dagger$ is $\mathtt{YI}\cdots\mathtt{I}$ then we have $M_{11} = M_{12} = 1$ and $v_1 = 0$. Then the complete tableau for U is the matrix $M = [M_{jk}]$ and the vector of bits $v = [v_j]$, which we typically write as

$$\left(\begin{array}{ccc|c} M_{11} & \cdots & M_{1(2n)} & v_1 \\ \vdots & \ddots & \vdots & \vdots \\ M_{(2n)1} & \cdots & M_{(2n)(2n)} & v_{2n} \end{array}\right).$$

[4] Note that the order of the terms in the product matters, since the Pauli basis elements do not necessarily commute, so we assume the terms are in the natural order from $p_1^{M_{j1}}$ up to $p_{2n}^{M_{j(2n)}}$.

Our ordering of the basis elements (which differs from other presentations) puts Pauli strings on the same qubit (e.g., $XI \cdots I$ and $ZI \cdots I$) side-by-side in the matrix, so the 2×2 submatrix

$$\begin{pmatrix} M_{2i-1,2j-1} & M_{2i-1,2j} \\ M_{2i,2j-1} & M_{2i,2j} \end{pmatrix}$$

completely describes how the ith qubit of the input affects the jth qubit of the output. In fact, it will be fruitful to think of the tableau as an $n \times n$ matrix of 2×2 blocks, along with a vector of $2n$ *phase bits*. To be clear, the blocks come from $\mathsf{R} := \mathbb{F}_2^{2\times 2}$, the ring of 2×2 matrices over the field of two elements, $\mathbb{F}_2$. Then the tableau is a matrix in $\mathsf{R}^{n\times n}$ (the $n \times n$ matrices over the ring R), combined with a vector of phase bits in $\mathbb{F}_2^{2n}$. Each row of the matrix is associated with a pair of phase bits from the vector.

However, not every matrix in $\mathsf{R}^{n\times n}$ corresponds to a Clifford circuit due to unitarity constraints. To best express these constraints, we define a unary operation * on R such that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^* = \begin{pmatrix} d & b \\ c & a \end{pmatrix}.$$

The * operator has the property that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix}^* = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} d & b \\ c & a \end{pmatrix} = \begin{pmatrix} ad+bc & 0 \\ 0 & ad+bc \end{pmatrix} = I\begin{vmatrix} a & b \\ c & d \end{vmatrix}.$$

Additionally,

$$\begin{aligned} I^* &= I, \\ (M+N)^* &= M^* + N^*, \\ (MN)^* &= N^* M^*, \\ (M^*)^* &= M, \end{aligned}$$

so * makes R a **-ring* or *ring with involution*. We also extend * to an operation on matrices (over R) which applies * to each entry and then transposes the matrix. It turns out that a tableau represents a unitary operation if and only if the matrix $M \in \mathsf{R}^{n\times n}$ satisfies $MM^* = M^*M = I$. This (intentionally) resembles the definition of a unitary matrix ($UU^\dagger = U^\dagger U = I$), we will call this the *unitarity condition*, and it corresponds to the unitarity of U as a gate, but M is certainly not a traditional complex unitary matrix (nor unitary over some finite field with conjugation).

4.2.1 Correspondence between Gates and Tableaux

We will find it useful to switch between gates and tableaux, as one notion often informs the other. In light of that, define $\mathcal{T}(g)$ to be the tableau of a Clifford gate g. In fact, we will use $\mathcal{M}(g)$ to denote only the matrix part of the tableau of g, as the phase bits of the tableau often prove to be irrelevant. Indeed, most non-degenerate gate sets generate the Pauli group, which alone suffices to set the phase bits of the tableau arbitrarily as follows: applying X to qubit j negates v_{2j} and applying Z to qubit

j negates v_{2j-1}. Furthermore, there is a surprising connection between individual entries of tableaux and elementary Clifford operations that can be extracted from them (see Section 4.7).

If $a \in \mathsf{R}$ is invertible, then let $\mathcal{G}(a)$ be the single-qubit gate with $\mathcal{M}(\mathcal{G}(a)) = a$ and zeros for phase bits. These are the gates in the first row of Table 4.1. Let $\mathcal{G}(a, i)$ be the gate $\mathcal{G}(a)$ applied to the ith qubit.

	$\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right)$
$\left(\begin{smallmatrix}0\\0\end{smallmatrix}\right)$	$\mathtt{I}$	$\mathrm{R}_X^\dagger$	θ_{X+Z}	R_Z	Γ_{+++}	Γ_{---}
$\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$	$\mathtt{X}$	R_X	$\mathrm{R}_Y^\dagger$	θ_{X+Y}	Γ_{--+}	Γ_{+-+}
$\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$	$\mathtt{Z}$	θ_{Y+Z}	R_Y	$\mathrm{R}_Z^\dagger$	Γ_{-+-}	Γ_{-++}
$\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$	$\mathtt{Y}$	θ_{Y-Z}	θ_{X-Z}	θ_{X-Y}	Γ_{+--}	Γ_{++-}

Table 4.1: Invertible tableau elements and the corresponding single-qubit gates produced by the universal construction. Row of the table corresponds to the sign bit of the row of the tableau in which the element occurs.

If $b \in \mathsf{R}$ is noninvertible, define $\mathrm{CNOT}(b, i, j)$ to be the generalized CNOT on qubits i and j corresponding to the singular matrix b. These are the gates in the first row of Table 4.2. The tableau $\mathcal{T}(\mathrm{CNOT}(b, i, j))$ is the identity tableau except for b^* and b in positions (i, j) and (j, i), respectively. We use the circuit in Figure 4-1 to designate such a gate.

Elt.	$\left(\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)$ / $\left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right)$ / $\left(\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&1\\0&0\end{smallmatrix}\right)$ / $\left(\begin{smallmatrix}0&1\\0&1\end{smallmatrix}\right)$
Gate	$C(\mathtt{X},\mathtt{X})$	$C(\mathtt{Y},\mathtt{Y})$	$C(\mathtt{Z},\mathtt{Z})$	$C(\mathtt{X},\mathtt{Z})$	$C(\mathtt{Y},\mathtt{X})$	$C(\mathtt{Z},\mathtt{Y})$

Table 4.2: Noninvertible tableau elements and the corresponding generalized CNOT gates produced by the universal construction.

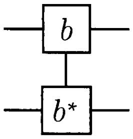


Figure 4-1: Circuit diagram for $\mathrm{CNOT}(b, 1, 2)$ gate.

Finally, we would like to have a direct way to compose two circuits by a simple operation on their tableaux. Suppose we wish to compute the composition of circuits C_1 and C_2. To compute $\mathcal{T}(C_2 \circ C_1)$, we must compute, for each Pauli basis element p_j, the product $C_2 C_1 p_j C_1^\dagger C_2^\dagger$. First consider the jth row of $\mathcal{T}(C_1)$, which gives

$$C_1 p_j C_1^\dagger = \alpha_j \prod_{k=1}^{2n} p_k^{M_{jk}^{(1)}},$$

where $M^{(1)}$ is the binary representation of $\mathcal{M}(C_1)$, and α_j is the phase. Similarly,

$$C_2 p_j C_2^\dagger = \beta_j \prod_{k=1}^{2n} p_k^{M_{jk}^{(2)}} .$$

Therefore,

$$\begin{aligned} C_2 C_1 p_j C_1^\dagger C_2^\dagger &= C_2 \left(\alpha_j \prod_{k=1}^{2n} p_k^{M_{jk}^{(1)}} \right) C_2^\dagger = \alpha_j \prod_{k=1}^{2n} \left(C_2 p_k^{M_{jk}^{(1)}} C_2^\dagger \right) = \alpha_j \prod_{k=1}^{2n} \left(\beta_k \prod_{\ell=1}^{2n} p_\ell^{M_{k\ell}^{(2)}} \right)^{M_{jk}^{(1)}} \\ &= \alpha_j \prod_{k=1}^{2n} \left(\beta_k^{M_{jk}^{(1)}} \prod_{\ell=1}^{2n} p_\ell^{M_{jk}^{(1)} M_{k\ell}^{(2)}} \right) \propto \prod_{\ell=1}^{2n} \prod_{k=1}^{2n} p_\ell^{M_{jk}^{(1)} M_{k\ell}^{(2)}} \propto \prod_{\ell=1}^{2n} p_\ell^{\oplus_{k=1}^{2n} M_{jk}^{(1)} M_{k\ell}^{(2)}} \\ &= \prod_{\ell=1}^{2n} p_\ell^{[M^{(1)} M^{(2)}]_{j\ell}} \end{aligned}$$

Notice that this implies $\mathcal{M}(C_2 \circ C_1) = \mathcal{M}(C_1)\mathcal{M}(C_2)$. Since it is cumbersome to write out explicitly, we did not include the exact phases in the above calculation. Nevertheless, one can compute the phase bits by tracking the intermediate steps in the above calculation, which includes the multiplication of Pauli basis elements.

4.3 Classes

Informally, the class generated by a set of Clifford gates is the collection of Clifford operations which can be constructed from those gates. Our goal is to determine the set of classes and the relationships between them. We now must lay out the precise operations we use to build circuits, such as composition and tensor product. Let us also introduce and justify our rules concerning the reordering of qubits and ancillary workspace.

First, we will always allow the swapping of qubits. This allows us to consider gates without needing to specify the qubits on which they must be applied. Indeed, we can relabel the input wires however we like. Secondly, we allow the use of quantum ancillas, that is, arbitrary states that we can "hardwire" into the computation. We only stipulate that these ancilla qubits be returned to their initial configuration at the end of the computation.

These ancilla inputs can be viewed as the workspace of the computation. If do not allow ancillary qubits, then a gate cannot be used to generate a smaller gate because there are not enough inputs to use it at all. Furthermore, if we want to apply a Clifford function as a subroutine of a general quantum computation, then we need that the ancilla states do not depend on the input. Otherwise, they would destroy the quantum coherence of the computation.

Finally, we turn to the question of the ancillas themselves. The weakest assumption one could make is that the ancillas are initialized to an unknown state, which the circuit may only change temporarily. This is somewhat artificial since we can, at the very least, initialize the workspace to the all-zeros state. Other classifications suggest that without this assumption the problem becomes *dramatically* more difficult.[5] A slightly stronger assumption would be to allow ancillas initialized to computational basis states, but this would break symmetry by introducing a bias towards the Z-basis.[6]

A next natural step would be to permit ancillas initialized to arbitrary stabilizer states. Although this would appear to be circular (i.e., Clifford gates are necessary to implement stabilizer states, which we then use as ancillas in Clifford circuits), the reusability of the ancilla states implies that even if the states are difficult to construct, at least we only have to construct them once. Unfortunately, we are unable to complete the classification in its entirety under this ancilla model. However, we have reduced the problem to finding a single stabilizer state which is stabilized by Γ and a permutation, and we conjecture that such a state exists (see Section 4.10). Moreover, the conjectured classification matches the one we will present (under a stronger ancilla model).

Finally, we arrive at our chosen model; that is, ancillas initialized to arbitrary quantum states. A priori, these states could be arbitrarily large, and arbitrarily

[5] For example, the lattice of classical many-to-one functions over bits is finite when we allow 0/1 inputs to any function, but infinite when we do not allow such freedom [90, 5].

[6] We can fix the bias by allowing all single-qubit ancillary states: $|0\rangle, |1\rangle, |+\rangle, |-1\rangle, |i\rangle, |-i\rangle$. This introduces new classes such as $\langle \theta_{X+Z} \otimes \theta_{X+Z} \rangle$, but we leave the classification under these assumptions as an open problem.

complicated to construct, which is clearly undesirable. It turns out, however, the classification only requires finitely many one or two qubit states, in particular, the eigenstates of the single-qubit Clifford gates and states that are locally equivalent to the Bell state. For comparison, we have determined the reversible gate lattice under quantum ancillas in Section 4.9.2, and observe that, in some cases, arbitrarily large, entangled states are necessary.

It is worth noting that there is a long line of work showing that weak gate sets, including Clifford gates, are universal for quantum computation when given access to magic states [28, 94, 106]. Importantly, these magic states do *not* need to be preserved after the computation. Conversely, under our model, we show that arbitrary quantum ancillas *cannot* boost the power of Clifford gates beyond the Clifford group.

One might assume that we could increase the power of ancillas by letting them change over the course of the computation, as long as the change is independent of the input (that is, from some constant initial state to some possibly different constant final state). Indeed, in the classification of reversible gates [5], these "loose ancillas" collapse a few pairs of classes. Nevertheless, we show that even with loose ancillas, our classification (as presented) still holds. When we formally define class invariants in Section 4.4 (see Theorem 32 in particular), we will see that the invariants hold under the loose quantum ancilla model, and therefore hold for all weaker models.

Now let us formally define a *class* $\mathcal{C}$ to be a set of Clifford gates satisfying the following four rules:

1. **Composition Rule** $\mathcal{C}$ is closed under composition of gates. If $f, g \in \mathcal{C}$, then $f \circ g \in \mathcal{C}$.

2. **Tensor Rule** $\mathcal{C}$ is closed under tensor product of gates. If $f, g \in \mathcal{C}$, then $f \otimes g \in \mathcal{C}$.

3. **Swap Rule** $\mathcal{C}$ contains the SWAP gate.

4. **Ancilla Rule** $\mathcal{C}$ is closed under ancillas. If $f \in \mathcal{C}$ and there exists g such that

$$f(|x\rangle \otimes |\psi\rangle) = g(|x\rangle) \otimes |\psi\rangle$$

for some $|\psi\rangle$ and for all inputs $|x\rangle$ (up to a global phase), then $g \in \mathcal{C}$.

Let's first see a few simple consequences of the model.

Proposition 30. *Let $\mathcal{C}$ be a class of Clifford gates. Then $\mathcal{C}$ contains the n qubit identity gate for any n.*

Proof. First, $\mathcal{C}$ contains SWAP. It follows that $\mathcal{C}$ contains the two qubit identity gate since it is the composition SWAP $\circ$ SWAP. By the ancilla rule, we can remove a qubit from the two qubit identity using *any* one-qubit state. Hence, the one qubit identity gate is in $\mathcal{C}$. Finally, $\mathcal{C}$ must contain the n-qubit identity gate because it is the tensor product of n one-qubit gates. □

Proposition 31. *Let $\mathcal{C}$ be a class of Clifford gates. For any $g \in \mathcal{C}$, the inverse, g^{-1}, belongs to $\mathcal{C}$.*

Proof. Consider the sequence

$$g, g \circ g, g \circ g \circ g, \ldots, g^n, \ldots.$$

Since there are finitely many Clifford gates on n qubits (certainly finitely many tableaux, and one gate per tableau), the sequence must eventually repeat. That is, $g^i = g^j$ for some $1 \leq i < j$. Since every Clifford gate has an inverse, we conclude that $1 = g^0 = g^{j-i}$, and hence $g^{-1} = g^{j-i-1}$. In other words, g^{-1} is a Clifford gate, and g^{-1} is a (positive) power of g and therefore in $\mathcal{C}$. □

The most practical way to talk about a class $\mathcal{C}$ is by its *generators*. We say a set of gates G *generates* a class $\mathcal{C}$ if $G \subseteq \mathcal{C}$ and every class containing G also contains $\mathcal{C}$. We introduce the notation $\langle \cdot \rangle$ for the class generated by a set of gates. Similarly, we say that G generates a specific gate g if $g \in \langle G \rangle$.

Our goal is therefore to identify all Clifford gate classes, determine their generators, and diagram the relationships between classes. As it turns out, there are 57 different classes, which we have split across Figure 4-2 (which contains the classes with single-qubit generators) and Figure 4-3 (which contains the multi-qubit classes). Each class is labeled by a set of generators for that class, except for ALL, the class of all Clifford gates; $\top$, the class of all single-qubit Clifford gates; and $\perp$, the class generated by the empty set. Additionally, we abbreviate some class names in Figure 4-2:

- θ_{+++}, θ_{+--}, θ_{-+-}, θ_{--+} denote the single-qubit classes containing Γ_{+++}, Γ_{+--}, Γ_{-+-}, and Γ_{--+} respectively, and three θ gates each, as indicated by the gray lines.
- θ_{xy} abbreviates θ_{x+y} or θ_{x-y} (it contains both) and similarly for θ_{xz} and θ_{yz}.

Some of the lines in Figure 4-2 are gray and dotted, not for any technical reason, but because the lattice would be unreadable otherwise.

In Figure 4-3 each class includes the label of the single-qubit subgroup, even when not all of the single-qubit generators are necessary to generate the class. This is intended to make the relationship between the degenerate and non-degenerate lattices clearer. For example, T_4 generates the Pauli group, $\mathcal{P}$, on its own (Lemma 40), but we label the class $\langle \mathrm{T}_4, \mathcal{P} \rangle$ to make it clear that the class $\langle \mathrm{T}_4 \rangle$ is above $\langle \mathcal{P} \rangle$ in the lattice.

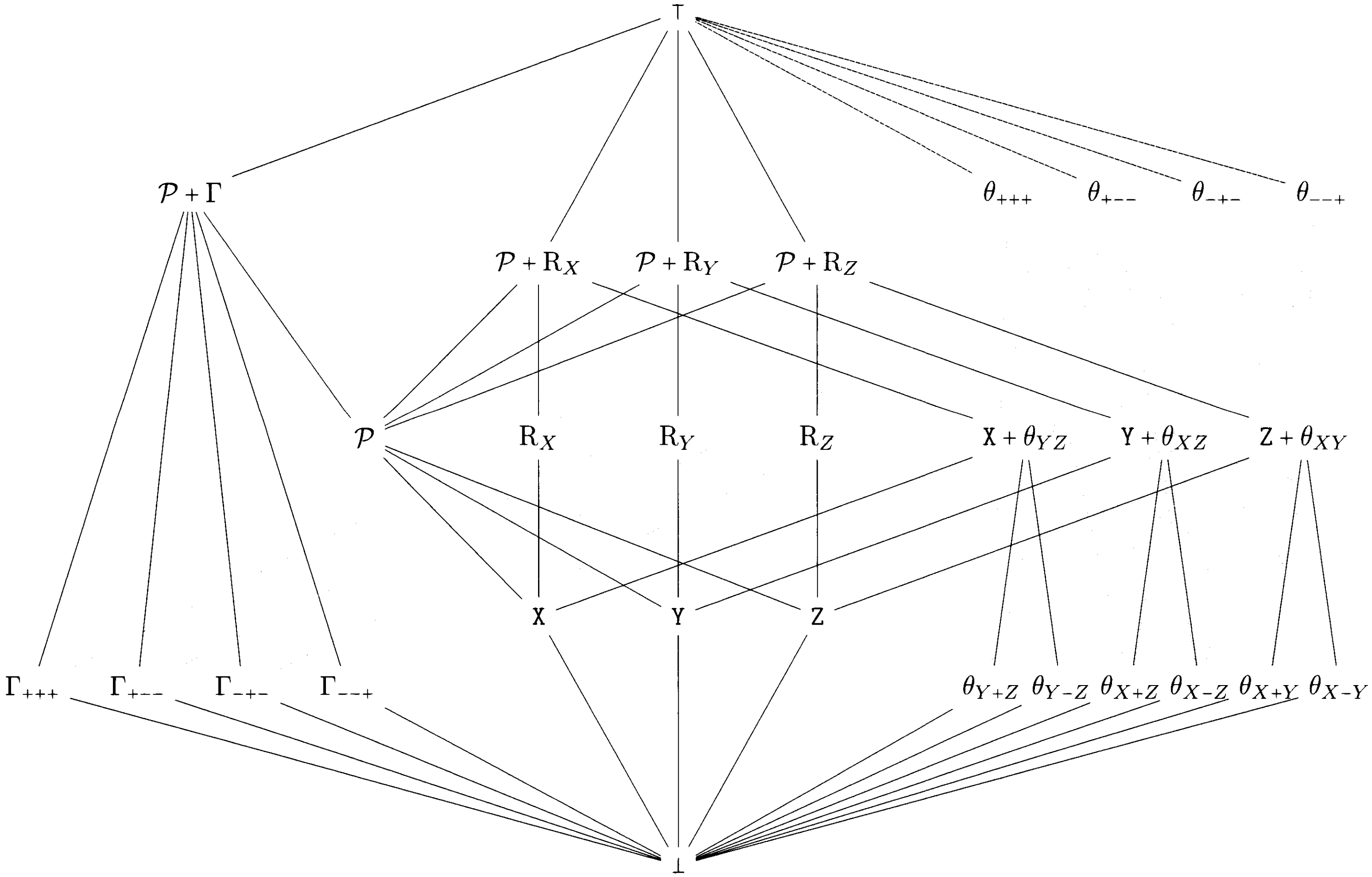


Figure 4-2: The inclusion lattice of degenerate Clifford gate classes

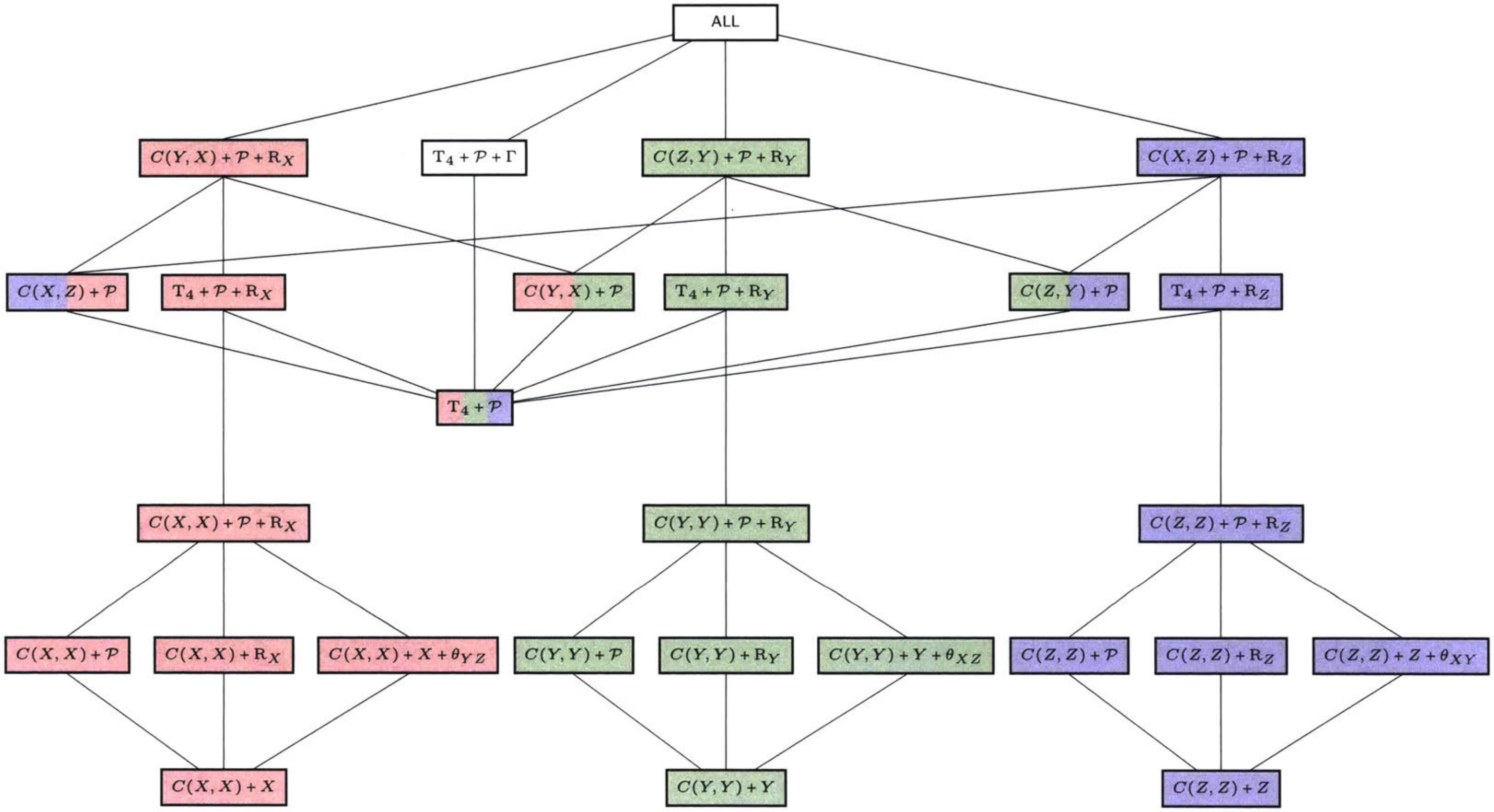

ALL
$\mathcal{C}(Y,X)+\mathcal{P}+\mathrm{R}_X$
$\mathrm{T}_4+\mathcal{P}+\Gamma$
$\mathcal{C}(Z,Y)+\mathcal{P}+\mathrm{R}_Y$
$\mathcal{C}(X,Z)+\mathcal{P}+\mathrm{R}_Z$
$\mathcal{C}(X,Z)+\mathcal{P}$
$\mathrm{T}_4+\mathcal{P}+\mathrm{R}_X$
$\mathcal{C}(Y,X)+\mathcal{P}$
$\mathrm{T}_4+\mathcal{P}+\mathrm{R}_Y$
$\mathcal{C}(Z,Y)+\mathcal{P}$
$\mathrm{T}_4+\mathcal{P}+\mathrm{R}_Z$
$\mathrm{T}_4+\mathcal{P}$
$\mathcal{C}(X,X)+\mathcal{P}+\mathrm{R}_X$
$\mathcal{C}(Y,Y)+\mathcal{P}+\mathrm{R}_Y$
$\mathcal{C}(Z,Z)+\mathcal{P}+\mathrm{R}_Z$
$\mathcal{C}(X,X)+\mathcal{P}$
$\mathcal{C}(X,X)+\mathrm{R}_X$
$\mathcal{C}(X,X)+X+\theta_{YZ}$
$\mathcal{C}(Y,Y)+\mathcal{P}$
$\mathcal{C}(Y,Y)+\mathrm{R}_Y$
$\mathcal{C}(Y,Y)+Y+\theta_{XZ}$
$\mathcal{C}(Z,Z)+\mathcal{P}$
$\mathcal{C}(Z,Z)+\mathrm{R}_Z$
$\mathcal{C}(Z,Z)+Z+\theta_{XY}$
$\mathcal{C}(X,X)+X$
$\mathcal{C}(Y,Y)+Y$
$\mathcal{C}(Z,Z)+Z$

Figure 4-3: The inclusion lattice of non-degenerate Clifford gate classes. Red, green, blue denote X-, Y-, and Z-preserving, respectively.

4.4 Invariants

Until now, we have defined each class in terms of the generators for that class. It turns out that each class can also be characterized as the set of all gates satisfying a collection of invariants. Section 4.5 formalizes this equivalence. This section focuses on the form of the invariants themselves.

Informally, an *invariant* is a property of a gate (or gate set), readily apparent from the tableau, which is preserved by the circuit building operations. In other words, if a collection of gates all satisfy a particular invariant then any circuit constructed from those gates must also satisfy the invariant. All our invariants are formally defined from the tableaux, but for now we give the following informal descriptions to make the intuition for each invariant clear.

X-, Y-, or Z-preserving: We say a Clifford gate is *Z-preserving* if it maps Z-basis states to Z-basis states, possibly with a change of phase. The Z-preserving gates include all (classical) reversible gates (e.g., $\mathtt{X}$, CNOT, and T_4), gates which only manipulate the sign (e.g., R_Z and CSIGN), and combinations of the two.

Symmetrically, there are X-preserving gates mapping X-basis states to X-basis states, and Y-preserving gates gates mapping Y-basis states to Y-basis states. Our definitions of classes, gates, invariants, etc., are completely symmetric with respect to X, Y and Z basis, so if some gate or class is X-preserving (resp. Y-preserving or Z-preserving), then there must be a corresponding gate or class which is Y-preserving (resp. Z-preserving or X-preserving). We will often appeal to this symmetry to simplify proofs.

Note that a gate can be any combination of X-, Y-, and Z-preserving. For instance, T_4 is X-, Y-, and Z-preserving; CNOT is X-preserving and Z-preserving but not Y-preserving (similarly $C(\mathtt{X},\mathtt{Y})$ and $C(\mathtt{Y},\mathtt{Z})$ fail to be Z-preserving and X-preserving respectively); R_Z is Z-preserving only (similarly R_X is X-preserving and R_Y is Y-preserving); and Γ is not X-, Y-, or Z-preserving.

Egalitarian We say a gate is *egalitarian* if it is fixed (up to a Pauli operation on each qubit) by the aforementioned $X/Y/Z$ symmetry, that is, the symmetry arising from conjugating all qubits by Γ (which cycles $\mathtt{X}$ to $\mathtt{Y}$, $\mathtt{Y}$ to $\mathtt{Z}$, and $\mathtt{Z}$ to $\mathtt{X}$). In particular, this implies that if egalitarian operation U maps Pauli string P to $Q = UPU^\dagger$ under conjugation, then U maps $\Gamma P \Gamma^\dagger$ to

$$U\Gamma P\Gamma^\dagger U^\dagger \propto \Gamma U\Gamma^\dagger\Gamma P\Gamma^\dagger\Gamma U^\dagger\Gamma^\dagger = \Gamma UPU^\dagger\Gamma^\dagger = \Gamma Q\Gamma^\dagger.$$

Not only are Γ and the Pauli matrices themselves egalitarian, but so is T_4.

Degenerate: We say a gate is *degenerate* if each input affects only one output. More precisely, when applying the gate to a string of Paulis, changing one Pauli in the input will change exactly one Pauli in the output. All single-qubit gates are degenerate, and all degenerate gates can be composed of single-qubit gates and SWAP gates.

X-, Y-, or Z-degenerate: A gate is *Z-degenerate* if it is Z-preserving and flipping any bit of a classical (Z-basis) input to the gate causes exactly one bit of the output to flip. The gate may or may not affect the phase. This class includes several Z-preserving single qubit gates, like R_Z, the Pauli operations, and θ_{X+Y}. It also includes CSIGN because this gate *only* affects phase, but CNOT is not Z-degenerate because flipping the control bit changes both outputs. Notice that CSIGN is Z-degenerate, but not degenerate. We define X-degenerate and Y-degenerate symmetrically.

X-, Y-, or Z-orthogonal: A gate G is *Z-orthogonal* if it can be built from T_4 and Z-preserving single-qubit gates. The term "orthogonal" comes from the fact that T_4 is an orthogonal linear transformation in the Z-basis, but not all Z-orthogonal gates are literally orthogonal transformations in the Z-basis (see, for example, Lemma 40). Similarly for X-orthogonal and Y-orthogonal.

Single Qubit Gates: There are thirty different classes of single-qubit gates. All of these classes are degenerate, and some can be distinguished by the other invariants above. However, many single-qubit invariants depend on the phase bits of the tableau. For instance, the tableau of θ_{X+Y}, θ_{X-Y}, and R_Z all have the same matrix part, $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$, but generate three distinct classes. One can write down explicit invariants for these classes where the phase bits are correlated to the tableau entries, but in most cases we present a single-qubit class as a subgroup of the symmetries of the cube/octahedron, as shown in Figure 2-1.

4.4.1 Formal invariants

An *invariant* is a property of tableaux which is preserved by the four circuit-building rules.

Swap Rule: Every class contains the SWAP gate, so every invariant we propose must be satisfied by the tableau for SWAP.

Composition Rule: If the invariant holds for two gates, then it must hold for their composition. We have seen that the tableau for the composition of two gates is essentially the matrix product of the two tableau, except for the phase bits (which are significantly more complicated to update).

Tensor Rule: The tensor product of two gates satisfying the invariant must also satisfy the invariant. Note that the tableau of the tensor product is the direct sum of the tableaux, and phase bits are inherited from the sub-tableaux in the natural way.

Ancilla Rule: The invariant must be preserved when some qubits are used as ancillas. It turns out the ancilla operation reduces the tableau to a submatrix (of non-ancilla rows and columns) and under certain conditions, the corresponding subset of the phase bits. This is somewhat technical, so we prove it in Theorem 32 below.

Theorem 32. *Let G be a Clifford gate on n qubits, and suppose there exist states $|\psi\rangle$ and $|\psi'\rangle$ such that*

$$G(|x\rangle \otimes |\psi\rangle) = H(|x\rangle) \otimes |\psi'\rangle$$

for all $|x\rangle$, for some unitary H on m-qubits. In particular, this is true if we use the ancilla rule to reduce G to H, where $|\psi\rangle = |\psi'\rangle$ is the ancilla state. Then

1. *H is a Clifford operation,*
2. *$\mathcal{M}(H)$ is obtained by removing the rows and columns corresponding to the ancilla bits from $\mathcal{M}(G)$,*
3. *If every bit (in $\mathcal{M}(G)$ as a binary matrix) we remove from a row is zero, then the phase bit for that row is the same in $\mathcal{T}(G)$ and $\mathcal{T}(H)$.*

Proof. Let $P \in \mathcal{P}_m$. Then for all $|x\rangle$,

$$G(P|x\rangle \otimes |\psi\rangle) = H(P|x\rangle) \otimes |\psi'\rangle .$$

On the other hand, G is a Clifford gate, so conjugating the Pauli string $P \otimes I^{n-m}$ by G produces $Q \otimes R$ for $Q \in \mathcal{P}_m$ and $R \in \mathcal{P}_{n-m}$. Equivalently,

$$G(P \otimes \mathtt{I}^{\otimes(n-m)}) = (Q \otimes R)G.$$

It follows that

$$\begin{aligned} H(P|x\rangle) \otimes |\psi'\rangle &= G(P|x\rangle \otimes |\psi\rangle) \\ &= (Q \otimes R)G(|x\rangle \otimes |\psi\rangle) \\ &= (Q \otimes R)\left(H(|x\rangle) \otimes |\psi'\rangle\right) \\ &= QH(|x\rangle) \otimes R|\psi'\rangle , \end{aligned}$$

so up to phase, $H(P|x\rangle) = QH(|x\rangle)$ for all $|x\rangle$, and $|\psi'\rangle = R|\psi'\rangle$. The first equation implies $HP = QH$, or $HPH^\dagger = Q$. Since P was arbitrary, the conjugation of a Pauli string by H is always another Pauli string, so H is a Clifford gate.

In the special case that P (and therefore $P \otimes \mathtt{I}^{\otimes(n-m)}$) is a Pauli basis element, then $Q \otimes R$ is represented in row of the binary tableau of G. We keep the bits representing Q in the tableau for H, since $HPH^\dagger = Q$, which is why $\mathcal{M}(H)$ is a submatrix of $\mathcal{M}(G)$. Clearly the phase of Q is the same as the phase of $Q \otimes R$ if and only if R is positive. In the special case $R = \mathtt{I}^{\otimes(n-m)}$, it is easy to see that R is positive, otherwise

$$|\psi'\rangle = R|\psi'\rangle = -\mathtt{I}^{\otimes(n-m)}|\psi'\rangle = -|\psi'\rangle$$

is a contradiction. Hence, the phase for the corresponding row of $\mathcal{T}(H)$ is inherited from $\mathcal{T}(G)$. □

As a direct consequence of these rules, our invariants take on a distinctly algebraic flavor. Let us consider, for the sake of illustration, invariants that depend only on the matrix part of the tableau and ignore the phase bits. Then an invariant is equivalent to a set of matrices closed under the four rules above. In particular, the matrices to form a group under multiplication as a consequence of the composition rule (and the fact that every gate has finite order).

On the other hand, not every group of matrices will correspond to an invariant. For instance, due to the swap rule, the group of matrices must also be closed under arbitrary reordering the rows and columns. This eliminates, e.g., the group of upper triangular matrices. Similarly, the ancilla rule excludes the special orthogonal group. In the end, we are left with just two kinds of matrix groups which lead to invariants:

Subring Invariants Matrices with elements restricted to a particular subring of R (analogous to the real matrices, integer matrices, etc.)

Permutation Invariants Permutation matrices, except where each 1 entry can be any one of a subset of invertible elements, and each 0 entry comes from a collection of non-invertible elements.

Now we are ready to present formal definitions for these invariants, and show that they really are preserved by the circuit-building rules.

4.4.2 Subring invariants

The first kind of invariant restricts the entries of the tableau to a subring of R. That is, given a subring $\mathsf{S} \subseteq \mathsf{R}$, a gate satisfies the invariant $\mathcal{I}(\mathsf{S})$ if and only if all entries of the tableau are in S. There are twelve classes, all near the top of the lattice, of the form

$$\mathcal{C} = \{\text{All gates satisfying } \mathcal{I}(\mathsf{S})\},$$

corresponding to all 12 subrings of R listed below.

- The entire ring, R, is technically a subring of itself, and $\mathcal{I}(\mathsf{R})$ is the trivial invariant satisfied by all Clifford gates. Notice that not *every* matrix over R gives a valid tableau because it must still be unitary.

- There are four maximal proper subrings of R:

$$\begin{aligned}
\mathsf{R}_X &= \{\left(\begin{smallmatrix} a & 0 \\ c & d \end{smallmatrix}\right) : \{a, c, d\} \in \{0, 1\}\}, \\
\mathsf{R}_Y &= \{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) : \{a, b, c, d\} \in \{0, 1\}, a + b + c + d = 0\}, \\
\mathsf{R}_Z &= \{\left(\begin{smallmatrix} a & b \\ 0 & d \end{smallmatrix}\right) : \{a, b, d\} \in \{0, 1\}\}, \\
\mathsf{R}_E &= \{\left(\begin{smallmatrix} a & b \\ b & a+b \end{smallmatrix}\right) : \{a, b\} \in \{0, 1\}\}.
\end{aligned}$$

Our formal definition for Z-preserving gates is the invariant $\mathcal{I}(\mathsf{R}_Z)$. The fact that the lower left entry is 0 implies that the gate maps Pauli strings of $\mathtt{I}$ and $\mathtt{Z}$ to strings of $\mathtt{I}$ and $\mathtt{Z}$. Hence, Z-basis strings are mapped to Z-basis strings. Similarly, the X-preserving and Y-preserving invariants are $\mathcal{I}(\mathsf{R}_X)$ and $\mathcal{I}(\mathsf{R}_Y)$ respectively. The egalitarian invariant, $\mathcal{I}(\mathsf{R}_E)$, comes from the subring R_E.

- The intersection of two subrings is itself a subring, giving us exactly four more subrings ($\mathsf{R}_X \cap \mathsf{R}_Y$, $\mathsf{R}_X \cap \mathsf{R}_Z$, $\mathsf{R}_Y \cap \mathsf{R}_Z$, and $\mathsf{R}_X \cap \mathsf{R}_Y \cap \mathsf{R}_Z$) since the intersection of R_E with any of the others is

  $$\mathsf{R}_X \cap \mathsf{R}_Y \cap \mathsf{R}_Z = \{\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\},$$

  the trivial ring.

- Three more subrings are obtained by taking only self-conjugate elements of R_X, R_Y, and R_Z respectively. An element $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ is *self-conjugate* if

  $$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)^*,$$

  or equivalently, $a = d$. These invariants correspond to the X-orthogonal, Y-orthogonal, and Z-orthogonal classes $\langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_X \rangle$, $\langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Y \rangle$, and $\langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Z \rangle$ respectively.

Theorem 33. *For any subring* $\mathsf{S} \subseteq \mathsf{R}$, *the property* $\mathcal{I}(\mathsf{S})$ *is an invariant. That is, the set of matrices over* S *respect the circuit building operations.*

Proof. Every subring contains $\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ by definition, and therefore the tableau (phase bits omitted) of the SWAP gate,

$$\left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right)$$

satisfies $\mathcal{I}(\mathsf{S})$.

Matrix multiplication is a polynomial in the entries of the two matrices, so composition cannot produce entries outside the subring. Similarly, combining tableau with tensor products or reducing tableau to submatrices via ancillas does not introduce any new ring elements; those operations only use elements already present in the tableau. We conclude that $\mathcal{I}(\mathsf{S})$ is an invariant for any subring S. □

4.4.3 Permutation invariants

The *permutation invariants* get their name from the matrix part of their tableaux, which is required to have the structure of a permutation matrix. That is, every row (or column) has exactly one element which is invertible, and the others are non-invertible. Permutation invariants are also sensitive to phase bits. It is natural to associate the unique invertible element in a row with the phase bits for that row, giving the tableau of a single-qubit gate. A permutation invariant $\mathcal{P}(G, S)$ is defined by the set of single-qubit gates G which can be obtained in this way, and the set of non-invertible elements S used to fill the rest of the tableau. In other words, a tableau satisfies $\mathcal{P}(G, S)$ if all entries are from S except exactly one entry per row which, when combined with the phase bits for the row, is the tableau of some gate in G.

Note that not all pairs of sets (G, S) produce an invariant. For instance, circuit-building operations will fail to preserve $\mathcal{P}(G, S)$ if G is not a group. The exact relationship between G and S required to produce an invariant is difficult to write down. Roughly speaking, products of elements in S should be zero, products of elements in G should remain in G, and products between S and $\mathcal{M}(G)$ should be manageable in some sense. Theorem 34 gives a list of $\mathcal{P}(G, S)$ invariants, which will turn out to be exhaustive by Theorem 48, the culminating theorem of this chapter.

Theorem 34. *We prove that the following permutation invariants are indeed invariant under the circuit-building operations.*

1. *If G is a group of single-qubit gates then*

   $$\mathcal{P}(G, \{(\begin{smallmatrix}0&0\\0&0\end{smallmatrix})\})$$

   is an invariant for $\langle G\rangle$. All thirty degenerate classes are characterized by invariants of this form.

2. *If $\langle X\rangle \subseteq G \subseteq \langle \mathcal{P}, \mathrm{R}_X\rangle$ is a group of single-qubit gates then*

   $$\mathcal{P}(G, \{(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}), (\begin{smallmatrix}0&0\\1&0\end{smallmatrix})\})$$

   is an invariant for $\langle C(X, X), G\rangle$. These invariants characterize the five X-degenerate classes.

3. *If $\langle Y\rangle \subseteq G \subseteq \langle \mathcal{P}, \mathrm{R}_Y\rangle$ is a group of single-qubit gates then*

   $$\mathcal{P}(G, \{(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}), (\begin{smallmatrix}1&1\\1&1\end{smallmatrix})\})$$

   is an invariant for $\langle C(Y, Y), G\rangle$. These invariants characterize the five Y-degenerate classes.

4. *If $\langle Z\rangle \subseteq G \subseteq \langle \mathcal{P}, \mathrm{R}_Z\rangle$ is a group of single-qubit gates then*

   $$\mathcal{P}(G, \{(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}), (\begin{smallmatrix}0&1\\0&0\end{smallmatrix})\})$$

   is an invariant for $\langle C(Z, Z), G\rangle$. These invariants characterize the five Z-degenerate classes.

Proof. Let $\mathcal{P}(G, S)$ be one of the invariants above. Let $M = \{\mathcal{M}(g) : g \in G\}$ be the set of matrices from tableaux in G. In all cases, S contains $(\begin{smallmatrix}0&0\\0&0\end{smallmatrix})$, and G contains the single-qubit identity operation,

$$\left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right),$$

so SWAP satisfies the invariant. And clearly the direct sum of two tableaux in $\mathcal{P}(G, S)$ is still in $\mathcal{P}(G, S)$ for any G and S.

Now consider the composition of two gates. Each entry in the tableau is a dot product of some row from one tableau with some column from the other. Hence, the entry is a sum of $S \times S$, $S \times M$, $M \times S$, or $M \times M$ products. Observe that the $S \times S$ products are all zero (for the particular sets S above), so we may ignore those products. Recall that the row and column each contain exactly one entry in M, so depending on whether those entries align, we get either $SM + MS \subseteq S$ or $M^2 = M$. Furthermore, for any row in one tableau there is exactly one column in the other such that the invertible entries line up. Therefore, exactly one entry in any row (or column) of the composition is in M and the rest are in S. Clearly the matrix part of the tableau has the correct form for the invariant.

We must also consider phase bits under composition. Recall that the phase bits associate with the invertible entries of the matrix to produce single-qubit gates. When we multiply two tableau, these single-qubit gates multiply to produce elements in G (since G is a group), as you would expect. If the non-invertible elements are all zero, then this is the only factor in determining phase bits, so the invariant is preserved by composition.

Now consider the phase bits in the case where S contains nonzero elements, for instance,

$$S = \{\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)\}.$$

Notice that in this case, both matrices in S have zeros in the bottom row, and the invertible matrices are of the form $\left(\begin{smallmatrix} 1 & a \\ 0 & 1 \end{smallmatrix}\right)$. Hence, every even-indexed row of the tableau (as a binary matrix) is all zeros except for one entry. Using the method of tableau composition in Section 4.2, one can easily show that for these even-indexed rows, the phase bits are exactly what one would get by composing the invertible elements as gates in G. For the other half of the rows, the non-invertible elements may flip the phase bits. But we assume G contains the Pauli element (in this case Z) which flips that sign, so the invertible elements and associated phase bits are still in G, therefore the invariant is preserved. The X- and Y-degenerate cases are similar.

Last, we show that $\mathcal{P}(G, S)$ is preserved under ancilla operations. Recall that when we use ancillas, we remove the rows and columns corresponding to those bits. Clearly the elements of the submatrix are still in M and S. There is a risk that the invertible element for some row could be in one of the removed columns, but if the submatrix is missing an invertible element in some row then the submatrix is not unitary and the ancilla rule must have been misapplied. Hence, only elements in S are removed in the non-ancilla part of the tableau, and each row still contains exactly one entry in M.

We appeal to Theorem 32 for the phase bits. The theorem says that removing the ancillas can only change the sign for a row if there is a nonzero entry in the non-ancilla bits of the row that are removed. For example, if $S = \{\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)\}$ then only the top phase bit can change. But changing the top phase bit is the same as applying a Z, and for this case Z is assumed to be in G, so the combination of the element in M and the phase bits is still in G. Therefore the Z-degenerate $\mathcal{P}(G, S)$ are invariants, and the X-degenerate and Y-degenerate invariants follow by symmetry.

□

4.5 Equivalence of Generator and Invariant Definitions

We have now defined each class by a set of generators, and by an invariant, but have not yet shown that these definitions coincide. Below are a collection of lemmas which prove this for all classes in our lattice. Note that one direction is always trivial: it is easy to check that the generators defining a class satisfy a particular invariant, and therefore everything they generate (i.e., the class) must satisfy the invariant. We encourage the reader to check these invariants against, say, the tableaux in Table 2.2.

For the other inclusion (i.e., every gate satisfying the invariant can be generated by the given generators), we start with an arbitrary gate g satisfying the invariant, and apply gates in the class to g to simplify its tableau step-by-step until it is the tableau of the identity operation. It follows that $AgB = I$ for circuits A and B in the class, which proves $g = A^{-1}B^{-1}$ is in the class. In many cases, the circuit derived this way is a *canonical form* for the gate, and can be used to count the number of gates on n qubits in a class.

Let us start with the degenerate classes.

Lemma 35. *Let G be a group of single-qubit gates, and let g be a gate satisfying the permutation invariant $\mathcal{P}(G, \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right)\})$. Then there is a circuit for g consisting of a permutation of the inputs followed by layer of single-qubit gates in G.*

Proof. Consider the tableau for g. Each row or column has exactly one invertible element, so we can read off a permutation π from the positions of those elements. Apply SWAP gates to g to remove this permutation, and put the invertible elements on the diagonal. When we pair a diagonal element with the phase bits for that row, we get a single-qubit gate g_i in G. Applying the inverse of this gate to qubit i will zero the phase bits for that row, and make $\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$ the diagonal entry. Once we do this for each row, we have the identity tableau, therefore g is in $\langle G\rangle$ and has a circuit of the desired form. □

Next, we consider the Z-degenerate classes and, by symmetry, the X-degenerate and Y-degenerate classes.

Lemma 36. *Let $\langle \mathcal{Z}\rangle \subseteq G \subseteq \langle \mathcal{P}, \mathrm{R}_Z\rangle$ be a subgroup of Z-preserving single-qubit gates. Let g be any gate satisfying the permutation invariant $\mathcal{P}(G, \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)\})$. Then there is a circuit for g consisting of a layer of single-qubit gates (from G), a layer of $C(\mathtt{Z}, \mathtt{Z})$ gates, and a permutation.*

Proof. Consider the tableau of g. We can read off a permutation π, and a single-qubit gate for each input. Assume we have removed those gates (i.e., we now consider the tableau of $g\pi^{-1}g_1^{-1}\cdots g_n^{-1})$, so the tableau has $\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$ on the diagonal, all other entries are either $\left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)$ or zero, and the phase bits are zero.

The non-zero, off-diagonal entries in the matrix indicate the positions of $C(\mathtt{Z}, \mathtt{Z})$ gates. Specifically, if the entry in row i and column j is nonzero then there is a $C(\mathtt{Z}, \mathtt{Z})$ on qubits i and j. Note that because the matrix part of the tableau is unitary, the symmetric entry in row j and column i must also be non-zero. The remainder of the circuit consists of the set of $C(\mathtt{Z},\mathtt{Z})$ gates indicated by the non-zero, off-diagonal entries. Notice that $C(\mathtt{Z},\mathtt{Z})$ gates always commute, so their ordering does not matter.. □

Now let us consider four Z-preserving classes which, when we consider symmetry (i.e., the X-preserving and Y-preserving equivalents) cover all but two of the remaining classes.

Lemma 37. *For each of the four classes $\langle \mathrm{T}_4, \mathcal{P}\rangle$, $\langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Z\rangle$, $\langle C(\mathtt{Z},\mathtt{X}), \mathcal{P}\rangle$, and $\langle C(\mathtt{Z},\mathtt{X}), \mathcal{P}, \mathrm{R}_Z\rangle$ is the set of all gates corresponding to a subring invariant, where the subrings are (respectively)*

$$\begin{aligned}
\mathsf{S}_1 &= \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\},\\
\mathsf{S}_2 &= \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)\},\\
\mathsf{S}_3 &= \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\},\\
\mathsf{S}_4 &= \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}1&1\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)\}.
\end{aligned}$$

Proof. In all four classes, elements of the tableau are of the form $\left(\begin{smallmatrix}a&b\\0&d\end{smallmatrix}\right)$. Suppose $\left(\begin{smallmatrix}?&?\\0&d_1\end{smallmatrix}\right)$ is the ith entry of some row, and $\left(\begin{smallmatrix}?&?\\0&d_2\end{smallmatrix}\right)$ is the jth entry in the same row, where entries labeled by "?" are unconstrained. If we apply a CNOT gate from qubit i to j, these entries will be of the form $\left(\begin{smallmatrix}?&?\\0&d_1\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}?&?\\0&d_1+d_2\end{smallmatrix}\right)$ respectively. That is, the bottom right bits change as though we applied the CNOT gate to those bits. Since a T_4 gate can be built from CNOT gates, it will (similarly) affect the bottom right bits as though we are applying a T_4.

Our strategy is to use either CNOT or T_4 gates (depending on the class) to perform Gaussian elimination on the bottom right entries of the matrix elements. If we have access to CNOT gates then we literally apply Gaussian elimination, using CNOT to add one column to another, and using SWAP to exchange columns.

If we only have T_4 gates then we are in subring $\mathsf{S}_1 \subseteq \mathsf{S}_2$ or S_2, so $\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$ are the only elements with a 1 in the bottom right position, and also the only invertible elements. It follows that the number of bottom right bits set to 1 in a row is the same as the number of invertible elements, which must be odd because the matrix is unitary. To reduce the number, we apply a T_4 to three 1 bits and a 0 bit (note: we may add a zero bit by adding an ancilla, if necessary), which changes the 0 to a 1 and the 1's to 0's, reducing the number of 1's (or invertible elements) in the row by two. When there is a single 1 left in the row, unitarity conditions imply that it is also the only 1 left in that column, so we may ignore that row and column for the moment and continue to eliminate the rest of the matrix.

Now suppose we have row reduced the matrix, using either CNOT or T_4, so that the bottom right entry of every element is 0, except along the diagonal where that bit 1. At this point, the diagonal element is the only element in a row that can possibly be invertible, therefore the diagonal elements are of the form $\left(\begin{smallmatrix}1&b\\0&1\end{smallmatrix}\right)$. Similarly, unitarity conditions imply that the off-diagonal elements are of the form $\left(\begin{smallmatrix}0&b\\0&0\end{smallmatrix}\right)$. In other words, the remaining tableau is Z-degenerate, since there is only one invertible element per row or column, and the off-diagonal elements are in $I = \{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)\}$. We can use either Lemma 35 or Lemma 36 to find a circuit from the remainder, which is in either $\langle \mathcal{P} \rangle$ or $\langle C(\mathsf{Z},\mathsf{Z}), \mathcal{P}, \mathrm{R}_Z \rangle$, depending on the class. □

There are only two classes remaining, ALL and $\langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle$, which we handle specially. For the first, we appeal to Aaronson and Gottesman [4] who give an explicit decomposition for any Clifford gate into layers of CNOT, Hadamard (θ_{X+Z} in our notation), and phase (R_Z) gates.

Lemma 38. *Any egalitarian gate g can be constructed from T_4, $\mathcal{P}$ and Γ gates.*

Proof. Egalitarian gates satisfy the invariant that all elements are in the subring

$$\{\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right), \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right).\}$$

In fact, this subring is isomorphic to $\mathbb{F}_4$, so it is a field. The unitarity of the matrix in our sense translates to unitarity as a matrix over $\mathbb{F}_4$.

Like the other T_4 classes, we use Gaussian elimination on the tableau of g. Consider a row of the tableau. If the entry in some column is not the identity, then apply Γ or Γ^{-1} to the corresponding qubit to make it the identity. By unitarity, there are an odd number of identity elements in the row. We may remove pairs of identity elements with a T_4, similar to Lemma 37, until there is only one left and the rest of the row is zero. Unitarity implies the column below the identity element is also zero, and we proceed to eliminate the rest of the tableau. Once the matrix part of the tableau is the identity, we apply Pauli matrices to zero out the phase bits.

We conclude that all egalitarian gates are in $\langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle$. □

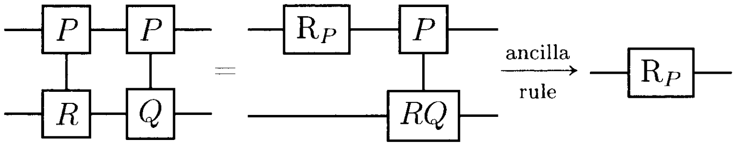
4.6 Circuit Identities

In this section, we give necessary tools to prove that a set of gates generates, in some sense, "all that one could hope for." Formally, we wish to prove that the gate set generates a particular class in the classification lattice when it is contained in that class but fails to satisfy the invariants of all classes below it. To this end, we give several useful circuit identities that will be used extensively in Section 4.7. For instance, one can show that any circuit on two qubits can be reduced to an equivalent circuit containing at most one generalized CNOT gate (see Section 4.9.4). The following lemma gives only the aspect of that theorem that is necessary to the classification, that is, the ability to extract single-qubit Clifford gates from the composition of generalized CNOT gates.

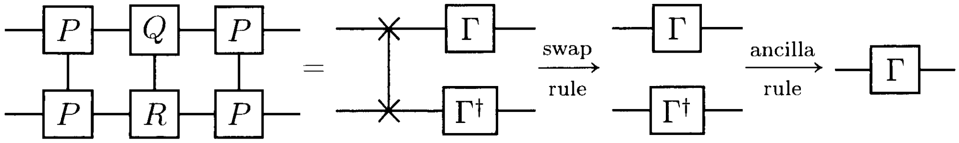
Lemma 39. *Let* $P, Q, R \in \mathcal{P}$*, and let* $\Gamma P \Gamma^\dagger = Q$ *and* $\Gamma Q \Gamma^\dagger = R$*. Then*

- $C(P,Q)$ *and* $C(P,R)$ *generate* R_P*.*
- $C(P,P)$ *and* $C(P,R)$ *generate* R_P*.*
- $C(P,P)$ *and* $C(Q,R)$ *generate* Γ*.*
- $C(P,P)$ *and* $C(Q,Q)$ *generate* θ_{P+Q}*.*

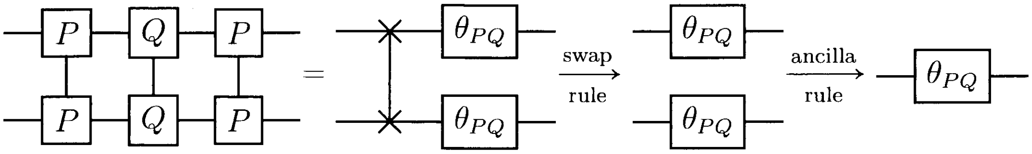
Proof. The first two inclusions come from the following identity, which holds whenever $\Gamma Q \Gamma^\dagger = R$ (i.e., regardless of P):



Similarly, for the third identity, we get



and for the final identity

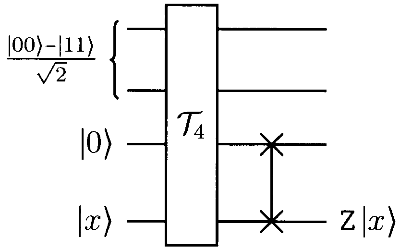


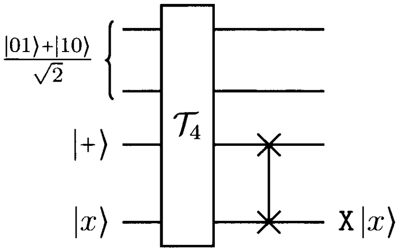
□

It might seem strange to reduce non-degenerate gates into less powerful single-qubit gates, but we will eventually see that single-qubit generators are the most crucial. Once we have shown that a particular set of gates generates all single-qubit operations, then that set of gates will generate the class of *all* Clifford operations provided it contains any non-degenerate gate. All non-degenerate gates generate at least one Pauli, often the entire Pauli group, which is why some single-qubit classes do not appear as the single-qubit subgroup of a non-degenerate class. For instance, consider the CNOT gate where the first qubit controls the second qubit. If we let the first input be $|1\rangle$, then a $\mathtt{X}$ operation is always applied to the second qubit. Similarly, if we let the input to the second qubit be $|-\rangle$, then a $\mathtt{Z}$ operation is always applied to the first qubit. Under the ancilla rule, we now have $\mathtt{X}$ and $\mathtt{Z}$ operations, so we can generate $\mathtt{Y}$ and the entire Pauli group. Clearly, the same is true for any heterogeneous CNOT gate. However, surprisingly, the following lemma shows that even the T_4 gate suffices to generate the entire Pauli group.

Lemma 40. T_4 *generates the Pauli group.*

Proof. Consider the following two circuits:





Under the ancilla rule, the first generates a Pauli Z operation while the second generates a Pauli X, from which we can clearly generate the Pauli group. □

There is another way to view the identity of Lemma 40 which will be useful later. Since T_4 is an affine gate over the computational basis states, $\mathrm{T}_4 = C_1 C_2 \dots C_n$ where each C_i is a CNOT gate. Furthermore, T_4 and CNOT are their own inverses, so $\mathrm{T}_4 = C_n C_{n-1} \dots C_1$. Finally, because T_4 is symmetric when represented as a 4×4 matrix over $\mathbb{F}_2$, $\mathrm{T}_4 = C_1^T C_2^T \dots C_n^T$. Notice that C_i^T just represents the CNOT gate C_i where the control and target qubits are swapped. Therefore, leveraging the well-known equivalence $\theta_{X+Z}^{\otimes 2} \circ \mathrm{CNOT} \circ \theta_{X+Z}^{\otimes 2} = \mathrm{SWAP} \circ \mathrm{CNOT} \circ \mathrm{SWAP}$ we arrive at the following consequence:[7]

$$\theta_{X+Z}^{\otimes 4} \mathrm{T}_4 \theta_{X+Z}^{\otimes 4} = \mathrm{T}_4.$$

Similarly, by straightforward calculation, we get

$$\mathrm{R}_Z^{\otimes 4} \mathrm{T}_4 \mathrm{R}_Z^{\otimes 4} = \mathrm{T}_4.$$

It is now easy to extend old circuit identities into new ones. For instance, conjugating the first circuit in the proof of Lemma 40 by $\theta_{X+Z}^{\otimes 4}$ (which does not change the circuit because of the above observations) and pushing the θ_{X+Z} gates into the inputs, yields

[7] Recall that θ_{X+Z} is commonly known as the Hadamard gate.

the second circuit. This technique is in fact very general and is used in the proof of the lemma below.

Lemma 41. T_4 *and* $C(P,P)$ *generate* R_P.

Proof. Figure 4-4 shows how to generate R_Z with $C(\mathsf{Z},\mathsf{Z})$. Using the argument by conjugation above, T_4 and $C(P,P)$ generate R_P. □

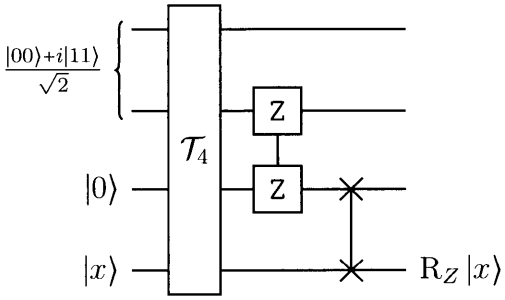

$\frac{|00\rangle+i|11\rangle}{\sqrt{2}}$
$|0\rangle$
$|x\rangle$
$\mathcal{T}_4$
Z
Z
$\mathrm{R}_Z\,|x\rangle$

Figure 4-4: Generating R_Z with T_4 and $C(Z,Z)$.

The following lemmas make precise our working assumption that single-qubit gates can significantly bolster the power of non-degenerate gate sets.

Lemma 42. *Suppose we have any* $C(P,Q)$ *gate with any single-qubit gate* G *that does not preserve the* P*-basis and any single-qubit gate* H *that does not preserve the* Q*-basis. Then* $\langle C(P,Q),G,H\rangle = \mathsf{ALL}$.

Proof. We will prove that the class $\langle C(P,Q),G,H\rangle$ contains all single-qubit gates. Then, to prove that the class generates all Clifford operations, it is be sufficient to show that it contains a CNOT gate. However, since all generalized CNOT gates are conjugates of each other, this is immediate.

First suppose $P = Q$. Since G does not preserve P-basis, we can use G to create a $C(R,R)$ gate where $R \neq P$. By Lemma 39, we can generate a θ_{P+R} gate. Conjugating $C(P,P)$ by θ_{P+R} on the second qubit yields a $C(P,R)$ gate. Once again leveraging Lemma 39, $C(P,R)$ and $C(P,P)$ generate an R_P gate. Referring to the single-qubit lattice (see Figure 4-2), we see that the class $\langle \mathcal{P}, \theta_{P+R}, \mathrm{R}_P\rangle$ contains all single-qubit gates.

Now suppose that $P \neq Q$. Once again, since G does not preserve P-basis, we can use G to create a $C(R,Q)$ gate. If $R = Q$, then by the logic above, we can use H to generate all single qubit gates, so suppose $R \neq Q$. By Lemma 39, we can use $C(P,Q)$ and $C(R,Q)$ to generate an R_Q gate. Conjugating both $C(P,Q)$ and $C(R,Q)$ by H appropriately, gives a $C(P,S)$ and $C(R,S)$ for some $S \neq Q$, which we can once again generate an R_S gate. Referring to the single-qubit lattice, we see that the class $\langle \mathcal{P}, \mathrm{R}_S, \mathrm{R}_Q\rangle$ contains all single-qubit gates. □

Lemma 43. T_4 *with the class of all single-qubit gates generates* ALL.

Proof. It is well known that CNOT, θ_{X+Z}, and R_Z generate all Clifford circuits. Therefore, it will be sufficient to show that T_4 plus all single-qubit gates generate CNOT. Under the ancilla rule, it is clear by Figure 4-5 that T_4 and R_Z suffice to generate $C(\mathtt{Z}, \mathtt{Z})$. Conjugating one qubit of $C(\mathtt{Z}, \mathtt{Z})$ by θ_{X+Z} yields a $C(\mathtt{Z}, \mathtt{X})$ = CNOT gate, completing the proof. □

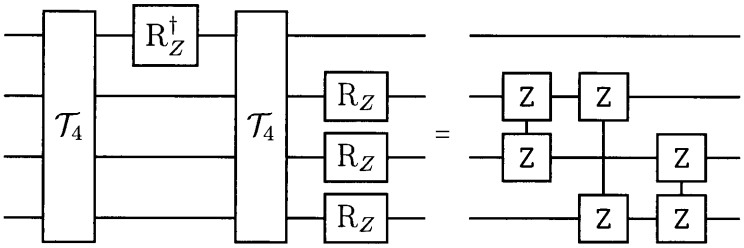


Figure 4-5: Generating $C(\mathtt{Z}, \mathtt{Z})$ with T_4 and R_Z.

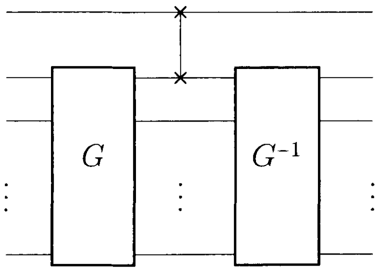


Figure 4-6: Universal Construction $\mathfrak{C}(G)$

4.7 Universal Construction

Suppose G is an n-qubit Clifford gate. It turns out there is a single circuit $\mathfrak{C}(G)$, the *universal construction*, which can help us extract useful generators (e.g., single-qubit gates, generalized CNOTs, etc.) from G. Specifically, the circuit $\mathfrak{C}(G)$ (shown in Figure 4-6) applies G to qubits 2 through $n+1$, swaps qubits 1 and 2, then applies G^{-1} to qubits 2 through $n+1$.

Note that after we apply G, all of the qubits but one go directly into G^{-1}, which should intuitively cancel out "most" of the effect G has on those qubits, isolating the effect of G on the swapped qubit. The following theorem makes this intuition more precise.

Theorem 44. *Let G be an n-qubit Clifford gate. Then*

$$\mathcal{M}(\mathfrak{C}(G)) = I_{n+1} + \begin{pmatrix} 1 \\ v \end{pmatrix} \begin{pmatrix} 1 & v^* \end{pmatrix}$$

where $v \in \mathsf{R}^{n\times 1}$ is the first column of $\mathcal{M}(G)$.

Proof. Let $\mathcal{M}(G) = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ with $A \in \mathsf{R}$, $B \in \mathsf{R}^{1\times(n-1)}$, $C \in \mathsf{R}^{(n-1)\times 1}$, and $D \in \mathsf{R}^{(n-1)\times(n-1)}$. Thus,

$$\mathcal{M}(G)\mathcal{M}(G^\dagger) = \begin{pmatrix} 1 & 0 \\ 0 & I_{n-1} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} A^* & C^* \\ B^* & D^* \end{pmatrix} = \begin{pmatrix} AA^* + BB^* & AC^* + BD^* \\ CA^* + DB^* & CC^* + DD^* \end{pmatrix}$$

which implies

$$\mathcal{M}(\mathfrak{C}(G)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & A & B \\ 0 & C & D \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & A^* & C^* \\ 0 & B^* & D^* \end{pmatrix} = \begin{pmatrix} 0 & A^* & C^* \\ A & BB^* & BD^* \\ C & DB^* & DD^* \end{pmatrix}$$
$$= \begin{pmatrix} 0 & A^* & C^* \\ A & 1 + AA^* & AC^* \\ C & CA^* & I_{n-1} + CC^* \end{pmatrix} = I_{n+1} + \begin{pmatrix} 1 \\ A \\ C \end{pmatrix} \begin{pmatrix} 1 & A^* & C^* \end{pmatrix}.$$

□

Tableaux of the form $I_n + vv^*$ have relatively simple circuit decompositions in terms of single-qubit generators and generalized CNOT gates, which is formalized in the following theorem.

Theorem 45. *Let $v = \{1, a_2, a_3, \dots, a_{2k}, b_1, \dots, b_\ell\} \in \mathsf{R}^{n\times 1}$ where each a_i is invertible and each b_i is singular, and let C be a Clifford circuit such that $\mathcal{M}(C) = I_n + vv^*$. Then C is equivalent to the circuit consisting of*

- *a T_{2k} gate on the first $2k$ qubits,*
- *conjugated by $\mathcal{G}(a_i, i)$ for all i in $\{2, \dots, 2k\}$,*
- *conjugated by $\mathrm{CNOT}(b_i, 1, 2k+i)$ for all $i \in \{1, \dots, \ell\}$, and*
- *a final layer of Pauli gates (*not *conjugated) on every qubit.*

That is, C is equivalent to the circuit in Figure 4-7.

Proof. Notice first that because each a_i is invertible, we can conjugate C by $\mathcal{G}(a_i, i)$ for each $i \in \{2, \dots, 2k\}$, yielding a circuit with the simpler tableau

$$\mathrm{Diag}(1, a_2, \dots, a_{2k}, 1, \dots, 1)(I_n + vv^*)\mathrm{Diag}(1, a_2^*, \dots, a_{2k}^*, 1, \dots, 1) = I_n + v'v'^*$$

where $v' = \{1, \dots, 1, b_1, \dots, b_\ell\}$. Furthermore, conjugating the circuit by the gate $\mathrm{CNOT}(b_i, 1, i+2k)$ corresponds to the simplification

$$\mathrm{CNOT}(b_i, 1, i+2k)(I_n + v'v'^*)\mathrm{CNOT}(b_i, 1, i+2k) = I_n + v''v''^*$$

where v'' is equal to v' with the exception that entry $i+2k$ is equal to zero. Repeating this procedure for each $i \in \{1, \dots, \ell\}$, we arrive at a circuit with a very simple tableau:

$$I_n + \{1, \dots, 1, 0, \dots, 0\}\{1, \dots, 1, 0, \dots, 0\}^T,$$

which is exactly the tableau of a T_{2k} gate applied to the first $2k$ qubits. Notice, finally, that by reversing the procedure and applying the appropriate Pauli gates to each qubit, we can ensure that the tableau of the decomposition is that of $\mathfrak{C}(G)$. □

Theorem 45 leads to a clean circuit decomposition of $\mathfrak{C}(G)$. All that is left to show is that we can actually generate each of the elementary gates that appears in the decomposition under the ancilla rule. First, we will need the following useful lemma, which will allow us to essentially disregard the Pauli operators in the decomposition of the universal construction when applying the ancilla rule.

Lemma 46. *Let G be a gate on n qubits which is stabilized by some state $|a\rangle$ on the first k qubits and generates H on the remaining qubits. Furthermore, let P be any gate on the first k qubits. Then $P \circ G$ generates $H \otimes H^{-1}$.*

Proof. By supposition we have that $G(|a\rangle \otimes |\psi\rangle) = |a\rangle \otimes H|\psi\rangle$. Therefore $P \circ G(|a\rangle \otimes |\psi\rangle) = P|a\rangle \otimes H|\psi\rangle$. Now apply the inverse of $P \circ G$ to $P|a\rangle \otimes H|\psi\rangle$ with the same first k qubits and $n-k$ new qubits. In the middle, P cancels with P^{-1}, and we can remove those k qubits by using $|a\rangle$ as an ancilla. On the remaining qubits, we have $H \otimes H^{-1}$. □

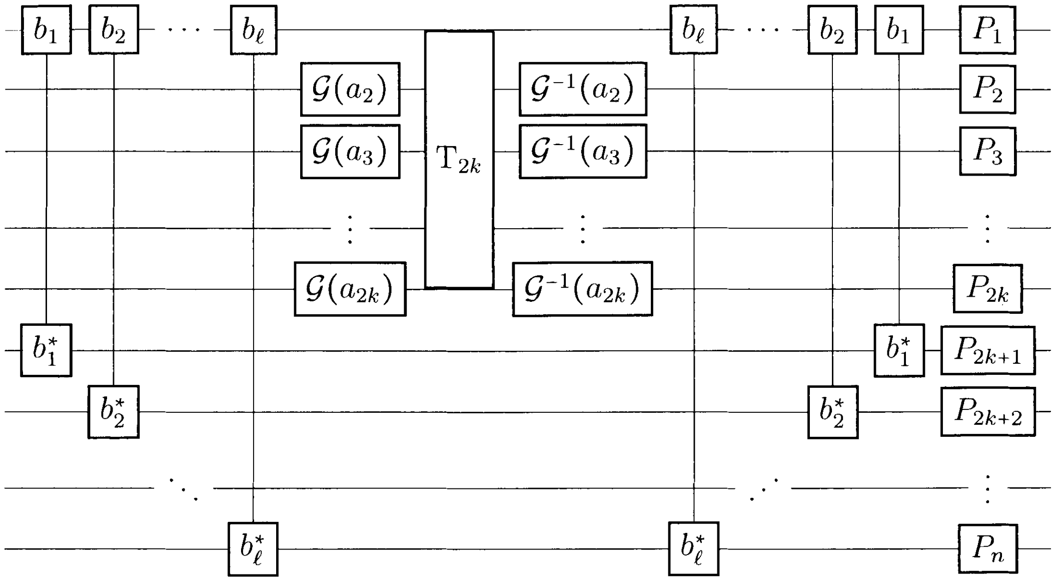


Figure 4-7: Decomposition of $\mathfrak{C}(G)$.

We are finally ready to prove the main theorem of this section.

Theorem 47. *Let G be a Clifford gate on n qubits. Furthermore, let*

$$v = (1, a_2, a_3, \ldots, a_{2k}, b_1, \ldots, b_\ell) \in \mathsf{R}^{n\times 1}$$

be a vector where each a_i is invertible and each b_i is singular be some row of $\mathcal{M}(G)$. Then G generates a gate $\mathcal{G}$ such that $\mathcal{M}(\mathcal{G}) = a_i$ for each $i \in \{2, \ldots, 2k\}$, CNOT(b_j) *for all $j \in \{1, \ldots, \ell\}$, and* T_{2k} *gate.*

Proof. From Theorem 44 and Theorem 45, we know that universal construction $\mathfrak{C}(G)$ can be decomposed as shown in Figure 4-7. The proof will proceed in the following manner. Starting with the decomposition of $\mathfrak{C}(G)$, we show that it generates some elementary gate. We then use that gate to simplify the original decomposition of $\mathfrak{C}(G)$, eventually generating all such gates in this manner.

First notice that for some input $i \in \{2k+1, \ldots, n\}$, the single-qubit Clifford state $|b_i\rangle$ serves to remove the effect of the generalized CNOT. By Lemma 46, we can remove the last ℓ qubits with ancillas, at the expense of creating another (inverted) copy of the remaining $2k$ bits. That is, we have $H \otimes H^{-1}$ where H is as shown in Figure 4-8.

Now let $|\phi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ be the Bell state on two qubits. Notice that we can use $|\phi\rangle$ as an ancilla to remove two bits from $\mathrm{T}_{2\ell}$ (i.e., leaving a $\mathrm{T}_{2\ell-2}$). However, the $\mathrm{T}_{2\ell}$ occurs in H conjugated by single-qubit gates, followed by Pauli operations. If we feed the state $\mathcal{G}(a_i, i)^\dagger \otimes \mathcal{G}(a_j, j)^\dagger |\phi\rangle$ to bits i and j, the single-qubit gates transform it to $|\phi\rangle$, it removes two bits from the $\mathrm{T}_{2\ell}$, then it is transformed to $P_i\mathcal{G}(a_i, i)^\dagger \otimes P_j\mathcal{G}(a_j, j)^\dagger |\phi\rangle$. We can do the exact same thing to H^{-1}, starting with $P_i\mathcal{G}(a_i, i)^\dagger \otimes P_j\mathcal{G}(a_j, j)^\dagger |\phi\rangle$ and going to $\mathcal{G}(a_i, i)^\dagger \otimes \mathcal{G}(a_j, j)^\dagger |\phi\rangle$. Then we swap the two states, and use the ancilla

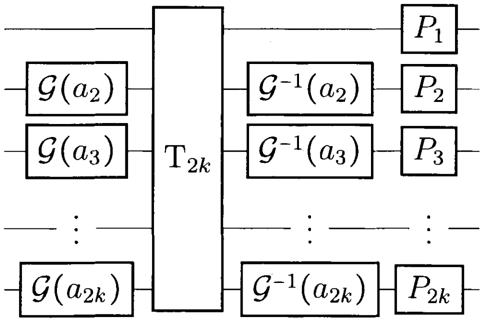
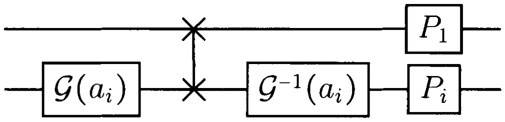


Figure 4-8: The circuit H.

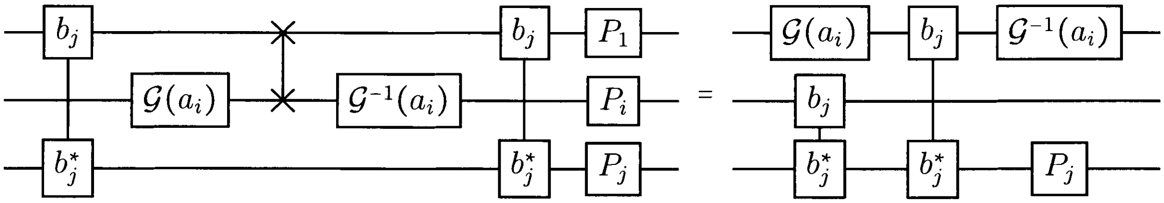
rule to remove them. The net result is that we can remove any two qubits of H, as long as we remove the same pair of qubits from H^{-1}.

Iterate this procedure to until H has been reduced to just the first two qubits, and so has H^{-1}. In particular, the $\mathrm{T}_{2\ell}$ gate in the middle is now a T_2, which we observe is actually a SWAP gate. From H, the remaining circuit is



and the inverse survives from H^{-1}. Remove the swaps, and observe that we have ancillas to remove any of the remaining single-qubit gates, so we can isolate each single-qubit gate, in particular $P_i \circ \mathcal{G}^{-1}(a_i)$, $P_1 \circ \mathcal{G}^{-1}(a_i)$ and their inverses. From this point on, everything we do for H can be repeated for H^{-1}, which culminates in removing all the single-qubit gates, at which point we can remove the extra $\mathrm{T}_{2\ell}$ gate easily. Hence, let us ignore H^{-1}.

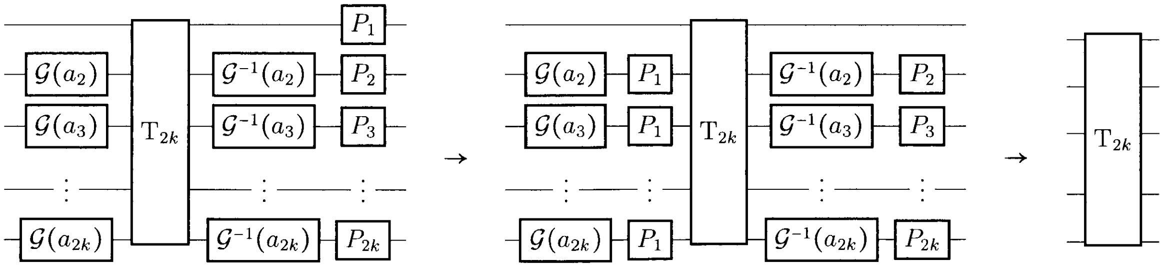
Now let us repeat the procedure above starting from the $\mathfrak{C}(G)$, but stop short of applying the ancilla rule to qubit $2k+j$. The result is the first circuit depicted below, which is then simplified by an application of the swap rule, and gates $P_i \circ \mathcal{G}^{-1}(a_i)$ and $P_1 \circ \mathcal{G}^{-1}(a_i)$:



Notice that the topmost qubit is stabilized by $\mathcal{G}^{-1}(a_i)\,|b_j\rangle$, from which we can see that the ancilla rule immediately generates $\mathrm{CNOT}(b_j)$. Finally, we exploit the identity,

$$T_{2k}(P \otimes I^{\otimes 2k-1})T_{2k} = I \otimes P^{\otimes 2k-1},$$

which holds up to a global phase. We have the following chain of consequences.



The last implication comes from the fact that we already generated $P_i \circ \mathcal{G}^{-1}(a_i)$ and $P_1 \circ \mathcal{G}^{-1}(a_i)$. □

4.8 Completing the Classification

The final step in the classification is to demonstrate that the classes we have defined are in fact the only classes that exist.

Theorem 48. *Let $\mathcal{S}$ be some class in the classification, and let G be a collection of gates. Suppose $\langle G \rangle \subseteq \mathcal{S}$, but $\langle G \rangle \nsubseteq \mathcal{S}'$ for all $\mathcal{S}'$ below $\mathcal{S}$ in our classification. That is, for all such $\mathcal{S}'$ there exists a gate $g \in G$ such that $g \notin \mathcal{S}'$. Then $\langle G \rangle = \mathcal{S}$.*

Proof. Let $\mathcal{C}$ be the class generated by G. There is a very general strategy for proving that the given class $\mathcal{C}$ is indeed one already stated in the classification. For each invariant described in Section 4.4 that $\mathcal{C}$ fails to preserve, the universal construction generates a (simple) gate which also fails to satisfy that invariant by Theorem 47. Composing these gates using identities from Section 4.6, one can show that they always generate some class in the classification. This would complete the classification.

Nevertheless, we now give a complete sequence of tests to identify the class $\mathcal{C}$. First consider the degeneracy invariant. If $\mathcal{C}$ is degenerate, then by Lemma 35 we can decompose each gate into a circuit of single-qubit gates and swap gates. Each single-qubit gate can be extracted with an appropriate tensor product of single-qubit ancillas on the rest. The question therefore reduces to the simple group-theoretic question about the subgroups of single-qubit gates, which can be solved straightforwardly. Therefore, assume $\mathcal{C}$ is non-degenerate.

Let us separate the remainder of the proof based on the X-, Y-, Z-preserving invariant. Let P, Q, and R be distinct Pauli operations.

Suppose first that $\mathcal{C}$ is X-, Y-, and Z-preserving. Because every generalized CNOT gate violates one such invariant and $\mathcal{C}$ is non-degenerate, some gate in $\mathcal{C}$ must have a tableau with multiple invertible elements in some row. Therefore, from the universal construction we extract a T_4 gate. By Lemma 40, we have then that $\mathcal{C} = \langle \mathrm{T}_4, \mathcal{P} \rangle$.

Suppose now that $\mathcal{C}$ is P- and Q-preserving but not R-preserving. Because $\mathcal{C}$ is non-degenerate, it must generate either a heterogeneous CNOT gate, a homogenous CNOT gate, or a T_4 gate. We wish to show that $\mathcal{C}$ contains a $C(P,Q)$ gate, which would imply $\mathcal{C} = \langle C(P,Q), \mathcal{P} \rangle$ as desired. First notice that no homogeneous CNOT gate can be P- and Q-preserving. Suppose then that from the universal construction, $\mathcal{C}$ generates a T_4 gate but no $C(P,Q)$ gate. Since T_4 is R-preserving, there must be some single-qubit gate from the universal construction that is not R-preserving but is P- and Q-preserving. It is straightforward to check that no such single-qubit gate exists, which implies that $\mathcal{C}$ must contain a $C(P,Q)$ gate.

Suppose now that $\mathcal{C}$ is P-preserving but not Q- and R-preserving. This is the most involved case and will require several more subdivisions. It will be first useful to notice that all P-preserving single-qubit gates are also P-orthogonal. Therefore, if $\mathcal{C}$ violates the P-orthogonality invariant, then the universal construction must produce some non-degenerate gate which violates P-orthogonality. Therefore, $\mathcal{C}$ contains a $C(P,Q)$ gate or a $C(P,R)$ gate. If it contains both via the universal construction, then indeed $\mathcal{C} = \langle C(P,Q), \mathrm{R}_P, \mathcal{P} \rangle$ by Lemma 39. Otherwise, the universal construction produces some single-qubit gate which is P-preserving but not Q- and R-preserving.

Since all heterogeneous CNOT gates generate the Pauli group as well, the class of single qubit gates must therefore contain an R_P gate.

Therefore let us now assume that $\mathcal{C}$ is P-orthogonal but not P-degenerate. Since $\mathcal{C}$ is P-orthogonal, it cannot contain a $C(P,Q)$ or $C(P,R)$. However, because $\mathcal{C}$ is P-preserving, it must contain a T_4 gate; otherwise it would be P-degenerate. Once again, since the T_4 gate generates the Pauli group and $\mathcal{C}$ is neither Q- nor R-preserving, the class of single qubit gates must contain an R_P gate, implying that $\mathcal{C} = \langle \mathrm{T}_4, \mathrm{R}_P, \mathcal{P} \rangle$.

Let us then assume that $\mathcal{C}$ is P-degenerate. Since $\mathcal{C}$ is non-degenerate, it must contain a $C(P,P)$ gate. There are five P-degenerate classes, which are determined by their single-qubit subgroup. Indeed, the five P-degenerate classes correspond to the five P-preserving single-qubit classes containing P. Unlike previous cases, such a diversity of classes exists because $C(P,P)$ does not suffice to generate the Pauli group on its own. Once again, the universal construction allows us to extract the $C(P,P)$ gate along with single-qubit gates which suffice to generate every gate in $\mathcal{C}$. It is straightforward to see why the entire single-qubit group must arise from the universal construction. If not, then each gate in $\mathcal{C}$ could be constructed by gates in a smaller class, a contradiction. This completes the classification of all gates that are P-preserving but not Q- and R-preserving.

Assume then that $\mathcal{C}$ is neither P-,Q-, nor R-preserving. If it is egalitarian, then it must contain a T_4 gate because it is non-degenerate. Therefore it must contain a single-qubit gate that is egalitarian, but not P-,Q-, or R-preserving. The only such single-qubit class which contains the Pauli group is $\langle \Gamma, \mathcal{P} \rangle$. Therefore $\mathcal{C} = \langle \mathrm{T}_4, \Gamma, \mathcal{P} \rangle$.

Finally, let us then assume that $\mathcal{C}$ also violates the egalitarian invariant. That is, $\mathcal{C}$ violates every invariant so should be equal to the class of all Clifford operations. Suppose the only non-degenerate gate generated by the universal construction is the T_4 gate. In particular, this implies that class of single-qubit gates generated by $\mathcal{C}$ must not be X-, Y-, Z-preserving, nor egalitarian. Therefore the single-qubit class of $\mathcal{C}$ must contain *all* single-qubit gates. Therefore by Lemma 43 we generate all Clifford circuits.

Similarly, Lemma 39 implies that if the set of generalized CNOT gates generated by the universal construction fails to be P-, Q-, or R-preserving, then $\mathcal{C}$ generates a single-qubit gate which fails to be P-, Q-, or R-preserving, respectively. Combining this fact with Lemma 42 implies that $\mathcal{C} = \mathsf{ALL}$. □

Corollary 49. *Given any set of gates G, there is a subset $S \subseteq G$ of at most three gates such that $\langle S \rangle = \langle G \rangle$.*

Proof. The result follows by a careful accounting of the gates used in the proof of Theorem 48. We give the argument only for the degenerate classes.

Let G be a set of single-qubit gates with $S \subseteq G$ and such that $\langle S \rangle = \langle G \rangle$. Suppose $S = \{g_1, \ldots, g_k\}$ with $k > 3$. We can assume that each generator in the set S is not contained in the subgroup generated by the other elements in S, otherwise we could remove that generator, reducing the size of S. Therefore, there is an ascending chain of subgroups $G_0 \subset G_1 \subset \ldots \subset G_k$, where $G_i = \langle g_1, \ldots, g_i \rangle$. Observing Figure 4-2, the longest ascending chain has length four, and in particular, all chains of length four end at the class of *all* single-qubit gates. Furthermore, that chain must contain one of the subgroups $\langle \mathcal{P}, \mathrm{R}_X \rangle$, $\langle \mathcal{P}, \mathrm{R}_Y \rangle$, or $\langle \mathcal{P}, \mathrm{R}_Z \rangle$, which we can assume is $\langle \mathcal{P}, \mathrm{R}_Z \rangle$ by symmetry.

Since not all gates in S are Pauli operators, we can assume that g_1 is not a Pauli operator. In particular, $g_1 \in \langle \mathcal{P}, \mathrm{R}_Z \rangle \smallsetminus \langle \mathcal{P} \rangle$. That is, g_1 is one of R_Z, $\mathrm{R}_Z^\dagger$, θ_{X+Y}, or θ_{X-Y}. Therefore, the class G_2 is either $\langle \mathrm{R}_Z \rangle$ or $\langle \mathsf{Z}, \theta_{XY} \rangle$. Clearly, $g_4 \notin \langle \mathcal{P}, \mathrm{R}_Z \rangle$ (i.e., it does not preserve the Z-basis), and by a simple case analysis, it is easy to see that $\langle g_1, g_2, g_4 \rangle$ is the entire class of single-qubit operations. □

4.9 Consequences of the Classification and Related Results

We have completed the classification of Clifford gates. Now we want to answer some questions (e.g., how many m-qubit Clifford gates are there in each class, or how can we better understand 2-qubit Clifford gates) using the new classification result, and also answer some questions that came up while proving the classification (e.g., what do the three-qubit gates generating $\langle \mathrm{T}_4, \Gamma, \mathcal{P} \rangle$ look like).

4.9.1 Enumeration

Theorem 50. *Let $\# \langle \cdot \rangle_n$ denote the number of n qubit gates in a class. Then*

$$\begin{aligned}
\# \langle G \rangle_n &= |G|^n n! && \text{for } G \text{ a group of single-qubit gates,}\\
\# \langle C(Z,Z), G \rangle_n &= |G|^n 2^{n(n-1)/2} n! && \text{for } \langle Z \rangle \subseteq G \subseteq \langle \mathcal{P}, \mathrm{R}_Z \rangle \text{ a group,}\\
\# \langle C(Z,X), \mathcal{P} \rangle_n &= 4^n 2^{n(n-1)/2} \prod_{i=1}^{n} (2^i - 1),\\
\# \langle C(Z,X), \mathcal{P}, \mathrm{R}_Z \rangle_n &= 8^n 2^{n(n-1)} \prod_{i=1}^{n} (2^i - 1),\\
\# \langle \mathrm{T}_4, \mathcal{P} \rangle_n &= 4^n a(n),\\
\# \langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Z \rangle_n &= 8^n 2^{n(n-1)/2} a(n),\\
\# \langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle_n &= 4^n 2^{n(n-1)/2} \prod_{i=1}^{n} (2^i - (-1)^i),\\
\# \langle \mathsf{ALL} \rangle_n &= 4^n 2^{n^2} \prod_{i=1}^{n} (4^i - 1),
\end{aligned}$$

where

$$a(n) = \begin{cases} 2^{m^2} \prod_{i=1}^{m-1} (2^{2i} - 1), & \text{if } n = 2m,\\ 2^{m^2} \prod_{i=1}^{m} (2^{2i} - 1), & \text{if } n = 2m+1. \end{cases}$$

Proof. Most of these numbers follow from the lemmas above. For example, consider the class $\langle C(\mathtt{Z},\mathtt{X}), \mathcal{P}, \mathrm{R}_Z \rangle$. It follows from Lemma 37 that any gate in this class has a circuit consisting of a layer of $C(\mathtt{Z},\mathtt{X})$ gates, then a layer of $C(\mathtt{Z},\mathtt{Z})$ gates, then a layer of single-qubit gates in G.

We would like to count the number of possible gates by multiplying the number of possibilities for each layer, but we must be careful that there is no gate with two circuit representations. Suppose for a contradiction that g_1 and g_2 generate the same gate, but some layer of g_1 differs from g_2. Then $g_1^{-1} g_2$ is the identity, since g_1 and g_2 generate the same transformation.

On the other hand, the $C(\mathtt{Z},\mathtt{X})$ layers of g_1 and g_2 meet in the middle of the circuit for $g_1^{-1} g_2$. If those layers do not generate the same linear transformation, then the combination is some non-trivial linear transformation which is, in particular, not Z-degenerate. The other layers of g_1 and g_2 *are* Z-degenerate, so we conclude that

$g_1^{-1}g_2$ is not Z-degenerate (if it were, we could invert the outer layers to show that the two middle layers are Z-degenerate). But $g_1^{-1}g_2 = I$ is clearly Z-degenerate, therefore the $C(\mathtt{Z},\mathtt{X})$ layers of g_1 and g_2 must generate the same linear transformation.

The $C(\mathtt{Z},\mathtt{X})$ layers of g_1 and g_2 cancel (since we have shown they are equivalent), so they effectively disappear, and we make a similar argument about the $C(\mathtt{Z},\mathtt{Z})$ layers, and then the single-qubit layers. That is, if the $C(\mathtt{Z},\mathtt{Z})$ layers do not contain the same set of $C(\mathtt{Z},\mathtt{Z})$ gates, then we obtain a contradiction because they produce a non-degenerate layer in the middle, implying that $g_1^{-1}g_2 = I$ is non-degenerate. Once we remove the $C(\mathtt{Z},\mathtt{Z})$ layers, the single-qubit layers must be the same or they would leave behind a non-trivial single-qubit gate. We conclude that all layers of g_1 and g_2 are actually the same, so the number of gates is the product of the number of choices for each layer.

Now the problem is to count the number of choices for each layer. For the single qubit layer, this is clearly just n independent choices of single-qubit gate from $\langle \mathcal{P}, \mathrm{R}_Z \rangle$, or 8^n. For the $C(\mathtt{Z},\mathtt{Z})$ layer, there is a choice whether or not to place a $C(\mathtt{Z},\mathtt{Z})$ gate in each of the $\binom{n}{2}$ possible positions, so $2^{n(n-1)/2}$ choices for the layer. For the $C(\mathtt{Z},\mathtt{X})$ layer, observe that $C(\mathtt{Z},\mathtt{X})$ generate precisely the set of invertible linear transformations, of which there are

$$2^{n(n-1)/2} \prod_{i=1}^{n} (2^i - 1)$$

by a classical argument. Multiplying the three layers, we have a total of

$$\# \langle C(\mathtt{Z},\mathtt{X}), \mathcal{P}, \mathrm{R}_Z \rangle_n = 8^n 2^{n(n-1)} \prod_{i=1}^{n} (2^i - 1)$$

n-qubit transformations generated by $C(\mathtt{Z},\mathtt{X})$, $\mathcal{P}$, and R_Z.

The numbers for $\langle G \rangle$, $\langle C(\mathtt{Z},\mathtt{Z}), G \rangle$, $\langle C(\mathtt{Z},\mathtt{X}), \mathcal{P} \rangle$, $\langle \mathrm{T}_4, \mathcal{P} \rangle$, and $\langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Z \rangle$ follow by a similar argument, although for the last two classes we need the fact that T_4 generates

$$a(n) = \begin{cases} 2^{m^2} \prod_{i=1}^{m-1} (2^{2i} - 1), & \text{if } n = 2m, \\ 2^{m^2} \prod_{i=1}^{m} (2^{2i} - 1), & \text{if } n = 2m + 1. \end{cases}$$

orthogonal transformations on n qubits.

For the final two classes, we use known expressions (from [35]) for the number of $n \times n$ unitary matrices over $\mathbb{F}_4$ (in the case of $\langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle$) and for the number of $2n \times 2n$ symplectic matrices over $\mathbb{F}_2$ (in the case of ALL). We multiply by 4^n in both cases to account for the phase bits, which are completely independent of the matrix part. □

Theorem 51. *The asymptotic size of each class is as follows.*

$$\begin{aligned}
\log_2 \# \langle G \rangle_n &= n \log_2(n|G|/e) + \frac{1}{2}\log_2 2\pi + O\left(\frac{1}{n}\right), \\
\log_2 \# \langle C(Z,Z), G \rangle_n &= \frac{n(n-1)}{2} + n \log_2(n|G|/e) + \frac{1}{2}\log_2 2\pi + O\left(\frac{1}{n}\right), \\
\log_2 \# \langle C(Z,X), \mathcal{P} \rangle_n &= n^2 + 2n - \alpha + O(2^{-n}), \\
\log_2 \# \langle C(Z,X), \mathcal{P}, \mathrm{R}_Z \rangle_n &= \frac{3}{2}n^2 + \frac{5}{2}n - \alpha + O(2^{-n}), \\
\log_2 \# \langle \mathrm{T}_4, \mathcal{P} \rangle_n &= \frac{1}{2}n^2 + \frac{3}{2}n - \beta + O(2^{-n}), \\
\log_2 \# \langle \mathrm{T}_4, \mathcal{P}, \mathrm{R}_Z \rangle_n &= n^2 + 3n - \beta + O(2^{-n}), \\
\log_2 \# \langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle_n &= n^2 + 2n + \gamma + O(2^{-n}), \\
\log_2 \# \langle \mathsf{ALL} \rangle_n &= 2n^2 + 3n - \beta + O(4^{-n}).
\end{aligned}$$

where G is the same as in Theorem 50, and

$$\begin{aligned}
\alpha &= -\sum_{i=1}^{\infty} \log_2(1 - 2^{-i}) \approx 1.7919, \\
\beta &= -\sum_{i=1}^{\infty} \log_2(1 - 4^{-i}) \approx 0.53839, \\
\gamma &= \sum_{i=1}^{\infty} \log_2(1 - (-2)^{-i}) \approx 0.27587.
\end{aligned}$$

Proof. We take the logarithm of each class size, which we can separate into the logarithm of each layer comprising that class, as in Theorem 50. For most layers this is straightforward, except for the layer of permutations, orthogonal transformations, or general linear transformations. The first we handle with Stirling's approximation. For the other two, we factor out powers of two leaving a partial sum of a convergent series, which we analyze with a Taylor expansion. The classes $\langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle_n$ and $\langle \mathsf{ALL} \rangle_n$ follow by similar techniques. □

Corollary 52. *Let $\mathcal{C}$ be any class, and let G be an n-qubit gate chosen uniformly at random from $\mathcal{C}$. Then*

$$\Pr[G \textit{ generates } \mathcal{C}] = 1 - O(2^{-n}).$$

4.9.2 Classical reversible gates with quantum ancillas

In this section we describe what the classical reversible gate lattice of Aaronson et al. [5] would have looked like had they allowed quantum rather than classical ancillas. Classical reversible gates, the subject of the Aaronson et al. paper, are immediately quantum gates. When we treat them as such, and allow quantum ancillas, the lattice simplifies dramatically Figure 4-10.

Some of the collapses in the lattice are immediate. For instance, the class of gates

$\langle \mathrm{NOT} \otimes \mathrm{NOT} \rangle$ collapses with $\langle \mathrm{NOT} \rangle$ because $\mathrm{NOT}|+\rangle = |+\rangle$ can be used to cancel a NOT gate. A similar collapse occurs for $\langle \mathrm{CNOTNOT} \rangle$ and $\langle \mathrm{CNOT} \rangle$, and so on for all pairs of classes separated only by parity.

A more interesting collapse occurs between all mod-k-preserving classes for $k \geq 2$. Consider the following gate $G : \{0,1\}^k \to \{0,1\}^k$ of order 2 which preserves Hamming weight mod k:

$$\begin{aligned} G(0^k) &= 1^k \\ G(1^k) &= 0^k \\ G(1^a 0^b 0) &= 1^{a-1} 0^{b+1} 1 \\ G(1^a 0^b 1) &= 1^{a+1} 0^{b-1} 0 \end{aligned}$$

where G acts as the identity on all other inputs. Since G preserves the Hamming weight mod k, it must appear in the class. We will show that G can generate a NOT gate. Let

$$|\psi_k\rangle := \frac{1}{\sqrt{k}} \sum_{i=0}^{k-1} \left|1^i 0^{k-i-1}\right\rangle$$

so, for example

$$|\psi_4\rangle = \frac{|000\rangle + |100\rangle + |110\rangle + |111\rangle}{2}.$$

Now, for $b \in \{0,1\}$, $G(|\psi_k\rangle |b\rangle) = |\psi_k\rangle |b \oplus 1\rangle$. Therefore, each mod-$k$-preserving class for $k \geq 2$ collapses to the $\langle \mathrm{Fredkin}, \mathrm{NOT} \rangle$ class. Furthermore, Figure 4-9 shows that the Fredkin and NOT gates are sufficient to generate a CNOT. Therefore, every non-conservative non-affine class generates all classical reversible transformations.

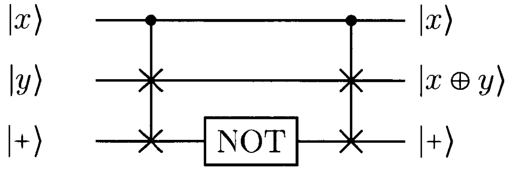


Figure 4-9: Generating CNOT from Fredkin and NOT gates

We now only need to prove that the classes appearing in Figure 4-10 are distinct. Notice, however, that the classes $\langle \mathrm{CNOT} \rangle$, $\langle \mathrm{T}_4 \rangle$, and $\langle \mathrm{NOT} \rangle$ all have Clifford generators, which by the classification result of this chapter, generate distinct classes. We only need to show then, that the $\langle \mathrm{Fredkin} \rangle$ class is distinct from the remaining classes. However, the invariant in [5] more or less functions to prove this separation. Namely, Fredkin conserves the Hamming weight of its input. Therefore the sum of the Hamming weights of the computational basis states of the input state is conserved. However, the NOT gate necessarily changes this sum, witnessing that $\mathrm{NOT} \notin \langle \mathrm{Fredkin} \rangle$, and therefore that the lattice is complete.

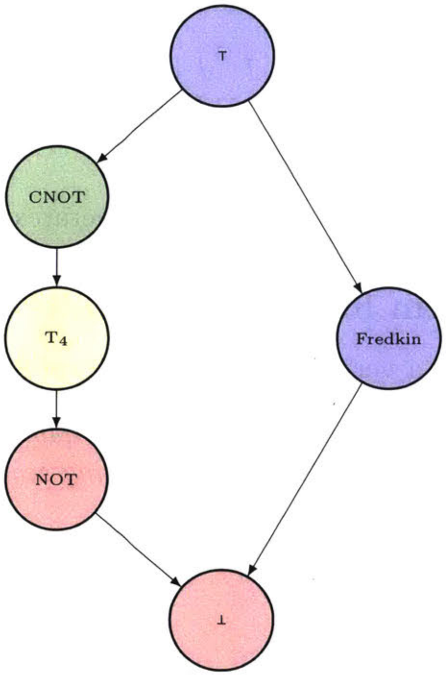


Figure 4-10: The inclusion lattice of classical gates using quantum ancillas

4.9.3 Three-qubit generator for $\langle \mathrm{T}_4, \Gamma, \mathcal{P}\rangle$

In this section, we point out a peculiar consequence of the classification. Namely, that there exists a class whose smallest generator is on 3 qubits. Contrasted with the affine gate sets of Aaronson et al. [5] which have minimal generators over 1, 2, 4, and 6 bits. The existence of such a class is an immediate consequence of the enumeration in Section 4.9.1. That is,

$$\begin{aligned}
&\#\langle \mathrm{T}_4, \mathcal{P}, \Gamma\rangle_3 = 2^{3(3-1)/2+2(3)} \prod_{i=1}^{3}(2^i - (-1)^i) = 41472\\
&\#\langle \Gamma, \mathcal{P}\rangle_3 = 12^3 3! = 10368\\
&\#\langle \mathrm{T}_4, \mathcal{P}\rangle_3 = 4^3 2^{1^2} \prod_{i=1}^{1}(2^{2i} - 1) = 384\\
&\#\langle \mathcal{P}\rangle_3 = 4^3 3! = 384
\end{aligned}$$

so the non-degenerate gates in $\langle \mathrm{T}_4, \mathcal{P}, \Gamma\rangle_3$ outnumber the degenerate gates (i.e., the class $\#\langle \Gamma, \mathcal{P}\rangle_3$), contrasted with the class $\langle \mathrm{T}_4, \mathcal{P}\rangle$, where this does not occur. Notice that there are 4 cosets of $\langle \Gamma, \mathcal{P}\rangle_3$ in $\langle \mathrm{T}_4, \mathcal{P}, \Gamma\rangle_3$ by Lagrange's Theorem, corresponding to 4 gates that are nonequivalent up to applications of elements in $\langle \Gamma, \mathcal{P}\rangle_3$. If we let

$\alpha = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right) \in R$, then one such gate is described by the following tableau

$$\left(\begin{array}{ccc|c} \alpha & I & I & 0 \\ I & \alpha & I & 0 \\ I & I & \alpha & 0 \end{array}\right).$$

By Theorem 48 it is clear that this gate indeed generates all of $\langle \mathrm{T}_4, \mathcal{P}, \Gamma \rangle$.

4.9.4 Canonical form for 2-qubit circuits

In this section, we describe a very clean canonical form for 2-qubit Clifford circuits.

Theorem 53. *Let C be any Clifford circuit on two qubits. Then, C is equivalent to a circuit of at most depth 3 composed of the following sequence of gates*

1. *a SWAP gate, and*
2. *a tensor product of single-qubit gates, and*
3. *a generalized CNOT gate,*

where we can choose at each step whether or not to include the gate. That is, C is of the form of the circuit depicted in Figure 4-11.

Proof. Since C is a Clifford circuit, it can be written as a product of CNOT, θ_{X+Z}, and R_Z gates.[8] Recall that conjugating a generalized CNOT gate by a single-qubit gate is simply another generalized CNOT gate. Therefore, we can push all the single-qubit gates left and all the generalized CNOT gates right. All that remains to show is that we can coalesce the generalized CNOT gates into a single CNOT gate. We refer to Table 4.3 for those equivalences, and note that identical generalized CNOT gates cancel. Eventually, what remains is a circuit composed of single-qubit gates, SWAP gates, and at most one generalized CNOT gate. We can push the SWAP gates to the left (they collapse to either a single SWAP gate or the identity) and combine the single-qubit gates, which completes the proof. □

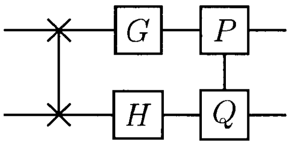


Figure 4-11: Canonical form of a 2-qubit circuit: optional SWAP gate, optional $C(P, Q)$ gate, and single-qubit gates G and H.

[8] Notice that we don't need to include SWAP gates because of the equivalence $\mathrm{SWAP}(1,2) = \mathrm{CNOT}(1,2) \circ \mathrm{CNOT}(2,1) \circ \mathrm{CNOT}(1,2)$.

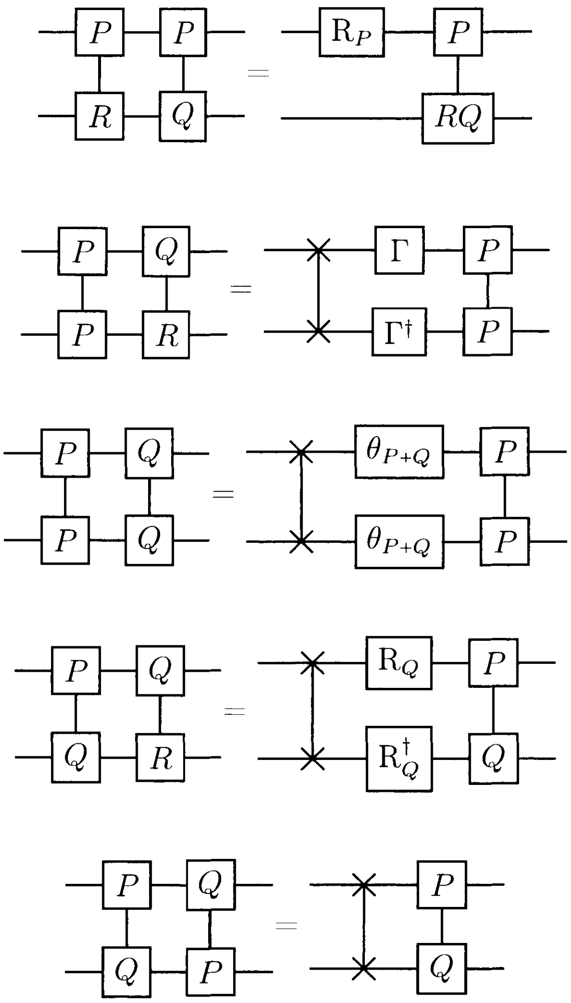


Table 4.3: Rules for coalescing generalized CNOT gates, assuming $\Gamma P \Gamma^\dagger = Q$ and $\Gamma Q \Gamma^\dagger = R$.

4.10 Open Problems

Our classification of Clifford gates resolves an open problem of Aaronson et al. [5], but leaves their central question, the classification of arbitrary quantum gates, completely open. It is unclear whether there is another piece of the full quantum gate classification that can be peeled off. Other discrete quantum gate sets are known, but none are known to have the rich structure and entanglement of Clifford gates (aside from conjugated Clifford gates). So we ask: are there other interesting discrete gate sets, and can they be classified like Clifford gates?

Another source of open problems is the choice of ancilla rule. As discussed, we permit ancillas initialized to quantum states. We have determined that the classification continues to hold under a stabilizer ancilla model if the following conjecture holds:

Conjecture 54. *For any single-qubit Clifford gate g, there exists a stabilizer state $|\psi\rangle$ and circuit of* SWAP *gates π such that $g \circ \pi |\psi\rangle = |\psi\rangle$.*

This is sufficient to remove single-qubit gates in situations where we would otherwise use an eigenstate.

For many single-qubit gates, there is a trivial stabilizer state which stabilizes it. For instance, X is stabilized by $|+\rangle$, R_Z is stabilized by $|0\rangle$, and many other single-qubit Clifford gates are conjugate to one of these cases. Now consider the gate θ_{X+Z}, whose eigenstates (unnormalized) $(1 \pm \sqrt{2})|0\rangle + |1\rangle$ are *not* stabilizer states. How then, given the gate $\theta_{X+Z} \otimes \theta_{X+Z}$, does one generate the gate θ_{X+Z} which acts only on one qubit? Han-Hsuan Lin discovered the first explicit nine qubit stabilizer state for this task.

Let π be a circuit that cyclicly permutes qubits 2 through 9, and suppose θ_{X+Z} is applied to qubit 1. Let $|\psi\rangle$ be the state stabilized by the following commuting Pauli strings,

$$\begin{array}{cccc}
+\mathtt{XXZXZIIII}, & +\mathtt{ZIXZXZIII}, & +\mathtt{XIIXZXZII}, & +\mathtt{ZIIIXZXZI}, \\
+\mathtt{XIIIIXZXZ}, & +\mathtt{ZZIIIIXZX}, & +\mathtt{XXZIIIIXZ}, & +\mathtt{ZZXZIIIIX}, \\
+\mathtt{YYIIIYIII}, & -\mathtt{YIYIIIYII}, & +\mathtt{YIIYIIIYI}, & -\mathtt{YIIIYIIIY},
\end{array}$$

9 of which are independent. One can check that conjugating each generator by $\theta_{X+Z} \circ \pi$ yields another element of the stabilizer group, so $(\theta_{X+Z} \circ \pi)|\psi\rangle = |\psi\rangle$. In other words, Conjecture 54 holds for θ_{X+Z}, and for all conjugates θ_{P+Q} by symmetry.

All that remains to verify the conjecture is to find a similar state stabilizing the eight remaining gates—the Γ gate and its conjugates. It suffices to find a stabilizer state $|\psi\rangle$ and circuit C, constructed of SWAP gates and a single Γ gate, such that $C|\psi\rangle = |\psi\rangle$.

Chapter 5

Low-Depth Quantum Circuits: Separation from AC^0

This chapter and the next are devoted to two results about shallow quantum circuits. In other words, quantum circuits where the gates can be executed in parallel in a small number of layers (specifically $O(1)$ layers for us, but more generally $\mathsf{polylog}(n)$ layers is considered shallow). Unlike the previous two chapters (on grid-based computation and Clifford gates), these results are much more directly related to proving a quantum advantage, and therefore inherently relevant to near-term quantum devices. Let us take a step back for a moment and look at why shallow quantum circuits are important for quantum computing in the near future.

First, shallow quantum circuits are well motivated from a practical perspective, as we might actually be able to implement such circuits on near-term quantum computers! In the current era of Noisy Intermediate-Scale Quantum (NISQ) computers, there is a high error rate on each quantum gate (in the neighborhood of 1% for 2-qubit gates [68]), as well as a rate of error just keeping idle qubits coherent (e.g., lifetimes on the order of tens of microseconds in IBM devices [73]). We are limited to running quantum algorithms for a short amount of time before errors accumulate and noise overwhelms the signal. In the circuit model, limited time implies limited depth, so we are motivated to find interesting problems that can still be implemented in shallow depth.

Of course, we are *far* from the first to consider shallow circuits. Nick Pippenger's early work on *classical* shallow circuits was already recognized in 1979 with the naming of "Nick's Class" [37], NC, and the field exploded in the following decade with a series of unconditional impossibility results [6, 49, 122, 61]. It cannot be overstated how rare it is in complexity theory to prove a natural problem cannot be computed in some nontrivial complexity class, especially a non-uniform class; invariably the proof is non-constructive (or the problem is unnatural, e.g., diagonalization), or is actually a reduction to a previous complexity assumption, or only holds for an extremely weak and/or uniform model of computation (e.g., showing a language is not a context-free language). Indeed, constant-depth circuits remain the frontier of circuit lower bounds

[0] This chapter is based on work from [120], and used with the permission the authors.

and an active area of research in classical complexity theory today [121, 86].

On the quantum side, there has naturally been similar interest over the past 20 years in low-depth quantum circuits [84, 83, 54, 113, 47, 64, 46, 110], in part for the practically minded reasons above, but in part to see what happens when we turn a celebrated classical model into a *quantum* model. Such circuits may be viewed as parallel quantum computers with a constant running time bound. Several variations on this theme have been studied (see [17] for a survey of older results), and in recent years there has been a resurgence of interest [111, 19, 26, 38, 76] in constant-depth quantum circuits. Our results are part of this resurgence, and in particular to a recent exciting result of Bravyi, Gosset, and König [26].

Bravyi, Gosset, and König [26] defined a relation problem called the 2D Hidden Linear Function (2D HLF) problem. (We define this problem in Section 5.5.) The 2D HLF problem can be solved by a constant-depth quantum circuit that uses bounded fan-in quantum gates, acting locally on a 2-dimensional grid of qubits, nicely combining the themes of this thesis.

Furthermore, Bravyi, Gosset, and König [26] show that the 2D HLF problem cannot be solved by any constant-depth classical circuit using unbounded fan-out and bounded fan-in gates. Their lower bound even holds when the classical circuit is allowed to sample from an arbitrary probability distribution on polynomially many bits that does not depend on the input. (In complexity theory, this resource is called "randomized advice.") With the complexity notation from Chapter 2, we can summarize the Bravyi et al. result as follows [26].

Theorem 55 (Bravyi, Gosset, and König). *The* 2D HLF *problem can be solved exactly by a* QNC^0 *circuit on a 2D grid, but no* $\mathsf{NC}^0/\mathsf{rpoly}$ *circuit can solve the problem with probability greater than* 7/8 *on every input. In other words,* $\mathsf{RelQNC}^0 \not\subseteq \mathsf{RelNC}^0/\mathsf{rpoly}$.

The fact that the separating problem is a relation problem and not a function (or decision) problem is unavoidable, since the decision class QNC^0 *is* contained in NC^0: any function in QNC^0 has output bits that only depend on a constant number of input bits, due to the bounded fan-in gates, and hence such a function would also be in NC^0.

This result was also recently improved by Coudron, Stark, and Vidick [38], and (independently) Le Gall [76], who extended the lower bound to an average-case lower bound. As opposed to saying that no NC^0 circuit can solve the problem on *all* inputs, an average-case hardness result says that no NC^0 circuit can solve the problem even on some fraction of the inputs.[1] These results show that no NC^0 circuit can solve the problem with input size n on an $\exp(-n^\alpha)$ fraction of the inputs for some $\alpha > 0$.

5.1 Results

In this chapter and the next, we attempt to strengthen the weakest aspect of the Bravyi, Gosset, König result: they prove a separation from the classical class RelNC^0,

[1] Note that [26, Appendix C.3] already shows mild average-case hardness for this problem.

which is abysmally weak. It does not suggest any real world experiment that could be done to demonstrate a quantum advantage, even though the quantum circuit is almost ideal for implementation on near-term quantum devices.

Our result in this chapter extends the separation to $\mathsf{RelQNC}^0 \not\subseteq \mathsf{RelAC}^0$, even using the same problem (2D HLF). That is, even with unbounded fan-in AND and OR gates, constant-depth classical circuits cannot compute the 2D HLF problem. This is especially interesting because AC^0 circuits can compute functions that depend on all bits, such as the logical OR of all its inputs, whereas NC^0 circuits cannot, and this breaks the proof of Bravyi, Gosset, and König. Our main result is the following.

Theorem 56 (2D HLF). *The* 2D HLF *problem on n bits cannot be solved by an AC^0 circuit of depth d and size at most $\exp(n^{1/10d})$. Furthermore, there exists an (efficiently sampleable) input distribution on which any AC^0 circuit (or $\mathsf{AC}^0/\mathsf{rpoly}$ circuit) of depth d and size at most $\exp(n^{1/10d})$ only solves the* 2D HLF *problem with probability at most $\exp(-n^\alpha)$ for some $\alpha > 0$.*

Thus our result proves a (still unconditional) separation against a larger complexity class and implies the worst-case lower bound of Bravyi, Gosset, and König [26]. It also implies the average-case lower bounds of Coudron, Stark, and Vidick [38] and Le Gall [76].

In the next chapter, we go beyond AC^0 circuits using an *interactive* simulation task. There exist interactive tasks which a constant-depth quantum circuit can perform, but are hard (in an unconventional, but well-defined sense) for NC^1 and $\oplus\mathsf{L}$ to compute. The precise definition of interactive tasks is in Chapter 6, but informally our strongest result is as follows:[2]

Theorem 142. *There exists an interactive problem,* $\mathsf{wide\text{-}CliffSim}[2]$*, which can be solved by constant-depth quantum circuits. However, any (interactive) classical algorithm A solving* $\mathsf{wide\text{-}CliffSim}[2]$ *can be used to construct an oracle $\mathcal{O}_{\text{rev}}$ such that*

$$\oplus\mathsf{L} \subseteq \mathsf{BPL}^{\mathcal{O}_{\text{rev}}}.$$

For nearly all complexity classes $\mathcal{C}$, the construction of $\mathcal{O}_{\text{rev}}$ is such that $A \in \mathcal{C} \implies \mathcal{O}_{\text{rev}} \in \mathcal{C}$. In particular, if L implements the interactive oracles then we have, morally,

$$\oplus\mathsf{L} \subseteq \mathsf{BPL}^{\mathsf{L}} = \mathsf{BPL}.$$

The containment $\oplus\mathsf{L} \subseteq \mathsf{BPL}$ is considered unlikely.

Furthermore, this problem is essentially the 2D HLF problem split into two rounds to make it interactive. One way to interpret this result is for that any classical algorithm simulating constant-depth quantum circuits, if the algorithm can be made interactive then it can solve $\oplus\mathsf{L}$-hard problems.

Again, this result is postponed until Chapter 6, and we continue now with the high-level overview of the first result separating RelAC^0 and RelQNC^0.

[2] We also give a similar result with NC^1-hardness (a weaker class, but the result adds other desirable properties, e.g., error correction, and better locality).

5.1.1 High-level overview of QNC^0 vs. AC^0 result

We now describe the problems we study en route to proving Theorem 56 and give a high-level overview of the proof.

Theorem 56 is proved via a sequence of increasingly stronger results. We first introduce a problem we call the Parity Halving Problem (PHP). PHP is not in RelQNC^0, but it can be solved exactly by a $\mathsf{QNC}^0/\mathsf{qpoly}$ circuit, which is a QNC^0 circuit with quantum advice. Similar to randomized advice, a circuit class with quantum advice is allowed to start with any polynomial-size quantum state that is independent of the input, but can depend on the input length. For the Parity Halving Problem (and other problems introduced later), the quantum advice state is a very simple state called the cat state, which we denote by $|\text{🐱}_n\rangle := \frac{1}{\sqrt{2}}(|0^n\rangle + |1^n\rangle)$. We denote the subclass of $\mathsf{RelQNC}^0/\mathsf{qpoly}$ where the advice state is the cat state $\mathsf{RelQNC}^0/\text{🐱}$.

Here's a bird's eye view of our proof: Our first result establishes that PHP is in $\mathsf{QNC}^0/\text{🐱}$, but any nearly exponential-size AC^0 circuit only solves the problem with probability exponentially close to $1/2$. Next we define a new problem called the Relaxed Parity Halving Problem on a grid (Grid-RPHP), which is indeed in RelQNC^0, but any nearly exponential-size AC^0 circuit only solves the problem with probability exponentially close to $1/2$. We then define parallel versions of these two problems, which we call Parallel-PHP and Parallel Grid-RPHP. We show that Parallel-PHP $\in$ $\mathsf{RelQNC}^0/\mathsf{qpoly}$ and Parallel Grid-RPHP $\in \mathsf{RelQNC}^0$, but any nearly exponential-size AC^0 circuit only solves these problems with exponentially small probability. Finally we show that Parallel Grid-RPHP can be reduced to 2D HLF, and hence our lower bound applies to 2D HLF as well. We now describe these problems and our proof techniques in more detail.

Parity Halving Problem. In the Parity Halving Problem on n bits, which we denote by PHP_n, we are given an input string $x \in \{0,1\}^n$ promised to have even parity: i.e., the Hamming weight of x, denoted $|x|$, satisfies $|x| \equiv 0 \pmod 2$. The goal is to output a string $y \in \{0,1\}^n$ that satisfies

$$|y| \equiv |x|/2 \pmod 2.$$

In other words, the output string's Hamming weight (mod 2) is half of that of the input string. Note that $|x|/2$ is well defined above because $|x|$ is promised to be even. An alternate way of expressing this condition is that $|y| \equiv 0 \pmod 2$ if $|x| \equiv 0 \pmod 4$ and $|y| \equiv 1 \pmod 2$ if $|x| \equiv 2 \pmod 4$.

We show in Section 5.2 that PHP can be solved with certainty on every input by a simple depth-2 $\mathsf{QNC}^0/\text{🐱}$ circuit. A quantum circuit solving PHP_3 is shown in Figure 5-1. The circuit has one layer of controlled phase gates followed by Hadamard gates on the output qubits, followed by measurement.

Although the problem is easy for constant-depth quantum circuits, we show that even an exponential-size $\mathsf{AC}^0/\mathsf{rpoly}$ circuit cannot solve the problem on the uniform distribution (over valid inputs) with probability considerably better than $1/2$, which is trivially achieved by the circuit that outputs the all-zeros string on all inputs.

Theorem 57 (PHP). *The Parity Halving Problem* (PHP_n) *can be solved exactly by a* $\mathsf{QNC}^0/🐱$ *circuit. But on the uniform distribution over all valid inputs (even parity strings), any* $\mathsf{AC}^0/\mathsf{rpoly}$ *circuit of depth* d *and size at most* $\exp\bigl(n^{\frac{1}{10d}}\bigr)$ *only solves the problem with probability* $\frac{1}{2} + \exp(-n^{\alpha})$ *for some* $\alpha > 0$.

Note that the parameters of the AC^0 lower bound in this theorem are essentially optimal, since the parity function on n bits can be computed by a depth-d AC^0 circuit of size $\exp\bigl(n^{\frac{1}{d-1}}\bigr)$ [61, Theorem 2.2]. Once we can compute the parity of the input bits, it is easy to solve PHP.

Since the quantum circuit for PHP is simple, it is clear that the difficult part of Theorem 57 is the AC^0 circuit lower bound. One reason for this difficulty is that if we allowed the output string y in PHP to be of quadratic size, then there is a simple depth-1 NC^0 circuit that solves this problem! The circuit simply computes the AND of every pair of input bits and outputs this string of size $\binom{n}{2}$. A simple calculation shows that the Hamming weight of this string will be $\binom{|x|}{2}$, which satisfies the conditions of the problem. Hence to prove the AC^0 lower bound, the technique used has to be sensitive to the output size of the problem. However, traditional AC^0 lower bound techniques were developed for decision problems, and do not explicitly take the output size of the problem into account. Hence we modify some known techniques and establish the this lower bound in three steps.

First, we use Håstad's switching lemmas [61, 62], or more precisely a recent refinement of it due to Rossman [98]. However, directly using the result of Rossman off the shelf gives us a weaker result than Theorem 57; we are only able to establish the theorem with a quasi-polynomially small correlation instead of the exponentially small correlation in Theorem 57. To obtain the result we want, we refine Rossman's result to work better for multi-output functions (Lemma 70). This result is quite technical, but the conceptual ideas already appear in the works of Håstad [62] and Rossman [98]. Applying this switching lemma reduces the problem of proving an average-case AC^0 lower bound to that of showing an average-case NC^0 lower bound for a modified version of the Parity Halving Problem. This modified version of the problem is similar to PHP, except it has n inputs and slightly more (say, $n^{1.01}$) outputs.

Our second step is to use a combinatorial argument to reduce this question to

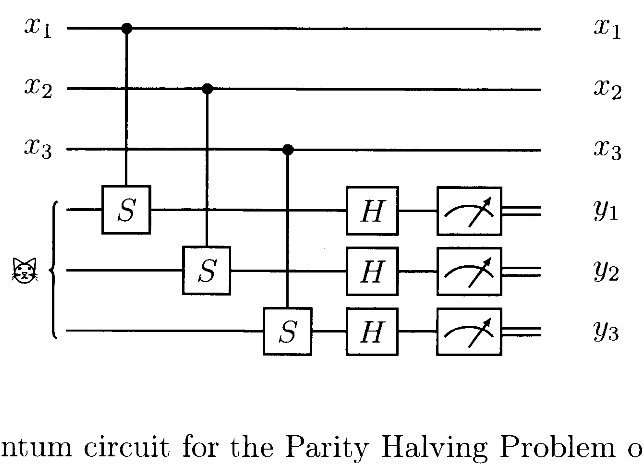


Figure 5-1: Quantum circuit for the Parity Halving Problem on 3 bits, PHP_3.

showing an average-case lower bound against NC^0 circuits with locality 1 (i.e., where each output only depends on a single input) for a further modified version of PHP.

The third and final step is to show that NC^0 circuits with locality 1 cannot solve this modifed PHP on a random input. We prove this by generalizing known lower bounds in the literature on quantum non-local games. Specifically we generalize lower bounds present in the work of Mermin [80], and Brassard, Broadbent, and Tapp [25].

This proof is presented in Section 5.2. We first prove the lower bound against NC^0 circuits of locality 1 in Section 5.2.2, then show the lower bound against general NC^0 circuits in Section 5.2.3, and finally introduce the switching lemma and conclude the proof of Theorem 57 in Section 5.2.4. The switching lemma itself is proved in Appendix A.

Now Theorem 57 is weaker than what we want (Theorem 56) in two ways. Aside from the fact that the lower bound is for a problem different from the 2D HLF problem, the problem in Theorem 57 is in $\mathsf{RelQNC}^0/$🐱 and not RelQNC^0, and the correlation lower bound is close to 1/2 instead of being exponentially small. We now tackle the first problem and get rid of the cat state.

Relaxed Parity Halving Problem. Since the cat state cannot be constructed by QNC^0 circuits (proved in Theorem 72), we have to modify the Parity Halving Problem to get by without a cat state.

Although we cannot create the cat state with QNC^0 circuits, we can construct a state we call a "poor man's cat state," which is the state

$$\frac{1}{\sqrt{2}}\left(|z\rangle + |\bar{z}\rangle\right),$$

where $z \in \{0,1\}^n$ is a bit string and $\bar{z}$ denotes its complement. When $z = 0^n$, this is indeed the cat state, but in general this is some entangled state that can be converted to the cat state by applying the $\mathtt{X}$ gate to some subset of the qubits.

Interestingly, we can create a poor man's cat state (with a uniformly random z) using QNC^0 circuits. Here is one simple construction. First arrange the n qubits on a line and set them all to be in the $|+\rangle$ state. Then in a separate set of $n-1$ qubits, compute the pairwise parities of adjacent qubits. In other words, we store the parity of qubit 1 and 2, 2 and 3, 3 and 4, and so on until qubit $n-1$ and n. And then we measure these $n-1$ qubits, and denote the measurement outcomes $d_1, \ldots, d_{n-1}$, which we will call the "difference string." It is easy to verify that if all the $d_i = 0$, then the resulting state is indeed the cat state. In general, for any $d \in \{0,1\}^{n-1}$, the resulting state is a poor man's cat state, and z can be determined from the string d up to the symmetry between z and $\bar{z}$. Since z and $\bar{z}$ are symmetric in the definition of the poor man's cat state, let us choose the convention that $z_1 = 0$. Now we can determine the remaining z_i from d using the fact that $d_1 = z_1 \oplus z_2 = z_2$, $d_2 = z_2 \oplus z_3$, and so on until $d_{n-1} = z_{n-1} \oplus z_n$. Note that because of this construction, some z_i depend on many bits of d_i. For example, z_n is the parity of all the bits in d.

This construction of the poor man's cat state easily generalizes to graphs other than the 1D line. We could place the n qubits on a balanced binary tree and measure the parity of all adjacent qubits, and hence get one d_i for every edge in the tree. If we call the root node $z_1 = 0$, then the value of any z_i will be the parity of all d_i on edges between vertex i and the root. In this case each z_i depends on at most $\log n$ bits of d_i. Similarly, we can choose a 2D grid instead of a balanced binary tree, and set the top left qubit to be $z_1 = 0$. Then each z_i will depend on at most $2\sqrt{n}$ bits of d. This grid construction is described more formally in Section 5.3.1.

Now there's an obvious strategy to try: Simply use a poor man's cat in our quantum circuit for PHP instead of using an actual cat state, and redefine the problem to match the output of this quantum circuit! So we simply run the circuit in Figure 5-1 on a poor man's cat state $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$ and see what the quantum circuit outputs. Unfortunately the output depends on z, but the poor man's cat state has been destroyed by the circuit and we do not have a copy of z around. But we do still have the string d from which it is possible to recover z, although this may not be computationally easy since a single bit of z may depend on a large number of bits of d. More subtly, a single bit of d may be involved in specifying many bits of z, which is also a problem for circuits without fan-out, such as QNC^0 circuits. Instead of trying to recover z, we can just modify the problem to include d as an output. The problem will now have two outputs, one original output y, and a second output string d, which is the difference string of the z in the poor man's cat state. This is the Relaxed Parity Halving Problem, which is more formally defined in Section 5.3.2.

More precisely, the Relaxed Parity Halving Problem, or RPHP, depends on the choice of the underlying graph, and is well defined for any graph. We choose the 2D grid to get a problem that reduces to 2D HLF.[3] We call this problem Grid-RPHP.

We show in Section 5.3.2 that Grid-RPHP can be solved by the 2D QNC^0 circuit we described, but even a nearly exponentially large AC^0 circuit cannot solve the problem with probability significantly larger than 1/2 on the uniform distribution over valid inputs.

Theorem 58 (Grid-RPHP). *Grid-RPHP$_n$ can be solved exactly by a QNC^0 circuit on a 2D grid. But on the uniform distribution over all valid inputs (even parity strings), any AC^0 circuit (or $\mathsf{AC}^0/\mathsf{rpoly}$ circuit) of depth d and size at most $\exp(n^{1/10d})$ can solve the problem with probability at most $\frac{1}{2} + \exp(-n^{\alpha})$ for some $\alpha > 0$.*

Note that just like Theorem 57, the lower bound here is essentially optimal, since the parity function itself can be computed by a depth-d AC^0 circuit of size $\exp\left(n^{\frac{1}{d-1}}\right)$ [61, Theorem 2.2]. Our separation essentially works for any graph with sublinear diameter, such as the grid or the balanced binary tree, but not the 1D line. In fact, when the underlying graph is the 1D line, RPHP becomes easy to solve, even for NC^0 circuits.[4]

We prove Theorem 58 by showing a reduction from the Parity Halving Problem with input size n and output size $O(n^{3/2})$ to Grid-RPHP. This version of PHP is

[3] Picking the balanced binary tree would give better parameters, but qualitatively similar results. We choose the 2D grid so that our problem can be solved by a constant-depth quantum circuit acting on qubits laid out in 2D.

[4] One can output $y = 0^n$ and $d_i = x_i$ for all $i \in \{1, \dots, n-1\}$ to solve the Relaxed Parity Halving Problem on the 1D line.

indeed hard for AC^0 circuits and this result follows from the work done in Section 5.2. This reduction and theorem are proved formally in Section 5.3.2.

Now Theorem 58 is still weaker than what we want (Theorem 56). The correlation lower bound is still close to 1/2 and not exponentially small. We now fix this issue using a simple idea.

Parallel Grid-RPHP. Let Parallel Grid-RPHP be the problem where we are given many instances of Grid-RPHP in parallel and are required to solve all of them correctly. For this problem the quantum circuit is obvious: Simply use the quantum circuit for Grid-RPHP for each instance of the problem. Clearly if the quantum circuit solves each instance correctly, it solves all of them correctly. But since a classical circuit only solves an instance with some probability close to 1/2, we expect that solving many copies of the problem gets much harder.

Theorem 59 (Parallel Grid-RPHP). Parallel Grid-RPHP$_n$ *can be solved exactly by a* QNC^0 *circuit on a 2D grid. But on the uniform distribution over all valid inputs (even parity strings for each instance of* Grid-RPHP*), any* AC^0 *circuit (or* $\mathsf{AC}^0/\mathsf{rpoly}$ *circuit) of depth d and size at most* $\exp(n^{1/10d})$ *can solve the problem with probability at most* $\exp(-n^{\alpha})$ *for some* $\alpha > 0$.

As before, the difficult part of Theorem 59 is proving the classical lower bound. While it seems intuitive that repeating the problem (in parallel) several times reduces the success probability, similarly intuitive statements can be false or difficult to prove [95]. More precisely, what we need is a direct product theorem, which also may not hold in some models of computation [102].

We consider the parallel version of the standard PHP, denoted Parallel-PHP, and reduce Parallel-PHP to Parallel Grid-RPHP as above. We then establish a lower bound for Parallel-PHP by using Vazirani's XOR lemma [116]. Vazirani's XOR lemma is an intuitive statement about how a probability distribution that is "balanced" in a certain sense must be close to the uniform distribution. The implication for our problem is the following: To understand the probability that a circuit solves all the instances of PHP in Parallel-PHP, or equivalently that it fails to solve 0 instances, it is enough to understand the probability that it fails to solve an even number of instances. This task turns out to be similar to the original PHP with larger input and output size, but with some additional constraints on the input. The techniques we have developed allow us to upper bound this probability, and hence (using the XOR lemma) upper bound the probability that a circuit solves all instances correctly.

Now we are almost done, since Theorem 59 looks very similar to Theorem 56, except that the hardness is shown for Parallel Grid-RPHP and not the 2D HLF problem.

Reduction to the Hidden Linear Function problem. The final step of our program is carried out in Section 5.5. First we show via a simple reduction in Theorem 87 that the Relaxed Parity Halving Problem (for any graph G) can be reduced to the Hidden Linear Function problem (not necessarily the 2D HLF). In particular, our reduction reduces Grid-RPHP reduces to the 2D HLF problem, as we describe in

Corollary 90. So far this shows that one instance of Grid-RPHP reduces to the 2D HLF problem. We then show, in Lemma 89, that we can embed multiple instances of 2D HLF in parallel into one instance of 2D HLF. Hence Parallel Grid-RPHP reduces to 2D HLF as well, and hence Theorem 59 implies Theorem 56.

5.1.2 Additional results

We also consider the question of showing a separation between RelQNC^0 and an even larger class, $\mathsf{RelAC}^0[2]$, where recall $\mathsf{AC}^0[2]$ is AC^0 with unbounded fan-in PARITY gates. We implement the first two steps of the strategy above, where we come up with a problem in $\mathsf{RelQNC}^0/$🐱 that cannot be solved by an $\mathsf{AC}^0[2]$ circuit, even on a $o(1)$ fraction of the inputs. Unfortunately, we do not know how to remove the reliance on the cat state in this setting, so we only get the separation $\mathsf{RelQNC}^0/$🐱 $\not\subseteq \mathsf{RelAC}^0[2]$.

Parity Bending Problem. In the Parity Bending Problem, which we denote PBP_n, we are given a string $x \in \{0,1\}^n$, and our goal is to output a string $y \in \{0,1\}^n$ such that if $|x| \equiv 0 \pmod 3$ then $|y| \equiv 0 \pmod 2$, and if $|x| \in \{1,2\} \pmod 3$ then $|y| \equiv 1 \pmod 2$. Our main result is Theorem 60, which says that PBP can be solved with high probability by a $\mathsf{QNC}^0/$🐱 circuit, but needs exponential-size $\mathsf{AC}^0[2]$ circuits to solve with probability significantly greater than half.

Theorem 60 (PBP). *There exists a $\mathsf{QNC}^0/$🐱 circuit that solves the Parity Bending Problem (PBP_n) on any input with probability $\geq \frac{3}{4}$. But there exists an input distribution on which any $\mathsf{AC}^0[2]/\mathsf{rpoly}$ of depth d and size at most $\exp\left(n^{\frac{1}{10d}}\right)$ only solves the problem with probability $\frac{1}{2} + \frac{1}{n^{\Omega(1)}}$.*

As with the Parity Halving Problem, the quantum circuit that solves this problem with bounded error is very simple as shown in Section 5.6. For this problem, the classical lower bound is easier to show than before, and follows from the work of Razborov and Smolensky [97, 108], which shows that $\mathsf{AC}^0[2]$ circuits correlate poorly with the Mod 3 function.

As before, we can strengthen the separation to make the quantum circuit's success probability arbitrarily close to 1 and the classical circuit's success probability arbitrarily close to 0 by defining a new version of the Parity Bending Problem that we call the Parallel Parity Bending Problem. In this problem, we are given many instances of the Parity Bending Problem, and required to solve at least 2/3 of them. Since $\mathsf{QNC}^0/$🐱 can solve this problem with probability 3/4, it can solve more than 2/3 of the instances with high probability.

Theorem 61 (Parallel PBP). *The Parallel Parity Bending Problem can be solved with probability $1 - o(1)$ by a $\mathsf{QNC}^0/\mathsf{qpoly}$ circuit, but any $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit can only solve the problem with probability $\frac{1}{n^{\Omega(1)}}$.*

At a high level this lower bound proceeds similar to Theorem 59, again employing Vazirani's XOR lemma [116], but there are technical difficulties caused by the fact that the Boolean version of the Mod 3 function is unbalanced and easy to compute on a 2/3 fraction of the inputs.

5.1.3 Discussion and open problems

Our main results show that there is a search problem (either the 2D HLF problem or the Parallel Grid-RPHP) in RelQNC^0 that is not in RelAC^0, and that there is a search problem (Parallel PBP) in RelQNC^0/🐱 that is not in $\mathsf{RelAC}^0[2]$. One open problem is to generalize both separations and show that there is a search problem in RelQNC^0 that is not in $\mathsf{RelAC}^0[2]$, or more generally $\mathsf{RelAC}^0[p]$ for any prime p. This is essentially the frontier of circuit lower bounds, and it will be difficult to go further without radically new techniques.

One could try to achieve a quantum advantage using even weaker classes than RelQNC^0 or classes incomparable to RelQNC^0. For inspiration, the recent result of Raz and Tal [96] exhibits a decision problem in $\mathsf{BQLOGTIME}$ (bounded-error quantum logarithmic time) that is not in AC^0.

5.2 Parity Halving Problem

Recall the Parity Halving Problem from the introduction. We now define a more general version of the problem with n input bits and m output bits.

Problem 62 (Parity Halving Problem, $\mathrm{PHP}_{n,m}$). *Given an input $x \in \{0,1\}^n$ of even parity, output a string $y \in \{0,1\}^m$ such that*

$$|y| \equiv \frac{1}{2}|x| \pmod 2.$$

Alternately, y must have even parity if $|x| \equiv 0 \pmod 4$ and odd parity if $|x| \equiv 2 \pmod 4$. We also define PHP_n to be $\mathrm{PHP}_{n,n}$.

The main result of this section is to show this problem is in QNC^0/🐱, but not $\mathsf{AC}^0/\mathsf{rpoly}$. We now restate this result (Theorem 57) more formally:

Theorem 57. *The Parity Halving Problem (PHP_n) can be solved exactly by a depth-2, linear-size quantum circuit starting with the $|🐱_n\rangle$ state. But on the uniform distribution over all valid inputs (even parity strings), any $\mathsf{AC}^0/\mathsf{rpoly}$ circuit of depth d and size $s \le \exp\left(n^{\frac{1}{2d}}\right)$ only solves the problem with probability*

$$\frac{1}{2} + \exp\left(-n^{1-o(1)}/O(\log s)^{2(d-1)}\right).$$

We prove this theorem in several parts. First we prove the quantum upper bound in Section 5.2.1 (Theorem 63). The lower bound on AC^0 circuits via a sequence of incrementally stronger lower bounds, culminating in the claimed lower bound. We start in Section 5.2.2 by showing a lower bound (Theorem 64) for a very simple class of circuits, NC^0 circuits of locality 1, i.e., NC^0 circuits where every output is an arbitrary function of exactly one input bit. We then extend the lower bound to arbitrary NC^0 circuits in Section 5.2.3 (Theorem 66), and to AC^0 circuits in Section 5.2.4 culminating in the AC^0 lower bound for $\mathrm{PHP}_{n,m}$ in Theorem 71, from which the lower bound in Theorem 57 follows straightforwardly by setting $m = n$.

5.2.1 Quantum upper bound

Before we get into the details of the proof, let us motivate the problem. Observe that the problem naturally defines an interesting n-player cooperative non-local game, which we call the Parity Halving Game. In this game, there are n players, and each player gets one of the n input bits and outputs a single bit, with no communication with the other players. The input and output conditions are the same as in PHP_n: The input is promised to be of even Hamming weight, and the players win the game if their output's parity satisfies the condition in Problem 62.

Because the players are not allowed to communicate, the strategies permitted in the non-local game are far more restricted than an AC^0 circuit or even an NC^0 circuit for PHP_n since each output bit is only allowed to depend on one input bit. We will call this model NC^0 with locality 1.

Now that we have defined a game, we can study the probability of success for classical players versus the probability of success for quantum players who share entanglement before the game begins. In fact, when $n = 3$, the Parity Halving Game coincides with the well-known Greenberger–Horne–Zeilinger (GHZ) game [55]. It is known that quantum players sharing entanglement, and specifically the state $|\text{🐱}_3\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, can always win the GHZ game with certainty, but classical players can win the GHZ game with probability at most $3/4$.

This n-player generalization of the GHZ game is very natural and quantum players can win the Parity Halving Game exactly using a $|\text{🐱}_n\rangle$ state. This game has been studied before, and we are aware of two other works that analyze this game: the first by Mermin [80], and the second by Brassard, Broadbent, and Tapp [25]. Both papers exhibit the quantum strategy that wins perfectly and argue that classical strategies fail (as we do in the next section).

The strategy for winning the 3-player GHZ game generalizes to yield a perfect strategy for winning the n-player game as well, which yields a depth-2 linear-size quantum circuit for PHP_n. We now describe the quantum strategy and the corresponding constant-depth quantum circuit.

Theorem 63 (Quantum circuit for PHP_n). *The Parity Halving Problem* (PHP_n) *can be solved exactly by a depth-2, linear-size quantum circuit starting with the* $\left|\text{🐱}_n\right\rangle$ *state.*

Proof. We describe this circuit in the language of the n-player Parity Halving Game described above. The circuit is depicted in Figure 5-1 (on page 115). Let the input to the i^{th} player in the Parity Halving Game be called x_i, and their output be called y_i. In our protocol, the players will share an n-qubit cat state $\left|\text{🐱}_n\right\rangle = \frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$, and each player receives one qubit of the cat state at the beginning.

Each player starts by applying a phase gate, $S = \left(\begin{smallmatrix}1&0\\0&i\end{smallmatrix}\right)$, to their qubit of the cat state if their input bit is 1. If their input bit is 0, they do nothing. In other words, the player applies a control-S gate with x_i as the source and their qubit of the cat state as the target. After this step, the cat state has been transformed to

$$\frac{1}{\sqrt{2}}\left(|0^n\rangle + i^{|x|}\,|1^n\rangle\right).$$

But since x has even parity, this state is either $|\text{🐱}_n\rangle$ or $\frac{1}{\sqrt{2}}(|0^n\rangle - |1^n\rangle)$, the "minus cat state". We will denote this state by $\mathsf{Z}|\text{🐱}_n\rangle$ since this is the state one obtains by applying the $\mathsf{Z} = \left(\begin{smallmatrix}1 & 0\\ 0 & -1\end{smallmatrix}\right)$ gate to any one qubit of the cat state. When $|x| \equiv 0 \pmod 4$, this state will be $|\text{🐱}_n\rangle$ and when $|x| \equiv 2 \pmod 4$, this will be $\mathsf{Z}|\text{🐱}_n\rangle$. Note that $|\text{🐱}_n\rangle$ and $\mathsf{Z}|\text{🐱}_n\rangle$ are orthogonal states.

Finally, each player applies the Hadamard gate $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right)$ to their qubit of the cat state, measures the qubit, and outputs that as y_i. The operator $H^{\otimes n}$ maps the cat state $|\text{🐱}_n\rangle$ to a uniform superposition over even parity strings, and maps $\mathsf{Z}|\text{🐱}_n\rangle$ to a uniform superposition over odd parity strings. This follows from the following equations:

$$H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle, \qquad \text{and} \qquad H^{\otimes n}|1^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{|x|}|x\rangle. \tag{5.1}$$

Thus, when the players measure their qubits, they will get either a random even parity string when $|x| \equiv 0 \pmod 4$ or a random odd parity string when $|x| \equiv 2 \pmod 4$, as desired. □

Note that the idea of inducing a relative phase proportional to the Hamming weight of a string is studied more generally and called "rotation by Hamming weight" in [64].

5.2.2 Lower bound for NC^0 circuits of locality 1

We now discuss the success probability of classical strategies for the Parity Halving Game. This was already studied by Mermin [80], and Brassard, Broadbent, and Tapp [25]. Both papers argue that classical strategies only succeed with probability exponentially close to 1/2 on the uniform distribution over even-parity inputs.

We reprove these lower bounds on the Parity Halving Game and also prove lower bounds for a restricted version of the game. In the restricted version of the game we only consider inputs consistent with some restriction of the input bits, i.e., where the values of some input bits have been fixed and are known to all the players, and we only consider all even-parity inputs consistent with this fixing of input bits. We need this generalization later on in the proof since some input bits will be fixed by a random restriction in the AC^0 lower bound argument.

Theorem 64 (Classical lower bound for Parity Halving Game). *On the uniform distribution over even-parity strings, the success probability of any classical strategy for the Parity Halving Game with n players is at most $\frac{1}{2} + 2^{-\lceil n/2 \rceil}$.*

Now consider the restricted *Parity Halving Game with n players, where d of the input bits have fixed values known to all players. On the uniform distribution over even-parity strings consistent with the fixed input bits, the success probability of any classical strategy is at most $\frac{1}{2} + 2^{-\lceil (n-d)/2 \rceil}$.*

Proof. We start with the lower bound for the unrestricted Parity Halving Game. Since we consider classical strategies against a fixed input distribution, we can without loss of generality only consider deterministic strategies. This is because a randomized strategy is simply a probability distribution over deterministic strategies, and we can pick the strategy that does the best against the chosen input distribution. (This is the easy direction of Yao's minimax principle.)

Since each player only has one input bit x_i, and one output bit y_i, there are only four deterministic strategies: output $y_i = 0$, $y_i = 1$, $y_i = x_i$, or $y_i = x_i \oplus 1$. In any case, each y_i is a degree-1 polynomial (over $\mathbb{F}_2$) in x_i. It follows that the parity of the outputs, $\bigoplus_{i=1}^n y_i$, can be expressed as multivariate linear polynomial in $x_1, \dots, x_n$, say $a + b \cdot x$ for some $a \in \mathbb{F}_2$ and $b \in \mathbb{F}_2^n$. We want to upper bound the success probability of any such strategy.

Now consider the function $f(x) = \mathrm{Re}(i^{|x|})$. We have

$$f(x) = \begin{cases} 1 & \text{if } |x| \equiv 0 \pmod 4 \\ -1 & \text{if } |x| \equiv 2 \pmod 4 \\ 0 & \text{otherwise.} \end{cases}$$

The function $f(x)$ matches the parity of the output bits (as ±1) of the PHP_n function on an input x. More precisely, $f(x)$ gives the correct parity (as ±1) when x satisfies the promise of PHP_n, and evaluates to 0 for inputs outside the promise.

It follows that the product $(-1)^{a+b\cdot x} f(x)$ is 1 if the strategy corresponding to $a + b \cdot x$ is correct, -1 if it is incorrect, and 0 on inputs that are outside the promise. We define the correlation χ of a classical strategy as the absolute value of the fraction of valid inputs on which it is correct minus the fraction of valid inputs on which it is incorrect. We can compute this quantity as follows:

$$\chi = \left| \mathop{\mathbb{E}}_{x \in \mathbb{F}_2^n : \sum_i x_i = 0} \left[(-1)^{a+b\cdot x} f(x) \right] \right| \tag{5.2}$$

$$= \left| \frac{1}{2^{n-1}} \sum_{x \in \mathbb{F}_2^n} (-1)^{a+b\cdot x} \, \mathrm{Re}(i^{|x|}) \right| \tag{5.3}$$

$$\leq \frac{1}{2^{n-1}} \left| \mathrm{Re} \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{b_1 x_1 + \cdots + b_n x_n} \cdot i^{x_1 + \cdots + x_n} \right) \right| \tag{5.4}$$

$$= \frac{1}{2^{n-1}} \left| \mathrm{Re} \left(\sum_{x_1 \in \mathbb{F}_2} (-1)^{b_1 x_1} i^{x_1} \cdots \sum_{x_n \in \mathbb{F}_2} (-1)^{b_n x_n} i^{x_n} \right) \right| \tag{5.5}$$

$$= \frac{1}{2^{n-1}} \left| \mathrm{Re} \left((1 + i^{1+2b_1}) \cdots (1 + i^{1+2b_n}) \right) \right|. \tag{5.6}$$

That is, we want to know the real part of a product of n terms, each of which is $1 \pm i$. Since $1 \pm i$ is $\sqrt{2}$ times a primitive eighth root of unity, the product is $2^{n/2}$ times an eighth root of unity. After factoring out the $\sqrt{2}$ from each term, we have to determine the possible values of the product of n numbers of the form $\frac{1}{\sqrt{2}}(1 \pm i)$. When n is even, their product must lie in the set $\{\pm 1, \pm i\}$, and when n is odd it must lie in the set $\frac{\pm 1 \pm i}{\sqrt{2}}$. In both cases, we see that the real part of the product is either 0 or $\pm 2^{\lfloor n/2 \rfloor}$, so the correlation is $\chi = 0$ or $\chi = 2^{-\lceil n/2 \rceil + 1}$. Since the success probability is $(1 + \chi)/2$, this proves the first part of the theorem.

Now let us move on the to restricted version of the game and fix some of the inputs. If some individual bit x_j is restricted, then the term $\sum_{x_j \in \mathbb{F}_2}(-1)^{b_j x_j} i^{x_j}$ in the analysis above becomes either 1 or $(-1)^{b_j} i$. This term is a fourth root of unity, so it does not contribute to the magnitude of the product, since the fourth roots of unity have magnitude 1. Furthermore, it does not change the set of potential phases, since both the sets above are invariant under multiplication by a fourth root of unity. Since the constraint also halves the number of possible inputs, the effect on the correlation is the same as just removing that bit. In other words, χ is at most $2^{-\lceil (n-d)/2 \rceil + 1}$. It follows that the success probability of a classical strategy is

$$\frac{1+\chi}{2} = \frac{1}{2} + 2^{-\lceil (n-d)/2 \rceil}. \qquad \square$$

It is interesting to note that for the unrestricted game, Brassard, Broadbent, and Tapp [25] show that there are strategies matching this upper bound.

5.2.3 From NC^0 circuits of locality 1 to general NC^0 circuits

We can view NC^0 circuits as a more powerful model of computation than the game considered in the previous section. Now each player is allowed to look at the input bits of a constant number of other players before deciding what to output. For example, the players could band together into constant-sized groups and look at all the other bits in the group to make a slightly more informed choice. However, intuitively it seems that the players cannot do much better than before. We will show this formally by proving that NC^0 circuits cannot solve PHP_n.

First, we define some terms. Fix a circuit C and define the *interaction graph* of the circuit C to be a bipartite graph on the input bits and output bits where there is an edge from an input bit x_i to an output bit y_j if there is a path from x_i to y_j in the circuit C (i.e., if x_i can affect y_j in C). The neighborhood of a vertex in this graph is sometimes called its *light cone*. That is, the light cone of an output bit, $LC(y_i)$, is the set of input bits which can affect it, and the light cone of an input bit, $LC(x_i)$ is the set of output bits which it can affect. For example, if all gates have fan-in 2, then the light cone of any output bit in a circuit of depth d is of size at most 2^d. In general, we say that a circuit C has locality ℓ if the light cone of any output bit is of size at most ℓ.

Note that while the fan-in of gates sets an upper bound on the light cone of an output bit, the fan-out sets an upper bound for the light cone of input bits. In all the classical circuit classes we study in this chapter, fan-out is unbounded, hence even in a constant-depth circuit one input bit can affect all output bits.

Proposition 65. *Let C be a circuit with n inputs, m outputs, and locality ℓ. There exists a subset of inputs bits S of size $\Omega\left(\min\left\{n, \frac{n^2}{\ell^2 m}\right\}\right)$ such that each output bit depends on at most one bit from S.*

Proof. Since each output bit has a light cone of size at most ℓ, the interaction graph has at most ℓm edges. This implies that, on average, an input bit has a light cone of size $\frac{\ell m}{n}$. Our goal is to find a set of input bits S such that their light cones are pairwise disjoint, since then the light cone of any output contains at most one element of S.

Consider the intersection graph between input variables. That is, we consider the graph on $x_1, \ldots, x_n$, where x_i is connected to x_j if their light cones intersect. A variable x_i that had degree d in the original graph has degree at most $d\ell$ in the intersection graph, since each output vertex has locality ℓ. Hence the average degree in the intersection graph, denoted by D, is at most $\frac{\ell^2 m}{n}$. By Turán's theorem, in any graph on n vertices with average degree at most D there exists an independent set of size at least $n/(1+D)$. Thus, we get a set $S \subseteq \{x_1, \ldots, x_n\}$ of size $\Omega\left(\min\left\{n, \frac{n^2}{\ell^2 m}\right\}\right)$ such that the light cones of every pair of input bits in S do not intersect. □

We are now ready to prove a lower bound on NC^0 circuits of locality ℓ solving $\mathrm{PHP}_{n,m}$.

Theorem 66 (PHP is not in RelNC^0). *Let C be an NC^0 circuit with n inputs, m outputs, and locality ℓ. Then C solves $\mathrm{PHP}_{n,m}$ on a random even-parity input with probability at most $\frac{1}{2} + 2^{-\Omega\left(\min\left\{n, \frac{n^2}{\ell^2 m}\right\}\right)}$.*

Proof. Let the circuit C solve $\mathrm{PHP}_{n,m}$ on a random even-parity input with probability p. By the previous theorem, there is a set of input bits with disjoint light cones, S, and $|S| = \Omega\left(\min\left\{n, \frac{n^2}{\ell^2 m}\right\}\right)$. For the remainder of this proof fix any such S.

Now consider choosing an arbitrary assignment for the bits outside S and running the circuit C on the distribution of random even-parity strings consistent with this arbitrary assignment. The probability of success of circuit C may depend on the arbitrary assignment chosen, but since the success probability for a random choice is p, there exists one assignment for which the success probability is at least p. Let us fix this assignment of bits outside S. Now we have an assignment for bits outside S such that C is correct with probability at least p on a random even-parity input consistent with this assignment.

We will now argue that the circuit gives a strategy for the restricted Parity Halving Game on n players with $n - |S|$ restricted bits with probability of success at least p. To do so, we assign a player for every input bit. Only the players assigned to bits in S will have unrestricted inputs. Since the light cones of bits in S do not intersect, a player with input bit in S can compute the values of all outputs in its light cone (since all the bits outside S are fixed and known to everyone). This player can now output the parity of all these output bits. Some output bits may not appear in any input light cone; we add the parity of these bits to an arbitrary player's output. Now the the parity of the players' outputs is the same as the parity of the circuit's output. This gives a classical strategy for the restricted Parity Halving Game with n players and $n - |S|$ restricted bits with success probability at least p. Finally, from Theorem 64 we get that $p \leq \frac{1}{2} + 2^{-\Omega\left(\min\left\{n, \frac{n^2}{\ell^2 m}\right\}\right)}$. □

Note that this theorem is essentially tight. It says that to achieve a high probability of success, we need $n^2 = \Theta(\ell^2 m)$. We can indeed achieve success probability 1 at both extremes: when $m = \Theta(n^2)$ and $\ell = 2$, or when $m = 1$ and $\ell = n$. For the first setting of parameters, as noted in the introduction, there is a simple depth-1 NC^0 circuit of locality 2 that solves the problem when $m = \binom{n}{2}$. The second parameter regime is even simpler, since any Boolean function can be computed by an NC^0 circuit of locality $\ell = n$.

5.2.4 From NC^0 circuits to AC^0 circuits

In this section we finally extend our lower bound to AC^0 circuits as stated in Theorem 57.

To do this, we use a technical tool known as a switching lemma [49, 6, 122, 61]. Informally, a switching lemma says that with high probability randomly restricting a large fraction of the input bits to an AC^0 circuit produces a circuit with small locality.

Average-case reductions from NC^0 to AC^0 have previously appeared in the literature (cf. [117]), based on the original switching lemma [61]. In this chapter, we will use multi-switching lemmas, which handle multiple output circuits much better, and were recently proved by Håstad [62] and Rossman [98]. See Appendix A. Using the multi-switching lemmas instead of Håstad's original switching lemma [61] allows us to improve the parameters dramatically.[5]

Preliminaries

We start with some definitions. In the following, we consider restrictions and random restrictions. A restriction $\rho \in \{0, 1, *\}^n$ defines a partial assignment to the inputs of a Boolean string of length n. For $i = 1, \ldots, n$, when $\rho_i \in \{0, 1\}$ we say that the restriction fixes the value of the i-th coordinate, and when $\rho_i = *$ we say that the restriction keeps the i-th coordinate alive.

A p-random restriction is a restriction sampled according to the following process: for each $i = 1, \ldots, n$ independently, sample $\rho_i = *$ with probability p, $\rho_i = 0$ with probability $(1 - p)/2$ and $\rho_i = 1$ with probability $(1 - p)/2$. We denote by $\mathbf{R}_p$ the distribution of p-random restrictions.

For a Boolean function $f : \{0, 1\}^n \to \{0, 1\}^m$ we denote by $f|_\rho : \{0, 1\}^n \to \{0, 1\}^m$ the restricted function defined by

$$f|_\rho(x) = f(y) \qquad \text{where} \qquad y_i = \begin{cases} x_i & \rho_i = * \text{ and} \\ \rho_i & \text{otherwise.} \end{cases}$$

Next, we give the standard definition of a decision tree. For an excellent survey on this topic, please see [30].

Definition 67 (Decision Tree). A **decision tree** is a rooted ordered binary tree T, where each internal node of T is labeled with a variable x_i and each leaf is labeled

[5] Based on the original switching lemma, we can show that PHP is hard to compute by AC^0 circuits on more than $1/2 + 1/n^{\Omega(\log n)}$ of the inputs. On the other hand, based on the multi-switching lemmas, we will show that PHP is, in fact, hard to compute on more than $1/2 + \exp\left(-n^{1-o(1)}\right)$ of the inputs.

with a value 0 or 1. Given an input $x \in \{0,1\}^n$, the tree is evaluated as follows. Start at the root. If this is a leaf then stop. Otherwise, query the variable x_i that labels the root. If $x_i = 0$, then recursively evaluate the left subtree, if $x_i = 1$ then recursively evaluate the right subtree. The output of the tree is the value (0 or 1) of the leaf that is reached eventually. Note that an input x deterministically determines the leaf reached at the end, and thus the output. We say a decision tree computes f if its output equals $f(x)$, for all $x \in \{0,1\}^n$. The complexity of such a tree is its depth, i.e., the number of queries made on the worst-case input. We denote by $\mathrm{DT}(t)$ the class of functions computed by decision trees of depth at most t.

Note that the decision tree complexity of a function f is also called the deterministic query complexity of f.

Definition 68 ($\mathcal{F}$-Decision Tree). Suppose $\mathcal{F}$ is a class of functions mapping $\{0,1\}^n$ to $\{0,1\}^m$. An $\mathcal{F}$-partial decision tree is a standard decision tree, except that the leaves are marked with functions in $\mathcal{F}$ (instead of constants). Given an input $x \in \{0,1\}^n$, the $\mathcal{F}$-Decision Tree is evaluated as follows. Starting from the tree's root, we go along the path defined by the input x until we reach a leaf. Then, we evaluate the function $f_v \in \mathcal{F}$ that labels the leaf v on the input x, and output its value, $f_v(x)$. We denote by $\mathrm{DT}(t) \circ \mathcal{F}$ the class of functions computed by $\mathcal{F}$-decision trees of depth at most t.

Note that $\mathcal{F}$-decision trees compute functions from $\{0,1\}^n \to \{0,1\}^m$ where n and m are the input and output lengths for the functions in $\mathcal{F}$, respectively.

Definition 69 (Tuples of functions classes). Suppose $\mathcal{F}$ is a class of functions mapping $\{0,1\}^n$ to $\{0,1\}$. We denote by $\mathcal{F}^m$ the class of functions $F : \{0,1\}^n \to \{0,1\}^m$ of the form $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$, where each $f_i \in \mathcal{F}$. That is, $\mathcal{F}^m$ is the class of m-tuples of functions in $\mathcal{F}$.

The multi-switching lemma

The main lemma that we are going to use is a slight adaption of Rossman's lemma [98], which combines both switching lemmas of Håstad [61, 62]. The lemma claims that a multi-output AC^0 circuit mapping $\{0,1\}^n \to \{0,1\}^m$ would reduce under a random restriction, with high probability, to a function in the class $\mathrm{DT}(2t) \circ \mathrm{DT}(q)^m$ (for some parameters t and q).

Let us pause for a second to spell out what is the class $\mathrm{DT}(2t) \circ \mathrm{DT}(q)^m$. This is the class of depth-$2t$ decision trees, whose leaves are labeled by m-tuples of depth-q decision trees, one per output bit. In other words, these are functions mapping $\{0,1\}^n$ to $\{0,1\}^m$ that can be evaluated by adaptively querying at most $2t$ coordinates globally, after which each of the m output bits can be evaluated by making at most q additional adaptive queries. Note that while the first $2t$ queries are global, the last q queries could differ from one output bit to another. We would typically set the parameters so that t is much larger than q (for example, $t = n^{1-o(1)}$ and $q = o(\log n)$).

Lemma 70 (Multi-switching lemma). *Let $f : \{0,1\}^n \to \{0,1\}^m$ be an AC^0 circuit of size s, depth d. Let $q \in \mathbb{N}$ be a parameter, and set $p = 1/(m^{1/q} \cdot O(\log s)^{d-1})$. Then*

$$\forall t : \Pr_{\rho\sim\mathbf{R}_p}[f|_\rho \notin \mathrm{DT}(2t) \circ \mathrm{DT}(q)^m] \leq s \cdot 2^{-t}.$$

We defer the proof of Lemma 70 to Appendix A as this is an adaptation of Rossman's lemma [98].

We would use the lemma as follows. First, we apply a p-random restriction that reduces the AC^0 circuit to a $\mathrm{DT}(2t) \circ \mathrm{DT}(q)^m$ function with high probability. Then, we further query at most $2t$ coordinates, and fix their values, by following a path in the common partial decision tree. After which, the restricted function would be an m-tuple of depth-q decision trees. Then, using the simple fact that a depth-q decision tree is a function with locality at most 2^q, we reduced an AC^0 circuit to an NC^0 circuit with locality at most 2^q with high probability.

On the choice of parameters. We have the freedom to choose q and t when applying Lemma 70 in Theorem 71. First, we discuss the choice of q. We would like the lemma to yield on one hand an NC^0 circuit with small locality, and on the other hand to keep many input variables alive. To get small locality, q should be small, say $q = o(\log n)$. To keep many variables alive, pn should be large, and since $p = 1/O(m^{1/q}(\log s)^{d-1})$, we would like q to be large, say $q = \omega(1)$. Balancing these two requirements leads to the choice $q = \Theta(\sqrt{\log n})$.

Once q is set, we would like to make t as large as possible, as it controls the failure probability in Lemma 70, but on the same time we want the number of alive variables after the two-step restriction process above to remain high. Since this number is roughly $pn - t$ we would choose t to be a small constant fraction of pn (which is $n^{1-o(1)}$). With these choices, we would be left with at least $\Omega(pn)$ variables alive and locality at most 2^q with extremely high probability.

AC^0 lower bound

Theorem 71 (PHP is not in RelAC^0). *Let $n \leq m \leq n^2$. Any $\mathsf{AC}^0/\mathsf{rpoly}$ circuit F of depth d and size $s \leq \exp\left(n^{\frac{1}{2d}}\right)$ solves $\mathrm{PHP}_{n,m}$ on the uniform distribution over valid inputs (even parity strings) with probability at most*

$$\frac{1}{2} + \exp\left(-n^2/\left(m^{1+o(1)} \cdot O(\log s)^{2(d-1)}\right)\right).$$

Proof. Since we have a fixed input distribution, we can without loss of generality prove the lower bound against an AC^0 circuit (instead of an $\mathsf{AC}^0/\mathsf{rpoly}$ circuit), since an $\mathsf{AC}^0/\mathsf{rpoly}$ circuit defines a distribution over AC^0 circuits and we can simply pick the one that does the best against our input distribution.

Suppose that F solves the Parity Halving Problem on a random even-parity input with probability $\frac{1}{2} + \varepsilon$. We shall show that $\varepsilon \leq \exp(-n^2/(m^{1+o(1)} \cdot O(\log s)^{2(d-1)}))$.

We defer the choice of q for later, to optimize the parameters. However, we will use the fact that $q = o(\log m)$ and that $q = \omega(1)$. We set

$$p = 1/(m^{1/q} \cdot O(\log s)^{d-1}), \qquad t = pn/8.$$

Note that under this choice of parameters $s \le 2^{t/2}$, by the following calculation: using the assumption that $s < \exp(n^{1/2d})$ twice, we have

$$2^{t/2} = \exp(\Omega(pn)) = \exp\big(\Omega(m^{-o(1)} \cdot n^{(d+1)/2d})\big) \gg \exp\big(n^{1/2d}\big) > s.$$

Let ρ be a p-random restriction. Denote by $\mathcal{E}$ the event that:

1. $F|_\rho \in \mathrm{DT}(2t) \circ \mathrm{DT}(q)^m$.
2. ρ keeps alive at least $pn/2$ variables.

Using Lemma 70 and Eq. (5.2.4), Item 1 holds with probability at least $1 - s \cdot 2^{-t} \ge 1 - 2^{-t/2} \ge 1 - \exp(-\Omega(pn))$. Item 2 holds with probability at least $1 - \exp(-\Omega(pn))$ by Chernoff's bound. Thus, by a simple union bound

$$\Pr[\mathcal{E}] \ge 1 - \exp(-\Omega(pn)).$$

If $\Pr[\mathcal{E}] \le 1 - \varepsilon/2$, then we are done as Eq. (5.2.4) implies $\varepsilon/2 \le \exp(-\Omega(pn))$. Going forward, we may assume that $\Pr[\mathcal{E}] > 1 - \varepsilon/2$. In such a case, we claim that there exists a fixed restriction ρ that satisfies $\mathcal{E}$, under which $F|_\rho$ solves the Parity Halving Problem on at least $1/2 + \varepsilon/2$ fraction of the even-parity inputs consistent with ρ. Assume by contradiction otherwise. Under our assumption:

- For restrictions satisfying $\mathcal{E}$, the success probability of F on the even-parity inputs consistent with ρ is at most $1/2 + \varepsilon/2$.
- For other restrictions, the success probability of F on the even-parity inputs consistent with ρ is at most 1.

The key idea is that sampling a p-random restriction, and then sampling an input with even parity consistent with this restriction (if such an input exists), gives the uniform distribution over even-parity inputs. Thus, under the above assumption, the probability that F solves the Parity Halving Problem on a uniform input with even parity, is at most

$$\Pr[\mathcal{E}] \cdot (1/2 + \varepsilon/2) + \Pr[\neg\mathcal{E}] \cdot 1 \;<\; 1 \cdot (1/2 + \varepsilon/2) + (\varepsilon/2) \cdot 1 \;=\; 1/2 + \varepsilon,$$

yielding a contradiction.

We get that there exists a restriction ρ keeping at least $pn/2$ of the variables alive, under which $F|_\rho \in \mathrm{DT}(2t) \circ \mathrm{DT}(q)^m$, such that the success probability of F on the even-parity inputs consistent with ρ is at least $1/2 + \varepsilon/2$.

In the next and final step, we will focus on a single leaf of the partial decision tree for $F|_\rho$. For each leaf λ consider the further restriction of $F|_\rho$ according to the path leading to λ. This yields a new function, denoted $F_{\rho,\lambda} \in \mathrm{DT}(q)^m$. That is, $F_{\rho,\lambda}$ is a tuple of m decision trees of depth q. Moreover, for each λ, the number of variables left alive in $F_{\rho,\lambda}$ is at least $pn/2 - 2t \geq pn/4$.

We claim that there must exists a λ such that $F_{\rho,\lambda}$ solves the Parity Halving Problem on even-parity inputs consistent with ρ and λ with probability at least $1/2 + \varepsilon/2$. This follows by an averaging argument similar to the one we performed above. Indeed, to uniformly sample an input with even-parity consistent with ρ, we can first uniformly sample a root-to-leaf path along the partial decision tree resulting in a leaf λ, and then uniformly sample an even-parity input consistent with (ρ, λ). Since we succeed with probability at least $1/2+\varepsilon/2$ on uniform inputs with even-parity to F_ρ, we also succeed with probability at least $1/2 + \varepsilon/2$ on the inputs to some $F_{\rho,\lambda}$.

We get that there exists a restriction defined by (ρ, λ), leaving at least $pn/4$ variables alive, under which each output bit of F can be computed as a depth-q decision tree, and therefore depends on at most 2^q input bits. Furthermore, $F_{\rho,\lambda}$ solves the Parity Halving Problem on even-parity inputs consistent with ρ and λ with probability at least $1/2 + \varepsilon/2$. Applying Theorem 66 we get that

$$\varepsilon/2 \leq \exp\left(-\Omega\left(\frac{(pn)^2}{m \cdot 2^{2q}}\right)\right),$$

and by Eq. (5.2.4), plugging $p = 1/(m^{1/q} \cdot O(\log s)^{d-1})$,

$$\varepsilon/2 \leq \exp\left(\frac{-n^2}{m \cdot 2^{2q} \cdot O(\log s)^{2(d-1)} \cdot m^{2/q}}\right). \tag{5.7}$$

Recall that we have not set q yet. To minimize $2^{2q} \cdot m^{2/q}$ we pick $q = \sqrt{\log m}$. This gives

$$\varepsilon \leq \exp\left(\frac{-n^2}{m \cdot O(\log s)^{2(d-1)} \cdot 2^{4\sqrt{\log m}}}\right),$$

which concludes the proof. □

5.3 Relaxed Parity Halving Problem

In this section we deal with the issue that the QNC^0 circuit for PHP (Problem 62) needs a cat state, but QNC^0 circuits cannot create a cat state.

In Section 5.3.1, we first prove that a QNC^0 circuit cannot create a cat state. But, as we show, QNC^0 circuits can construct what we call a "poor man's cat state." This is a state of the form $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$ for some uncontrolled $z \in \{0,1\}^n$ alongside classical "side information" about z that allows us to determine it. In Section 5.3.2 we show the poor man's cat state lets us solve a relaxed version of the Parity Halving Problem, which is nevertheless hard for AC^0 circuits.

The goal of this section is to establish Theorem 58, which will follow from Theorem 77 and Theorem 78.

5.3.1 A poor man's cat state

We start by proving our claim that a QNC^0 circuit cannot construct a cat state in constant depth.

Theorem 72 (Cat states cannot be created in QNC^0). *Let C be a depth-d QNC^0 circuit over the gates set of all 2-qubit gates that maps $|0^{n+m}\rangle$ to $|😺_n\rangle \otimes |0^m\rangle$. Then $d \geq (\log n)/2$.*

Proof. Since C is a depth-d QNC^0 circuit over the gate set of 2-qubit gates, each input and output bit has a light cone of size 2^d. Similar to proposition 65, consider the intesection graph of the first n output bits, which hold the cat state. In this graph, the n output bits are the vertices, and two output bits are adjacent if their light cones contain a common input. Since the maximum number of input bits in the light cone of an output bit is 2^d, and each input bit can have at most 2^d output bits in its light cone, the maximum degree of an output bit in this intersection graph is 2^{2d}. Assume toward a contradiction that $d < (\log n)/2$. Then the maximum degree of the graph is less then n, and there must exist two disconnected vertices in the graph.

Let two such output qubits of $|😺_n\rangle$ be called y_i and y_j. These qubits depend on disjoint sets of input bits. Now we focus on the output of the the circuit C on these two qubits. Since they are part of the cat state, on measuring these in the computational basis, we will see either 00 or 11, with equal probability. Since the gates that are not in the light cones of these two qubits do not affect this, let us delete all these gates. Since the light cones were disjoint, we are now left with a circuit composed of two disjoint parts, one acting on a set of qubits that contains y_i and another acting on a set of qubits that contains y_j. Consider the cut between these two sets of qubits. Observe that the initial state, $|0^{n+m}\rangle$, is separable across this cut, but the output state is correlated across this cut although we have not performed any gates that cross the cut. This is impossible, and hence $d \geq (\log n)/2$. □

Now although QNC^0 circuits cannot create the cat state, we show that QNC^0 circuits are able to construct states of the form $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$, where z is some string in $\{0,1\}^n$ and $\bar{z}$ the complement of z. Note that this state is exactly the cat state when $z = 0^n$ or $z = 1^n$. The circuits that create this state also output an auxiliary classical string d such that z can be determined from d, up to the symmetry between z and $\bar{z}$. There is actually a family of QNC^0 circuits which construct these states, which we now describe.

Theorem 73 (Poor man's cat state construction). *For any connected graph $G = (V,E)$ with maximum degree Δ, there is a depth $\Delta + 2$ QNC^0 circuit which outputs a $|V|$ qubit state $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$, along with a bit string $d \in \{0,1\}^E$. Indexing the bits of z by vertices of V and the bits of d by edges of E, z and d satisfy the property that*

$$z_u + z_v \equiv \sum_{e \in P(u,v)} d_e \pmod 2 \tag{5.8}$$

for any two vertices $u, v \in V$, and any path $P(u,v)$ from u to v. Note that this condition also implies that the sum of d_e along any cycle in the graph is 0 (mod 2).

Proof. We first describe the QNC^0 circuit. Begin with $|V| + |E|$ qubits in the state $|0\rangle$, and identify each of the qubits with either an edge or a vertex of the graph. Apply the Hadamard transform for each vertex qubit. Now the state is $|+\rangle^{|V|} \otimes |0\rangle^{|E|}$. Then, for every edge $e = (u, v)$ in the graph, XOR the qubits indexed by u and v onto the edge qubit indexed by e (i.e., let the edge qubit store the parity of the two vertex qubits). Explicitly, this can be done by implementing CNOT gates from qubits u, v onto qubit e. (As discussed below, this can be done in $\Delta + 1$ parallel local steps.) Finally, measure all edge qubits in the standard basis.

To complete this proof we need to establish two claims: First, that the circuit leaves the unmeasured vertex qubits in the state $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$, while the measured edge qubits give the classical bitstring d. Second, that the circuit can be implemented in depth $\Delta + 2$.

We begin with the first claim. Imagine that we first only measure the $n - 1$ edges of some spanning tree T. Before measurement, the vertex qubits were in a uniform superposition over all possible 2^n states. Each measurement on an edge qubit had two equally probable outcomes, and observing the result of this measurement reduced the number of states in the superposition by half. More precisely, measuring the qubit for edge $e = (u, v)$ yields a bit $d_e \in \{0, 1\}$, which gives a linear equation on the state: $z_u \oplus z_v = d_e$. Thus, after all edges in the spanning tree are measured, the vertex qubits must be left in some two state superposition. Furthermore, after the spanning tree is measured any two vertex qubits u and v must differ by the parity of the observed measurements on edge qubits along the path from u to v. This shows the vertex qubits must be in the state $\frac{1}{\sqrt{2}}(|z\rangle + |\bar{z}\rangle)$, with the measurements on the edge qubits so far consistent with the requirements of Theorem 73. Now for any edge $e = (v, w)$ not in the spanning tree, the XOR measurements on the associated edge qubit is fixed to be equal to the XOR of edge qubit measurements along the path in T from v to w. This shows this measurement must also be consistent with the requirements of Theorem 73 and cannot affect the state of the vertex qubits. The establishes the first claim.

The second claim is more straightforward. The first layer of our QNC^0 circuit consists of Hadamard gates applied to all vertex qubits. It remains to show that we can implement all the desired CNOT gates in depth $\Delta + 1$. To show this we introduce a new graph G' with $|V| + |E|$ vertices, that is obtained from G by replacing each edge $e = (a, b)$ in E with a vertex v_e connected to its two end-points, a and b. Note the edges of G' are in one to one correspondence with the CNOT gates we want to implement in our circuit. By our assumptions on G, G' has degree at most Δ, and so Vizing's theorem tells us the edges of G' can be colored using at most $\Delta + 1$ colors. Since the edges in each color class are non-overlapping we can apply all the CNOT gates in one color class simultaneously, and thus apply all the CNOT gates in depth $\Delta + 1$. □

We will apply Theorem 73 when G is a spanning tree of a 2D grid, with diameter $2\sqrt{n}$ as depicted in Figure 5-2, which also describes the associated QNC^0 circuit.

This QNC^0 circuit has the nice feature that it is spatially local,[6] while any bit of z is specified by relatively few, $O(\sqrt{n})$, bits of d. This graph has constant degree $\Delta = 3$.

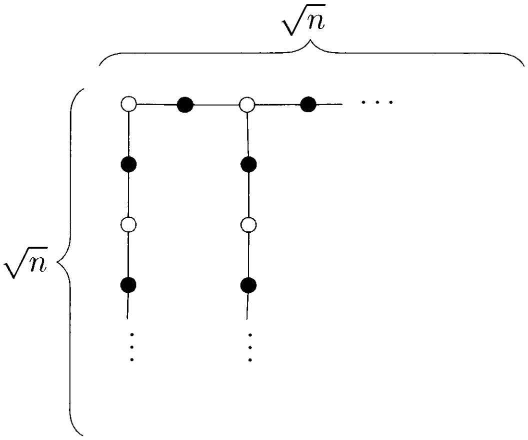


Figure 5-2: Grid Implementation of a Poor Man's Cat State. Black vertices are "edge" qubits, and are used to measure the parity of their neighbors. White vertices are "vertex" qubits. They are initialized in the $|+\rangle$ state, and make up the poor man's cat state after the edge qubits are measured.

It is worth mentioning that if we relax the requirement that our implementation be spatially local, we can improve on the number of bits of d required to specify any bit of z. In particular, applying Theorem 73 to a balanced binary tree gives an output string d with at most $\log n$ bits of d required to specify any bit of z. This version of the problem would lead to slightly better parameters in Theorem 58, but then our final problem would not longer be solved by a 2D quantum circuit and would no longer reduce to the 2D HLF problem. Hence the construction illustrated in Figure 5-2 will be sufficient for our purposes.

5.3.2 The Relaxed Parity Halving Problem

Having constructed a poor man's cat state, a natural idea would be to try and use this state instead of the cat state to solve the Parity Halving Problem. For example, we can feed the poor man's cat state into the quantum circuit (instead of a true cat state) and hope for the best. Unsurprisingly, this does not solve the Parity Halving Problem.

However, we can define a new problem from this failed attempt, which has QNC^0 circuits by construction. We call this problem the *Relaxed Parity Halving Problem,* although we will see that the precise definition of the problem depends on how the poor man's cat state is constructed.

Theorem 74 (PHP circuit applied to a poor man's cat state)**.** *The quantum circuit for* PHP_n *applied to the state* $\frac{|z\rangle+|\bar{z}\rangle}{\sqrt{2}}$, *where* $z \in \{0,1\}^n$ *(instead of the cat state), and*

[6] Here spatially local means here that circuit may be implemented in hardware with the qubits placed on a 2D grid and CNOT gates allowed only between neighbouring qubits.

an input $x \in \{0,1\}^n$ of even parity, yields an output string $y \in \{0,1\}^n$ such that

$$|y| \equiv \frac{1}{2}|x| + \langle z, x\rangle \pmod 2,$$

where $\langle z, x\rangle := \sum_{i\in[n]} z_i \cdot x_i$. Note that this is the same condition as for PHP_n *(Problem 62) except for the addition of the $\langle z, x\rangle$ term.*

Proof. Let us apply the quantum circuit solving PHP (as depicted in Figure 5-1) to the poor man's cat state. We first apply a phase gate (S gate) to qubit i of the poor man's cat state if $x_i = 1$. This yields the state

$$\frac{i^{\langle z,x\rangle}|z\rangle + i^{\langle \bar{z},x\rangle}|\bar{z}\rangle}{\sqrt{2}} = i^{\langle z,x\rangle} \cdot \frac{|z\rangle + i^{|x|-2\langle z,x\rangle}|\bar{z}\rangle}{\sqrt{2}} = i^{\langle z,x\rangle} \cdot \frac{|z\rangle + (-1)^{|x|/2-\langle z,x\rangle}|\bar{z}\rangle}{\sqrt{2}}.$$

Up to a global phase, which can be ignored, the state is $\frac{1}{\sqrt{2}}\big(|z\rangle + (-1)^{|x|/2-\langle z,x\rangle}|\bar{z}\rangle\big)$. The next stage of the algorithm applies Hadamard gates to all input qubits. Thus we have

$$H^{\otimes n}\left(\frac{|z\rangle + (-1)^{|x|/2-\langle z,x\rangle}|\bar{z}\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{y\in\{0,1\}^n} \left((-1)^{\langle y,z\rangle} + (-1)^{\langle y,\bar{z}\rangle}(-1)^{|x|/2-\langle z,x\rangle}\right)|y\rangle. \tag{5.9}$$

On measuring this state in the computational basis, we get only those y whose coefficient is nonzero. Hence we get a uniform distribution over all strings y satisfying $\langle y, z\rangle \equiv \langle y, \bar{z}\rangle + |x|/2 - \langle z, x\rangle \pmod 2$ or equivalently,

$$|y| \equiv \frac{1}{2}|x| + \langle z, x\rangle \pmod 2. \qquad \square$$

We now define a new problem based on this observation.

Problem 75 (Relaxed Parity Halving Problem for graph G). *Fix a connected graph $G = (V, E)$. Given an input $x \in \{0,1\}^V$ promised to have even parity, the* Relaxed Parity Halving Problem *or RPHP outputs $y \in \{0,1\}^V$ and $d \in \{0,1\}^E$, such that there exists a $z \in \{0,1\}^V$ with the property*

$$\forall (u,v) \in E,\ z_u \oplus z_v = d_{(u,v)}, \quad \text{and} \tag{5.10}$$

$$|y| \equiv \frac{1}{2}|x| + \langle z, x\rangle \pmod 2. \tag{5.11}$$

Note that in the definition above, a string z satisfying the first constraint exists if and only if the parity of d along every cycle is 0. If there are no cycles, then there always exists a z. When z does exist, it is unique up to complement, which does not change the second condition in the problem statement because $\langle z, x\rangle \equiv \langle \bar{z}, x\rangle \pmod 2$. To see this, recall that x has even parity, and hence $0 \equiv \langle 1, x\rangle \equiv \langle z, x\rangle + \langle \bar{z}, x\rangle \pmod 2$.

To fully specify the problem, we need a family of graphs with $|V| = n$ for infinitely many n. We are primarily interested in the 2D grid (so that the quantum circuit is specially local) and some reasonable (say, diameter $O(\sqrt{n})$) spanning tree of this graph. This gives us the Grid Relaxed Parity Halving Problem below. We use a spanning tree rather than the grid graph itself because deleting edges from G only makes the problem easier (since we can drop the corresponding bits of the output string d), which makes our lower bounds stronger. Additionally, choosing a spanning tree ensures that a string z satisfying the constraints of the problem always exists. It turns out the upper bounds (i.e., QNC^0 circuit we construct) can be easily modified to solve the problem on the entire 2D grid without deleting edges.

Problem 76 (Grid Relaxed Parity Halving Problem). *Consider the 2D grid of size $\sqrt{n} \times \sqrt{n}$ and fix an n vertex spanning tree $G = (V, E)$ of low diameter. For concreteness, fix the spanning tree which takes the first row of edges and all columns (as depicted in Figure 5-2). Then the* Grid Relaxed Parity Halving Problem *is the* RPHP *associated with G. That is, given $x \in \{0,1\}^V$, output $y \in \{0,1\}^V$ and $d \in \{0,1\}^E$ such that*

$$\forall (u,v) \in E,\ \ z_u \oplus z_v = d_{(u,v)}, \quad \text{and} \tag{5.12}$$

$$|y| \equiv \frac{1}{2}|x| + \langle z, x\rangle \pmod 2. \tag{5.13}$$

As discussed, we can use other graphs instead of the grid to define different variants of this problem. For instance, a balanced binary tree has lower diameter, which leads to slightly better parameters, but we use the grid to achieve a spatially local quantum circuit.

A path graph (sometimes called the line graph) is even simpler than the grid or tree. Unfortunately, the Relaxed Parity Halving Problem corresponding to the path graph can be solved by an NC^0 circuit (see section 5.1.1), which makes it unsuitable for proving a separation against RelNC^0.

5.3.3 Quantum circuit and AC^0 lower bound

In this section we establish Theorem 58, which states that Grid-RPHP can be solved in RelQNC^0, but it is average-case hard for AC^0 circuits. We start by establishing the quantum upper bound.

Theorem 77 (Grid-RPHP is in RelQNC^0). *There exists a depth-5 spatially local QNC^0 circuit that exactly solves* Grid-RPHP.

Proof. Let $G = (V, E)$ be a $O(\sqrt{n})$-diameter spanning tree of the $\sqrt{n} \times \sqrt{n}$ grid graph with $|V| = n$ and $|E| = n - 1$. This graph has degree $\Delta = 3$. As shown in Theorem 73, there is a spatially local QNC^0 circuit of depth $\Delta + 2$ to construct a random poor man's cat state $\frac{|z\rangle + |\bar{z}\rangle}{\sqrt{2}}$ (for some $z \in \{0,1\}^V$) and the associated string $d \in \{0,1\}^E$ for graph G such that

$$d_{(u,v)} = z_u \oplus z_v$$

for all $(u,v) \in E$. We run the QNC^0 circuit for the Parity Halving Problem as per

Theorem 74, and get an output $y \in \{0,1\}^V$ such that

$$|y| \equiv \frac{1}{2}|x| + \langle z, x\rangle \pmod 2.$$

We have defined Grid-RPHP so that it is not necessary to compute z, just a vector d consistent z, which we have from the construction of the poor man's cat state. Hence, we return y and d satisfying the condition, and we are done. □

This result is not surprising, since the relaxed parity halving problem is a relaxation of the parity halving problem explicitly constructed with the goal of having a QNC^0 circuit.

The nontrivial direction of the argument is to show that AC^0 cannot solve the Grid-RPHP problem. We accomplish this by exhibiting an NC^0 reduction to the Grid-RPHP problem from an instance of $\mathrm{PHP}_{n,m}$ with $m = \Theta(n^{3/2})$, which is still hard for RelAC^0. Note that although our reduction is an NC^0 reduction, it cannot be carried out with a QNC^0 circuit, so we are not showing that $\mathrm{PHP}_{n,m}$ is in RelQNC^0. While this might seem mysterious, the reason is that NC^0 has one ability that we have not given QNC^0: unbounded fan-out. Our reduction uses the fact that NC^0 can make unlimited copies of the output of a gate, whereas QNC^0 cannot do so.

Theorem 78 (Grid-RPHP is not in RelAC^0). *There is an NC^0 reduction from the problem $\mathrm{PHP}_{n,O(n^{3/2})}$ to* Grid-RPHP$_n$. *In particular, an AC^0 circuit of size $s \le \exp(n^{1/2(d+1)})$ and depth d cannot solve* Grid-RPHP$_n$ *with probability better than*

$$\frac{1}{2} + \exp\Big(-n^{1/2-o(1)} \Big/ O(\log s)^{2d}\Big)$$

on a uniformly random input with even parity.

Proof. Suppose we want to solve an instance of $\mathrm{PHP}_{n,O(n^{3/2})}$, and we can solve Grid-RPHP$_n$. Let $T = (V,E)$ be a $O(\sqrt{n})$-diameter spanning tree of an n vertex grid graph.

Take the input $x \in \{0,1\}^n$ from the PHP instance as input for a Grid-RPHP$_n$ instance, mapping the n bits arbitrarily to vertices of the grid. Solving this Grid-RPHP instance gives us two outputs $y \in \{0,1\}^V$ and $d \in \{0,1\}^E$ such that

$$|y| \equiv |x|/2 + \langle z, x\rangle \pmod 2,$$

where $z \in \{0,1\}^V$ satisfies the parity constraints in d. In particular, if we fix $z_1 = 0$ then each z_i is the parity of all d_j along a path from z_1 to z_i in the graph. Let $D_i \subseteq |E|$ denote the edges in the path from z_1 to z_i. Then we can write

$$\langle z, x\rangle = \sum_i z_i x_i \tag{5.14}$$

$$= \sum_i \sum_{j \in D_i} d_j x_i. \tag{5.15}$$

Since the diameter of the grid graph is $O(\sqrt{n})$, we may assume each D_i has size at most $O(\sqrt{n})$. Thus, we have expressed $\langle z, x\rangle$ as a sum of $O(n^{3/2})$ terms of the form $d_j x_i$. Note that any such term $d_j x_i$ is easy to compute with a single AND gate, since we have the string x (our input) and string d (the output of the Grid-RPHP circuit) available.

We now create $O(n^{3/2})$ new output bits, one for each term $d_j x_i$ that appears in the sum above. If we call this string of length $O(n^{3/2})$ y', then our final output for $\mathrm{PHP}_{n,O(n^{3/2})}$ is the string y (the output of the Grid-RPHP circuit) concatenated with the string y'. We claim this is a correct solution to our $\mathrm{PHP}_{n,O(n^{3/2})}$ instance. This is because $|y'| = \langle z, x\rangle$ by construction, and hence the output to $\mathrm{PHP}_{n,O(n^{3/2})}$, which is the concatenated string (y, y'), has parity

$$|y| + |y'| \equiv |x|/2 + \langle z, x\rangle + |y'| \equiv |x|/2 \pmod 2,$$

which satisfies the output condition of $\mathrm{PHP}_{n,O(n^{3/2})}$.

Note that using this reduction, if Grid-RPHP is solved by an AC^0 circuit of size s and depth d, then we get an AC^0 circuit for $\mathrm{PHP}_{n,O(n^{3/2})}$ of size $s + O(n^{3/2})$ and depth $d+1$. Applying Theorem 71 gives the required bound assuming $s \le \exp(n^{1/2(d+1)})$. □

5.4 Parallel Grid-RPHP

A common way to decrease the success probability of a problem is to repeat it in parallel and require success on every instance. We define the Parallel version of Grid-RPHP to simply be some number of copies of Grid-RPHP where the correctness condition is that all outputs must be correct. Obviously if there is a QNC^0 circuit for Grid-RPHP then there is one for Parallel Grid-RPHP. So the RelQNC^0 upper bound in Theorem 59 is clearly true. The remainder of this section is devoted to proving the lower bound in Theorem 59.

We need to show that Parallel Grid-RPHP becomes harder for AC^0 circuits, but let us start with showing that a parallel version of the Parity Halving Problem gets harder with more copies, and then we will reduce Parallel Grid-RPHP to this. Note that although the copies of the game are played in parallel, this does not represent a so-called "parallel repetition result" for Grid-RPHP because different copies of the game are played by *different* players.

5.4.1 Parallel Parity Halving Problem

We now define the parallel version of PHP, with k copies of the problem.

Problem 79 (Parallel Parity Halving Problem, $\mathrm{PHP}_{n,m}^{\otimes k}$). *Given k strings $x_1, \dots, x_k \in \{0,1\}^n$ of length n as input, promised that each x_i has even parity, output k strings $y_1, \dots, y_k \in \{0,1\}^m$ of length m such that*

$$|y_i| \equiv \frac{1}{2}|x_i| \pmod 2$$

for all $1 \le i \le k$.

In other words, $\mathrm{PHP}_{n,m}^{\otimes k}$ is simply k independent copies of the Parity Halving Problem, and the players win if they solve all of the subgames simultaneously. Clearly a QNC^0 circuit will have no problem solving this, given k cat states.

Proposition 80. *There is a depth-2, linear-size* $\mathsf{QNC}^0/\mathsf{qpoly}$ *circuit which solves* $\mathrm{PHP}_{n,n}^{\otimes k}$ *with certainty. More specifically, the quantum advice is* k *cat states of size* n, $|🐱_n\rangle^{\otimes k}$.

The classical lower bound, however, will require some new ideas. It is not always easy to show that solving many independent instances of a problem is as hard as solving all of them independently, because of the possibility of correlating success in one instance with success in another. We use Vazirani's XOR Lemma [116] to attack this problem indirectly.

Lemma 81 (Vazirani's XOR Lemma). *Let* D *be a distribution on* $\mathbb{F}_2^m$ *and* p_S *denote the parity function on the set* $S \subseteq [m]$*, defined as* $p_S(x) = \oplus_{i \in S} x_i$. *If* $|\mathbb{E}_{x \in D}[(-1)^{p_S(x)}]| \le \varepsilon$ *for every non-empty subset* $S \subseteq [m]$*, then* D *is* $\varepsilon \cdot 2^{m/2}$ *close (in statistical distance) to the uniform distribution over* $\mathbb{F}_2^m$.

Note that this bound is very intuitive when $\varepsilon = 0$. It says that if a distribution has the property that on every subset of bits, if the induced distribution places equal mass on even and odd parity strings, then this distribution must be the uniform distribution.

To get an effective bound in Lemma 81 we need to guarantee that $\varepsilon < 2^{-m/2}$. The following simple lemma handles bigger ε effectively, but only guarantees that the probability to sample the all zeros input is small, as opposed to guaranteeing that the distribution is close to uniform.

Lemma 82 (Special case of the XOR Lemma). *Let* D *be a distribution on* $\mathbb{F}_2^m$. *If* $|\mathbb{E}_{x \in D}[(-1)^{p_S(x)}]| \le \varepsilon$ *for every non-empty subset* $S \subseteq [m]$*, then,* $\Pr_{x \sim D}[x = 0^m] \le 2^{-m} + \varepsilon$.

Proof. Fix $y \in \mathbb{F}_2^m$. Let $f : \{0,1\}^m \to \{0,1\}$ be the indicator function that checks whether a given input is equal to 0^m.

$$f(x) = \prod_{i=1}^{m} \frac{(-1)^{x_i} + 1}{2} = 2^{-m} \sum_{S \subseteq [m]} (-1)^{p_S(x)} = 2^{-m} + 2^{-m} \sum_{\varnothing \neq S \subseteq [m]} (-1)^{p_S(x)}$$

Thus,

$$\Pr_{x \sim D}[x = 0^m] = \mathbb{E}_{x \sim D}[f(x)] \le 2^{-m} + 2^{-m} \cdot \sum_{\varnothing \neq S \subseteq [m]} \left|\mathbb{E}_{x \sim D}[(-1)^{p_S(x)}]\right| \le 2^{-m} + \varepsilon. \qquad \square$$

The relevance of the XOR lemma is the following: Consider the task of solving k instances of some problem with some class of circuits. Say we know that solving 1 instance of the problem is hard, in the sense that no circuit from our class solves the problem with probability significantly greater than half on some hard distribution over inputs. For our task with k instances we will choose the input distribution to be this hard distribution on all instances independently. Now define a single bit random variable for each instance that indicates whether a given circuit correctly solved that instance on our chosen distribution. We know that for 1 instance this bit, the random variable we defined, is essentially a coin flip. If we can prove that each bit is essentially a coin flip, and furthermore that the XOR of any subset of bits is essentially a coin flip, then we will get that the distribution is essentially uniform. Which means the probability of getting the all zeros output, which corresponds to the circuit correctly solving all instances, is exponentially small.

Consider an instance of $\mathrm{PHP}_{n,m}^{\otimes k}$. If we solve all the instances correctly, then the parity of the entire output of length km (all instances included) is the same as half the entire input's Hamming weight mod 2. This just follows from the definition of PHP. If we solve all but one instance correctly, then this condition will not hold. In general, we fail on an even number of subgames if the parity of the entire output is the same as half the entire input Hamming weight mod 2. But that is just the usual condition for the Parity Halving Problem on an input of size kn and output of size km. The only difference is that each instance additionally has an even-parity input. Thus we need a stronger version of Theorem 64 which allows for parity constraints on the input. As in Theorem 64, we prove this theorem in the language of non-local games.

Theorem 83. *Consider the* constrained *Parity Halving Game on n players, in which the inputs of d_1 players are fixed, and the remaining $n-d_1$ players are partitioned into d_2 parts (of size ≥ 2) with each part constrained to some fixed parity. The probability of winning this version of the problem is*

$$\Pr[\mathrm{Win}] \leq \frac{1}{2} + 2^{-(n-d_1)/2+d_2}.$$

Proof. The proof builds on Theorem 64. The main change is that we need a different function f, to capture the different promise. Suppose for now that $d_1 = 0$. Say the input bits are divided into sets $S_1, \ldots, S_d$ and constrained to have parity $p_1, \ldots, p_d \in \{0,1\}$. We define f such that

$$f(x) := \frac{1}{2^d} \prod_{k=1}^{d} \left(i^{\sum S_k} + (-1)^{p_d} (-i)^{\sum S_k} \right)$$

The idea is that $\frac{1}{2}\left(i^{\sum S_k} + (-i)^{\sum S_k}\right)$ is exactly $\mathrm{Re}(i^{\sum S_k})$, so it is 0 if the parity on S_k is odd and $i^{\sum S_k}$ otherwise. Similarly, $\frac{1}{2}\left(i^{\sum S_k} - (-i)^{\sum S_k}\right)$ is 0 if the parity on S_k is even and $i^{\sum S_k}$ otherwise. Altogether, this means that $f(x)$ is 0 if the promise is violated and $i^{|x|}$ otherwise.

On the other hand, if we expand $f(x)$, we see that it is a convex combination of terms of the form $\pm(\pm i)^{x_1}(\pm i)^{x_2}\cdots(\pm i)^{x_n}$, and we have essentially already argued that

$$\left|\sum_x (-1)^{a+b\cdot x}(\pm i)^{x_1}\cdots(\pm i)^{x_n}\right| \le 2^{n/2}.$$

It follows that the correlation of $(-1)^{a+b\cdot x}$ and $f(x)$, denoted by χ, is at most $\frac{2^{n/2}}{2^{n-d}} = 2^{-n/2+d}$.

The probability of winning the game on a random input satisfying the promise is $\frac{1+\chi}{2}$, or

$$\Pr[\text{Win}] \le \frac{1}{2} + 2^{-n/2+d-1}.$$

Plugging d_1 and d_2 back in, we have

$$\Pr[\text{Win}] \le \frac{1}{2} + 2^{-(n-d_1)/2+d_2-1},$$

as desired. □

With this result we can now show that the Parallel Parity Halving Problem is hard for AC^0 circuits.

Theorem 84 (Parallel PHP is not in RelAC^0). *Let $k = n$ and $m \in [n, n^2]$. Any AC^0 circuit $F : \{0,1\}^{nk} \to \{0,1\}^{mk}$ with depth d and size $s \le \exp((kn)^{1/2d})$ solves $\mathrm{PHP}_{n,m}^{\otimes k}$ with probability at most $\exp(n^2/(m^{1+o(1)} \cdot O(\log s)^{2(d-1)}))$.*

Proof. Much like Theorem 71, we assume F solves the problem and randomly restrict it. We pick parameters $q = \sqrt{\log(mk)}$, $p = 1/(O(\log s)^{d-1} \cdot (mk)^{1/q})$ that are similar to the ones picked in Theorem 71, but adjusted to the input and output lengths. We apply p-random restrictions. Most of the time this will simplify F to the point that $F|_\rho \in \mathrm{DT}(pnk/4) \circ \mathrm{DT}(q-1)^{mk}$ and ρ keeps at least $pnk/2$ variables alive. However, with probability $\exp(-\Omega(pnk))$ the restriction will fail, and we are forced to assume (pessimistically) that $F|_\rho$ solves the problem perfectly in these cases. This probability of failure is negligible compared to $\exp(-n^2/(m^{1+o(1)} \cdot O(\log s)^{d-1}))$.

Let us assume the random restriction did its job, and now $F|_\rho$ is computed by common partial decision tree followed by a forest of depth-q decision trees. By averaging argument, it suffices to show that for each leaf λ in the partial decision tree, $F_{\rho,\lambda}$ solves the Parallel Parity Halving Problem on the legal inputs consistent with (ρ, λ) with exponentially small probability.

For each leaf in the partial decision tree, λ, the circuit $F_{\rho,\lambda}$ has $pnk/4$ inputs, mk outputs, and locality 2^q. By Proposition 65, we may further restrict down to a subset of $\Omega((pnk)^2/(mk2^{2q}))$ inputs so that each output bit depends on at most one input bit. Finally, we restrict one more time to eliminate subgames where fewer than say, half the average number of inputs (which is $\Omega((pn)^2/(m2^{2q}))$ are unrestricted. Subgames with too few inputs may be too easy to win and prevent us from using the XOR Lemma. This last step kills at most half of the input bits that were alive before this step.

The remaining circuit, which we will call C, satisfies the following:

- C has $\Omega((pn)^2k/(m2^{2q}))$ unrestricted inputs.
- Each output bit depends on at most one input bit.
- Each subgame is either fixed or has at least $n' = \Omega((pn)^2/(m2^{2q}))$ unrestricted bits.

Note that there are at least $\ell = \Omega(p^2nk/(m2^{2q})) = \Omega(n')$ remaining subgames. We shall show that the probability to win the remaining subgames is at most $2^{-\Omega(n')}$.

Define a distribution D on ℓ bit strings which runs the circuit C on a random input satisfying the promise, and outputs a string of bits, $w \in \{0,1\}^\ell$, one bit for each of the remaining ℓ subgames, which is 0 if the circuit wins the subgame and 1 if the circuit loses the subgame. We will argue that

$$\left|\mathbb{E}_{w\in D}[(-1)^{ps(w)}]\right| \leq 2^{-\Omega(n')}$$

for each non-empty $S \subseteq [\ell]$. That is, the parity of any non-empty subset of the games is exponentially close to a coin flip. Once we show this, Lemma 82 applies and says that the probability to sample 0^ℓ from D is at most $2^{-\ell} + 2^{-\Omega(n')} \leq 2^{-\Omega(n')}$. Thus, we win with probability $2^{-\Omega(n')}$, which dominates the $\exp(-\Omega(pnk))$ probability that the restriction fails.

All that remains to show is that for any non-empty subset of subgames, the circuit loses an even number of subgames with probability exponentially close to $\frac{1}{2}$. Notice that losing an even number of subgames is equivalent to winning a constrained Parity Halving game on the combined inputs and outputs of the subgames, where the constraints ensure each subgame has even parity input. For a subset $S \subseteq [\ell]$ of the subgames, Theorem 83 says that the probability of winning is at most $\frac{1}{2} + 2^{-\Omega(|S|n')+|S|}$, since there are at least $|S|n'$ inputs and at most $|S|$ relevant constraints. This is maximized when $|S| = 1$, where we get that the probability of winning is at most $1/2 + 2^{-\Omega(n')}$.

To finish we note that $n' = \Omega((pn)^2/(m2^{2q}) \geq n^2/(m^{1+o(1)} \cdot O(\log s)^{2(d-1)})$. □

5.4.2 Parallel Grid-RPHP

We are now ready to prove the lower bound for Parallel Grid-RPHP by reduction to Theorem 84. As before, we define Grid-RPHP$_n^{\otimes k}$ to be the problem where we are given k copies of Grid-RPHP$_n$, and the correctness condition is that all copies are correct. We are now ready to show the lower bound.

Theorem 85 (Parallel Grid-RPHP is not in RelAC^0). *Choose $k = n$. Any AC^0 circuit of size s and depth d for Grid-RPHP$_n^{\otimes k}$ succeeds on a random input with probability at most $\exp(-n^{1/2-o(1)}/O(\log s)^{2d})$.*

Proof. We use the reduction from PHP$_{n,O(n^{3/2})}$ to Grid-RPHP$_n$ (Theorem 78) on each of the k instances. Note that the reduction does not change the inputs, and only manipulates the output bits. Assume C is a circuit of size s and depth d for Grid-RPHP$_n^{\otimes k}$ that succeeds with probability ε (over the uniform distribution over inputs that satisfy the promise). Then, there exists a circuit C' of size poly(s) and depth $d+1$ that succeeds with probability at least ε on the same input distribution. By Theorem 84, with $m = O(n^{3/2})$, we get that

$$\varepsilon \leq \exp\left(-n^{1/2-o(1)} \middle/ O(\log s)^{2d}\right),$$

which yields the bound stated in the theorem. □

This implies the second part of Theorem 59 and completes its proof.

5.5 Relation to Hidden Linear Function Problems

Finally, to establish our main result (Theorem 56), we have to show that our final problem, Parallel Grid-RPHP, reduces to the 2D HLF problem. Since we have already established the required hardness for Parallel Grid-RPHP in Theorem 59, we will then be done.

We start by recalling the general Hidden Linear Function problem (HLF) defined by Bravyi, Gosset, and König [26].

Problem 86 (Hidden Linear Function problem). *We are given as input a symmetric matrix $A \in \{0,1\}^{n\times n}$ and vector $b \in \{0,1,2,3\}^n$. From these, define a quadratic form $q\colon \mathbb{F}_2^n \to \mathbb{Z}_4$ as $q(u) := u^T A u + b^T u \pmod 4$. Define $\mathcal{L}_q$ as follows:*

$$\mathcal{L}_q := \{u \in \mathbb{F}_2^n : \forall v \in \mathbb{F}_2^n,\ q(u \oplus v) \equiv q(u) + q(v) \pmod 4\}.$$

Bravyi et al. [26] show that (i) $\mathcal{L}_q$ is a linear subspace of $\mathbb{F}_2^n$, (ii) for all $u \in \mathcal{L}_q$, $q(u) \in \{0,2\}$, and (iii) q is linear on $\mathcal{L}_q$. Since q is linear on $\mathcal{L}_q$, there exists a $p \in \mathbb{F}_2^n$ such that $q(u) \equiv 2p^T u \pmod 4$ for all $u \in \mathcal{L}_q$. The goal is to output any string p satisfying this condition.

Theorem 87. *There is an* NC^0 *reduction from* RPHP *on any graph G to the* HLF *problem.*

Proof. As discussed, the Relaxed Parity Halving Problem is well defined for any connected graph $G = (V, E)$ (Problem 75). Given an even-parity vector $x \in \{0,1\}^V$, the goal is to output $y \in \{0,1\}^V$ and $d \in \{0,1\}^E$, such that the parity of d on any cycle is 0, and the unique (up to complement) $z \in \{0,1\}^V$ satisfying the parity conditions implied by d also satisfies

$$|y| \equiv \frac{|x|}{2} + z^T x \pmod 2.$$

Note that if we take the complement of z instead, the condition does not change because $(z^T + \overline{z}^T)x \equiv 1^T x \equiv |x| \equiv 0 \pmod 2$.

We now describe our reduction from RPHP on $G = (V, E)$ to the HLF problem. In our reduction, $n = |V| + |E|$. We define the input $A \in \{0,1\}^{n\times n}$ to HLF from the graph G, and the input $b \in \{0,1,2,3\}^n$ to HLF from the input $x \in \{0,1\}^V$ to RPHP. Let $M \in \mathbb{F}_2^{|V|\times|E|}$ be the incidence matrix of the graph G. Then we define A and b as

$$A = \begin{pmatrix} 0 & M \\ M^T & 0 \end{pmatrix}, \text{ and } b = \begin{pmatrix} x \\ 0 \end{pmatrix},$$

where the 0 above refers to the all zeros matrix or vector as appropriate. The solution to this HLF problem is some vector p, which we claim solves the Relaxed Parity Halving Problem with the identification $p = \begin{pmatrix} y \\ d \end{pmatrix}$. Let us verify this claim.

With this choice of A and b, the quadratic form q becomes

$$q\begin{pmatrix} u_V \\ u_E \end{pmatrix} \equiv u_V^T M u_E + u_E^T M^T u_V + x^T u_V \pmod 4 \tag{5.16}$$

$$\equiv 2u_V^T M u_E + u_V^T x \pmod 4. \tag{5.17}$$

Now note that since G is connected, M has rank $|V|-1$. Specifically, the column span $\{Mw : w \in \mathbb{F}_2^{|E|}\} \subseteq \mathbb{F}_2^{|V|}$ is the set of all vectors of even parity. Since our input to RPHP, $x \in \mathbb{F}_2^{|V|}$, has even parity, there exists a $w \in \mathbb{F}_2^{|E|}$ such that $x = Mw$. Let us show that the vector u is in $\mathcal{L}_q$, where

$$u := \begin{pmatrix} 1 \\ w \end{pmatrix},$$

and 1 is the all ones vector of size $|V|$. This means we want that for all v, $q(u \oplus v) - q(u) - q(v) \equiv 0 \pmod 4$. Let us verify this for an arbitrary vector $v = \begin{pmatrix} v_V \\ v_E \end{pmatrix}$ using the following calculation (performed modulo 4):

$$q\left(\begin{pmatrix} 1 \\ w \end{pmatrix} \oplus \begin{pmatrix} v_V \\ v_E \end{pmatrix}\right) - q\begin{pmatrix} 1 \\ w \end{pmatrix} - q\begin{pmatrix} v_V \\ v_E \end{pmatrix} \tag{5.18}$$

$$\equiv 2(1 \oplus v_V)^T M(w \oplus v_E) + (1 \oplus v_V)^T x - 2(1^T M w) - 1^T x - 2v_V^T M v_E - v_V^T x \tag{5.19}$$

$$\equiv 2(1 + v_V)^T M(w + v_E) + (1 \oplus v_V)^T x - 2(1^T M w) - 1^T x - 2v_V^T M v_E - v_V^T x \tag{5.20}$$

$$\equiv 2(1^T M v_E) + 2v_V^T M w + (1 \oplus v_V)^T x - 1^T x - v_V^T x \tag{5.21}$$

$$\equiv 2v_V^T x + (1 \oplus v_V)^T x - 1^T x - v_V^T x \tag{5.22}$$

$$\equiv 2v_V^T x + (1 - v_V)^T x - 1^T x - v_V^T x \tag{5.23}$$

$$\equiv 0. \tag{5.24}$$

In eq. (5.20), we used that for $a, b \in \{0,1\}$, we have $2(a \oplus b) \equiv 2(a+b) \pmod 4$. In eq. (5.22) we used $Mw \equiv x \pmod 2$ and that $1^T M \equiv 0 \pmod 2$, since M is an incidence matrix and any column has exactly two ones. In eq. (5.23) we used that for any $a \in \{0,1\}$, $1 \oplus a = 1 - a$.

Since $u \in \mathcal{L}_q$, and q is linear on $\mathcal{L}_q$, we have

$$q(u) \equiv 2p^T u \equiv 2 \begin{pmatrix} y \\ d \end{pmatrix}^T \begin{pmatrix} 1 \\ w \end{pmatrix} \equiv 2(y^T 1 + d^T w) \pmod 4. \tag{5.25}$$

To prove that the output of HLF on our chosen inputs is a valid output for RPHP, we need to verify the two conditions in Problem 75. The first condition requires d to be related to some $z \in \{0,1\}^V$; the entries of d are differences (along edges) of two entries in z. If such a z existed, then we would have $d = M^T z$. As noted after Problem 75, for z to exist it suffices to show that the parity of d along any cycle is 0. In other words, we need to show that if $c \in \{0,1\}^E$ is the indicator vector of any cycle, then $d^T c \equiv 0 \pmod 2$.

To show this, note that for any cycle we have $Mc \equiv 0 \pmod 2$. Since $q(\binom{0}{c}) = 0$, we also have

$$\begin{aligned} q\left(\begin{pmatrix} u_V \\ u_E \end{pmatrix} \oplus \begin{pmatrix} 0 \\ c \end{pmatrix}\right) &\equiv 2u_V^T M(u_E \oplus c) + u_V^T x \pmod 4 && (5.26) \\ &\equiv 2u_V^T M u_E + 2u_V^T Mc + u_V^T x \pmod 4 && (5.27) \\ &\equiv q\begin{pmatrix} u_V \\ u_E \end{pmatrix} + q\begin{pmatrix} 0 \\ c \end{pmatrix} \pmod 4. && (5.28) \end{aligned}$$

In other words, $(\binom{0}{c})$ is in $\mathcal{L}_q$ for all cycles. Thus,

$$0 \equiv q\begin{pmatrix} 0 \\ c \end{pmatrix} \equiv 2\begin{pmatrix} y \\ d \end{pmatrix}^T \begin{pmatrix} 0 \\ c \end{pmatrix} \equiv 2d^T c \pmod 4,$$

which implies that $d^T c \equiv 0 \pmod 2$.

Now that we know $d = M^T z$ for some z, it follows that $d^T w = z^T M w = z^T x$, and therefore from eq. (5.25) we have $q(u) = 2(|y| + z^T x)$. On the other hand, we can also evaluate $q(u)$ from the quadratic form definition, eq. (5.17), which gives

$$\begin{aligned} q(u) &\equiv 2(1^T M w) + 1^T x \pmod 4 && (5.29) \\ &\equiv 3|x| \pmod 4 && (5.30) \\ &\equiv |x| \pmod 4, && (5.31) \end{aligned}$$

where the last equivalence follows from the fact that x has even parity.

Thus solving the HLF problem on our chosen inputs produces a solution such that $2|y| + 2z^T x \equiv |x| \pmod 4$, which satisfies the second condition of problem 75. Hence the outputs to HLF, y and d (and implicitly z), satisfy the conditions of the Relaxed Parity Halving Problem. □

The main problem studied in Bravyi, Gosset and König [26] is actually a version of HLF on an $N \times N$ grid called the 2D Hidden Linear Function problem (2D HLF). More specifically, in 2D HLF, the matrix A is supported only on the grid in the sense that $A_{ij} = 0$ if there is no edge from vertex i to vertex j on the 2D grid. The reduction in Theorem 87 roughly preserves the topology of the graph, so it is not difficult to show a reduction from Relaxed Parity Halving on the 2D grid to the 2D HLF.

Corollary 88. *There is an* NC^0 *reduction from* Grid-RPHP *to* 2D HLF.

Proof. The plan is use the same reduction to HLF as above, and observe that the reduction actually creates a 2D HLF instance. Obviously, RPHP on a grid starts from a grid graph. The reduction creates a matrix with $|V| + |E|$ rows and columns from this graph on $|V|$ vertices. We can think of the reduction as transforming the graph by creating a new vertex for each edge, then splitting each edge into two, both incident at the new vertex. The matrix A is then supported on this transformed graph. Fortunately, the transformation takes the $n \times n$ grid graph to a subgraph of the $(2n-1) \times (2n-1)$ grid graph. This means the HLF instance constructed when starting from a 2D graph is actually a 2D HLF instance, which completes the proof. □

Furthermore, we can solve multiple instances of Grid-RPHP by solving a single instance of 2D HLF.

Lemma 89. *Consider an instance of* HLF *with input* $A \in \{0,1\}^{n\times n}$ *and* $b \in \{0,1,2,3\}^n$ *such that* A *is block diagonal with blocks* $A_1, \dots, A_k$ *(all square matrices of various sizes), and the corresponding division of* b *is* $b_1, \dots, b_k$*. Then any solution* p *to the instance* (A, b) *is a direct sum of solutions* p_i *to instances* (A_i, b_i) *of* HLF.

Proof. It suffices to prove the result for $k = 2$ blocks, since we can break up k blocks into two blocks of size 1 and $k - 1$ and prove the claim by induction.

Now let $u = (u_1, u_2)$. Then the block structure of A gives $q(u) \equiv q_1(u_1) + q_2(u_2) \pmod 4$, where

$$q_1(u_1) := u_1^T A_1 u_1 + b_1^T u_1 \pmod 4, \quad \text{and} \quad q_2(u_2) := u_2^T A_2 u_2 + b_2^T u_2 \pmod 4. \tag{5.32}$$

Suppose we have $u = (u_1, u_2) \in \mathcal{L}_q$. By definition, for all $v = (v_1, v_2) \in \{0,1\}^n$ we have

$$q(u_1 \oplus v_1, u_2 \oplus v_2) \equiv q(u_1, u_2) + q(v_1, v_2) \pmod 4.$$

By rearranging the terms, we have

$$q_1(u_1 \oplus v_1) \equiv q_1(u_1) + q_1(v_1) + q_2(u_2) + q_2(v_2) - q_2(u_2 \oplus v_2) \pmod 4.$$

In particular, if $v_2 = 0$ then for all v_1 we have

$$q_1(u_1 \oplus v_1) \equiv q_1(u_1) + q_1(v_1) \pmod 4,$$

which proves $u_1 \in \mathcal{L}_{q_1}$. Similarly, $u_2 \in \mathcal{L}_{q_2}$, and so we have that $\mathcal{L}_q = \mathcal{L}_{q_1} \oplus \mathcal{L}_{q_2}$.

Finally, suppose $p = (p_1, p_2)$ is a valid solution to HLF on input (A, b). For all $u = (u_1, u_2) \in \mathcal{L}_q$ we have

$$q_1(u_1) + q_2(u_2) \equiv q(u) \equiv 2p^T u \equiv 2p_1^T u_1 + 2p_2^T u_2 \pmod 4.$$

In particular, for all $u_1 \in \mathcal{L}_{q_1}$ we have $(u_1, 0) \in \mathcal{L}_q$ and hence

$$q_1(u_1) \equiv 2p_1^T u_1 \pmod{4}$$

for all $u_1 \in \mathcal{L}_{q_1}$. It follows that p_1 is a solution to HLF on the instance (A_1, b_1), and (by symmetry) p_2 is a solution to HLF on the instance (A_2, b_2).

Conversely, if p_1 and p_2 are solutions to (A_1, b_1) and (A_2, b_2), then $p = (p_1, p_2)$ is a solution to (A, b) since

$$q(u) \equiv q_1(u_1) + q_2(u_2) \equiv 2p_1^T u_1 + 2p_2^T u_2 \equiv 2p^T u \pmod{4},$$

and $u \in \mathcal{L}_q$ implies $u_1 \in \mathcal{L}_{q_1}$ and $u_2 \in \mathcal{L}_{q_2}$. □

Corollary 90. *Parallel* Grid-RPHP *reduces to* 2D HLF.

Proof. The parallel version of Grid-RPHP is just many instances of RPHP which we are expected to solve simultaneously. If we reduce each instance to a 2D HLF problem, then combine the instances as in the lemma above, then solving the combined HLF instance gives solutions to all the individual HLF instances, and hence solutions for all the parallel Grid-RPHP instances. □

5.6 Parity Bending Problem

We now move on to the Parity Bending Problem discussed in the introduction.

Problem 91 (Parity Bending Problem, PBP$_n$). *Given an input $x \in \{0,1\}^n$, output a string $y \in \{0,1\}^n$ such that*

$$|y| \equiv 0 \pmod{2} \quad \textit{if} \quad |x| \equiv 0 \pmod{3} \ \textit{and} \tag{5.33}$$
$$|y| \equiv 1 \pmod{2} \quad \textit{otherwise.} \tag{5.34}$$

Our goal is to prove a quantum advantage for QNC^0/🐱 circuits. Theorem 60 states our main results about this problem.

Theorem 60. *The Parity Bending Problem* (PBP$_n$) *can be solved by a depth-2, linear-size quantum circuit starting with the* $|🐱_n\rangle$ *state with probability at least* $3/4$ *on any input. But there exists an input distribution on which any* $\mathsf{AC}^0[2]/\mathsf{rpoly}$ *circuit of depth d and size at most* $\exp\left(n^{\frac{1}{10d}}\right)$ *only solves the problem with probability* $\frac{1}{2} + \frac{1}{n^{\Omega(1)}}$.

Most of this section is devoted to a proof of Theorem 60. We begin by introducing the quantum circuit that solves this problem (Theorem 92) in Section 5.6.1, and then prove the classical lower bound (Theorem 96) in Section 5.6.2. Finally, in Section 5.6.3 we establish Theorem 61, which shows that a parallel version of the game can further separate the success probabilities of classical and quantum circuits.

5.6.1 Upper bounds

As in the previous section, the QNC^0 circuit solving the Parity Bending Problem can be thought of as an implementation of a quantum strategy for a cooperative non-local game. In the Parity Bending Problem, the players are each given one bit of an n-bit string, and want to give an output satisfying the same criterion as in Problem 91. It is known that a quantum strategy can win this game with probability larger than classically possible, although even quantum players cannot achieve probability 1 [14]. We now describe the quantum strategy.

Theorem 92 (Quantum circuit for PBP_n). *There is a quantum strategy for the Parity Bending Problem which wins with certainty on inputs with Hamming weight* $0 \pmod 3$ *and with probability* $3/4$ *on any other input. For an* n*-player game, this strategy only requires an* n*-qubit cat state,* $|😸_n\rangle$.

Proof. The quantum strategy for this game is similar to the one for Problem 62 described in Theorem 63 and Figure 5-1, except that the controlled-S gates, which add a phase of i if both qubits are 1, are replaced with controlled-$R_z(2\pi/3)$ gates, which add a phase of $e^{2\pi i/3}$ when both qubits are 1. We define $R_z(2\pi/3)$ to be the matrix $\left(\begin{smallmatrix}1 & 0\\ 0 & e^{2\pi i/3}\end{smallmatrix}\right)$.

We now describe the strategy in more detail. Each player starts with their input bit x_i and a single qubit of the $|😸_n\rangle$ state. If their input bit is $x_i = 1$, they apply a rotation $R_z(2\pi/3)$ to their qubit of the cat state, otherwise they do nothing. This is equivalent to applying a controlled-$R_z(2\pi/3)$ gate with x_i as the control qubit and their qubit of the cat state as the target. Then they apply a Hadamard gate to their qubit of the cat state and measure that qubit.

Given an input string x, after the controlled rotations and before the Hadamard gate, the cat state has been transformed into

$$\bigotimes_{j=0}^{n}\left(R_z(2\pi/3)^{x_j}\right)\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right) = \frac{1}{\sqrt{2}}\left(|0^n\rangle + \exp\left(\frac{2\pi i|x|}{3}\right)|1^n\rangle\right). \tag{5.35}$$

If $|x| \equiv 0 \pmod 3$ this state is just the cat state. As noted in Section 5.2

$$H^{\otimes n}\left(\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)\right) = |\Psi_{\text{even}}\rangle, \tag{5.36}$$

where $|\Psi_{\text{even}}\rangle$ is a uniform superposition over all even parity n-bit strings. So we see the players win the game with probability 1 on an input with Hamming weight 0 (mod 3).

If $|x| \equiv 1 \pmod 3$ or $|x| \equiv 2 \pmod 3$, the state after rotation and before the Hadamard gates is given by

$$\frac{1}{\sqrt{2}}\left(|0^n\rangle + \exp(\pm 2\pi i/3)\,|1^n\rangle\right).$$

Note that this state lives in the span of the states $\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$ and $\frac{1}{\sqrt{2}}\left(|0^n\rangle - |1^n\rangle\right)$, and that

$$H^{\otimes k}\left(\frac{1}{\sqrt{2}}\big(|0^n\rangle - |1^n\rangle\big)\right) = |\Psi_{\text{odd}}\rangle\,, \tag{5.37}$$

with $|\Psi_{\text{odd}}\rangle$ being the uniform superposition over odd parity n-bit strings. Then the players win the game given input with Hamming weight 1 or 2 (mod 3) with probability exactly

$$\left|\frac{1}{2}\big(\langle 0^n| - \langle 1^n|\big)\big(|0^n\rangle + \exp(\pm 2\pi i/3)\,|1^n\rangle\big)\right|^2 = \frac{1}{4}(2 + 2\cos(\pm 2\pi/3)) = \frac{3}{4}. \qquad \square$$

It is clear that the strategy described in Theorem 92 can be implemented by a QNC^0/🐱 circuit. By the analysis given above it is also clear that this circuit succeeds at Parity Bending with worst case probability 3/4, and probability 5/6 against inputs drawn from a uniform distribution.

5.6.2 Lower bounds

The main tool used in this section is a reduction from the Parity Bending Problem to the Mod 3 problem. We begin with a formal definition of the Mod 3 problem.

Problem 93 (Mod 3). *Given an input $x \in \{0,1\}^n$, output $y \in \{0,1\}$ such that $y = 0$ if $|x| \equiv 0 \pmod 3$, and $y = 1$ otherwise.*

For our purposes, the key feature of Mod 3 problem is that it is hard to solve for $\mathsf{AC}^0[2]$ circuits, as shown by Smolensky [108]:

Theorem 94 (Mod 3 is not in $\mathsf{RelAC}^0[2]$). *Any $\mathsf{AC}^0[2]$ circuit of depth d that computes the Mod 3 function (Problem 93) must have size $\exp\big(\Omega(n^{\frac{1}{2d}})\big)$.*

From this worst-case lower bound it is not too hard to obtain an average-case lower bound.

Lemma 95 (Average-case lower bound for Mod 3). *There exists an input distribution on which any $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit of depth d and size at most $\exp\big(n^{\frac{1}{10d}}\big)$ only solves the Mod 3 function (Problem 93) with probability $\frac{1}{2} + \frac{1}{n^{\Omega(1)}}$.*

Proof. Toward a contradiction, assume that for all input distributions there exists an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit of depth d and size $\exp\big(n^{\frac{1}{10d}}\big)$ that solves the Mod 3 problem with probability $1/2 + \varepsilon$ for $\varepsilon = 1/n^{o(1)}$.

Then by Yao's minimax principle, there exists a probability distribution over $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuits, or equivalently over $\mathsf{AC}^0[2]$ circuits, that solves the Mod 3 problem with probability $1/2+\varepsilon$ on every input. By sampling $O(1/\varepsilon^2)$ $\mathsf{AC}^0[2]$ circuits from this probability distribution and taking the majority vote of their outcomes, we get a new $\mathsf{AC}^0[2]$ circuit that solves the Mod 3 problem with probability at least 0.99 on every input. Now $1/\varepsilon^2 = n^{o(1)}$, and it is easy to construct a depth-d circuit of size $\exp(m^{O(1/d)})$ to compute the majority of m variables [61]. Hence this majority circuit has depth d and size $\exp(n^{o(1/d)})$, which doubles the depth and does not significantly increase the size of our circuit.

Now we can amplify success probability 0.99 to $1 - \exp(-n)$ by again sampling $O(n)$ circuits that succeed with probability 0.99 and taking their majority vote. This majority vote can be performed in AC^0, since we only need to perform an approximate majority as constructed by Ajtai and Ben-Or [7].

Since this distribution over $\mathsf{AC}^0[2]$ circuits fails with probability less than 2^{-n}, there exists one circuit in the distribution that works for all inputs. This yields an $\mathsf{AC}^0[2]$ circuit of depth $2d + O(1)$ and size $\exp\left(n^{\frac{1}{10d}}\right)$ computing Mod 3, which contradicts Theorem 94. □

We can finally use this average-case bound to show our lower bound for the Parity Bending Problem.

Theorem 96 (PBP is not in $\mathsf{RelAC}^0[2]$). *There exists an input distribution on which any $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit of depth d and size at most $\exp\left(n^{\frac{1}{10d}}\right)$ only solves PBP_n with probability $\frac{1}{2} + \frac{1}{n^{\Omega(1)}}$.*

Proof. Observe that any $\mathsf{AC}^0[2]$ circuit which solves the Parity Bending Problem with some probability can be extended to one that solves the Mod 3 problem with the same probability by adding a final parity gate over the output of the original circuit. The result then follows from Lemma 95. □

5.6.3 Parallel Parity Bending Problem

Now our goal is to establish Theorem 61, which strengthens the separation shown above. In this section we will consider input and output strings over the ternary alphabet $\{0,1,2\}$. Since we are talking about Boolean circuits manipulating these symbols, we actually mean that we encode these trits in binary using some canonical encoding, e.g., $\{0 \mapsto 00, 1 \mapsto 01, 2 \mapsto 11\}$. For a vector $x \in \{0,1,2\}^n$ we denote by $|x| = \sum_{i=1}^{n} x_i$.

To simplify our lower bounds, we modify the Parity Bending Problem to accept inputs drawn from $\{0,1,2\}^n$ when we move to the parallel version of the problem.

Problem 97 (k-Parallel Parity Bending Problem). *Given inputs $x_1, \ldots, x_k$ with $x_i \in \{0,1,2\}^n$ for all $i \in [k]$, produce outputs $y_1, \ldots y_k \in \{0,1\}^n$ such that y_i satisfies:*

$$|y_i| \equiv 0 \pmod 2 \text{ and } |x_i| \equiv 0 \pmod 3 \text{ or} \tag{5.38}$$

$$|y_i| \equiv 1 \pmod 2 \text{ and } |x_i| \not\equiv 0 \pmod 3 \tag{5.39}$$

for at least a $\frac{2}{3} + 0.05$ fraction of the $i \in [k]$.

This problem is k copies of a problem very similar to PBP, except that the inputs are now in $\{0,1,2\}$ instead of being in $\{0,1\}$. But the quantum algorithm described in Section 5.6.1 can be easily modified to work in this case, by applying a controlled gate that applies the phase $\exp(2\pi i/3)$ when the control and target are 11, and applies the phase $\exp(4\pi i/3)$ when the control and target are 21. By using this strategy for each individual gate, we get a $\mathsf{QNC}^0/\mathsf{qpoly}$ circuit which solves any given instance of the Parity Bending Problem with probability at least 3/4. Since the Parallel Parity Bending Problem only requires $2/3 + 0.05$ of the instances to be solved correctly, by using this quantum strategy for each instance independently, the quantum circuit solves the Parallel Parity Bending Problem with probability $1-o(1)$. This establishes the quantum upper bound in Theorem 61.

We now show that this problem is hard for $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuits. We start by introducing a related problem:

Problem 98 (3 Output Mod 3). *Given an input $x \in \{0,1,2\}^n$, output a trit $y \in \{0,1,2\}$ such that $y \equiv |x| \pmod 3$.*

As expected, an $\mathsf{AC}^0[2]$ circuit cannot solve this problem. In fact, on the uniform distribution, an $\mathsf{AC}^0[2]$ circuit succeeds with probability close to 1/3, which is trivially achieved by a circuit that just outputs 0. The easiest way to see this is by using random self-reducibility.

Lemma 99 (Worst case to average case). *Suppose there is an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit C of size S and depth d that solves Problem 98 on a uniformly random input with probability $1/3 + \varepsilon$ for some ε. That is,*

$$\Pr_{x \in \{0,1,2\}^n}[C(x) \equiv |x| \pmod 3] = \frac{1}{3} + \varepsilon.$$

Then there exists an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit C' of depth $d + O(1)$ and size $S + O(n)$ such that for any $x \in \{0,1,2\}^n$,

$$\Pr[C'(x) \equiv |x| \pmod 3] = \frac{1}{3} + \varepsilon, \quad \text{and} \tag{5.40}$$

$$\Pr[C'(x) \equiv |x| + 1 \pmod 3] = \Pr[C'(x) \equiv |x| + 2 \pmod 3] = \frac{1}{3} - \frac{\varepsilon}{2}. \tag{5.41}$$

Proof. Although AC^0 circuits cannot compute $|x| \bmod 3$ for a ternary input string $x \in \{0,1,2\}^n$, it is possible for an AC^0 circuit to sample a uniformly random vector $b \in \{0,1,2\}^n$ and its Hamming weight mod 3, $|b| \bmod 3$, as follows.

Sample random trits[7] $c \in \{0,1,2\}^n$ and set $b_i = c_{i+1} - c_i$ for all $1 \le i \le n-1$ and $b_n = -c_n$. We claim that b is a uniformly random string over $\{0,1,2\}^n$. To see this, observe that b_n equals a uniformly random trit, and furthermore for every $i \in [n-1]$, b_i is uniformly random over $\{0,1,2\}$, even conditioned on $c_{i+1}, \dots, c_n$, and therefore b_i is also uniformly random conditioned on $b_{i+1}, \dots, b_n$. It follows that b is a uniformly random string over $\{0,1,2\}^n$. However, the advantage of this method of producing a random string (as opposed to simply sampling a random string) is that we know $|b|$ since

$$|b| \equiv \sum_{i=1}^{n} b_i \equiv \sum_{i=1}^{n-1} (c_{i+1} - c_i) - c_n \equiv -c_1 \pmod 3.$$

[7] Technically, an AC^0 circuit cannot sample a trit with probability exactly 1/3, but the probability can be made exponentially close to 1/3.

Now we use a random self-reduction to construct the claimed circuit C'. The circuit C' first samples a random $a \in \{1, -1\}$ and a random $b \in \{0, 1, 2\}^n$ with known Hamming weight $|b|$, as described above. Then let C' output

$$C'(x) := \frac{C(a \cdot x + b \bmod 3) - |b|}{a} \pmod 3.$$

It is clear that $|a \cdot x + b| \equiv a|x| + |b| \bmod 3$, and that $a \cdot x + b$ is uniformly random regardless of a and x, so we have

$$\begin{aligned} \Pr_{a,b}[C'(x) \equiv |x| + k \pmod 3] &= \Pr_{a,b}[C(a \cdot x + b) \equiv a|x| + |b| + ak \pmod 3] && (5.42)\\ &= \Pr_{a,b}[C(a \cdot x + b) \equiv |a \cdot x + b| + ak \pmod 3] && (5.43)\\ &= \Pr_{y \in \{0,1,2\}^n, a}[C(y) \equiv |y| + ak \pmod 3]. && (5.44) \end{aligned}$$

In particular, when $k = 0$ we have

$$\Pr_{a,b}[C'(x) \equiv |x| \pmod 3] = \Pr_{y}[C(y) \equiv |y| \pmod 3] = \frac{1}{3} + \varepsilon.$$

When $k \neq 0$, we observe that $ak \neq 0$ is uniformly random and independent of y, so

$$\Pr[C'(x) \equiv |x| + 1 \pmod 3] = \Pr[C'(x) \equiv |x| + 2 \pmod 3] = \frac{1}{3} - \frac{\varepsilon}{2}.$$

Observe that the sampling circuit above is of constant depth and linear size. The modular arithmetic performed in section 5.6.3 can be performed by a constant-sized circuit. Overall this reduction increases the depth by a constant and the size by $O(n)$. □

Lemma 100 (Average-case lower bound). *An* $\mathsf{AC}^0[2]/\mathsf{rpoly}$ *circuit of depth d and size at most* $\exp(n^{\frac{1}{10d}})$ *solves Problem 98 on a uniform distribution with probability at most* $\frac{1}{3} + \frac{1}{n^{\Omega(1)}}$.

Proof. Let A be the circuit that solves Problem 98 on the uniform distribution with probability $\frac{1}{3} + \varepsilon$. By the worst-case to average-case reduction in Lemma 99, we get an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit (of similar size and depth) that succeeds with probability $\frac{1}{3} + \varepsilon$ on every input, and outputs each wrong answer with probability $\frac{1}{3} - \frac{\varepsilon}{2}$.

Construct a circuit for the Mod 3 Problem (Problem 93) such that on input x,

- with probability $\frac{1}{4}$, it outputs 0,
- with probability $\frac{3}{4}$, it outputs 0 if $A(x) = 0$ and 1 otherwise.

If $|x| \bmod 3 = 0$ then the circuit outputs 0 w.p. $\frac{1}{4} + \frac{3}{4}(\frac{1}{3} + \varepsilon) = \frac{1}{2} + \frac{3}{4}\varepsilon$. If $|x| \bmod 3 \neq 0$ then the circuit outputs 1 w.p. $\frac{3}{4}(\frac{1}{3} + \varepsilon + \frac{1}{3} - \frac{\varepsilon}{2}) = \frac{1}{2} + \frac{3}{8}\varepsilon$. In other words, the circuit solves the Mod 3 Problem with probability at least $\frac{1}{2} + \frac{3}{8}\varepsilon$ on arbitrary inputs. By Lemma 95, this implies $\varepsilon = \frac{1}{n^{\Omega(1)}}$. □

From this we get the following corollary.

Corollary 101. *Let C be an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit outputting a trit. For any fixed $x \in \{0,1,2\}^n$, we denote by $C(x)$ the random variable giving the output of the randomized circuit C on input x. Then, for all $i \in \{0,1,2\}$*

$$\frac{1}{3} - n^{-\Omega(1)} \le \Pr_{x \in \{0,1,2\}^n}\left[C(x) - |x| \equiv i \pmod 3\right] \le \frac{1}{3} + n^{-\Omega(1)}. \tag{5.45}$$

Proof. Since

$$\sum_{i=0}^{2} \Pr\left[C(x) - |x| \equiv i \pmod 3\right] = 1, \tag{5.46}$$

it suffices to show

$$\Pr\left[C(x) - |x| \equiv i \pmod 3\right] \le \frac{1}{3} + n^{-\Omega(1)} \tag{5.47}$$

for $i \in \{0,1,2\}$. For $i = 0$ this claim is exactly the statement of Lemma 100. For $i \in \{1,2\}$ we note a circuit satisfying

$$\Pr\left[C(x) - |x| \equiv i \pmod 3\right] \ge \frac{1}{3} + n^{-o(1)} \tag{5.48}$$

can be converted to one violating Lemma 100 by subtracting i from every output. □

We need an analog of Vazirani's XOR Lemma for finite groups [91, Lemma 4.2].

Lemma 102 (XOR lemma for finite abelian groups). *Let X be a distribution on a finite abelian group G such that $|\mathbb{E}[\psi(X)]| \le \epsilon$ for every non-trivial character ψ. Then X is $\epsilon\sqrt{|G|}$ close (in statistical distance) to the uniform distribution over G.*

We now consider the parallel version of the previous problem and show that it is hard.

Problem 103 (k-Parallel 3 Output Mod 3). *Given inputs $x_1, \ldots, x_k \in \{0,1,2\}^n$ for all $i \in [k]$, output a vector $\vec{y} \in \{0,1,2\}^k$ such that*

$$y_i \equiv |x_i| \pmod 3 \tag{5.49}$$

for at least a $\frac{1}{3} + 0.01$ fraction of the $i \in [k]$.

Theorem 104. *There exists a $k \in \Theta(\log n)$ for which any $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit solves the k-Parallel 3 Output Mod 3 Problem (Problem 103) with probability at most $n^{-\Omega(1)}$.*

Proof. Our proof is be similar to that of Theorem 61: we will first prove that solving the sum of any subset of subgames is hard, and then apply Lemma 102 to deduce that winning more than $1/3 + 0.01$ fraction of the games is hard.

Let C be an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit trying to solve Problem 103. Let $y_1, \dots, y_k$ be its k output trits. We consider the distribution X over k trits defined by

$$\bigotimes_{i=1}^{k} |x_i| - y_i \pmod 3$$

for a uniform input $x \in \{0,1,2\}^k$. We shall show that the distribution X is close to the uniform distribution over $\{0,1,2\}^k$.

Let χ_a be a non-trivial character of $\mathbb{F}_3^k$. That is $\chi_a(z) = \omega^{\sum_{i=1}^k a_i z_i}$ where ω is a third root of unity, and $a \in \mathbb{F}_3^k$. To show that X is close to uniform it suffices to show that the expectation of $\chi_a(X)$ is small for all non-zero vectors of coefficients $a_1, \dots, a_k \in \{0,1,2\}$.

Given $a \in \{0,1,2\}^k$ let S be the support of a, i.e., the set of indices on which $a_i \neq 0$. Given strings $x_1, \dots x_k \in \{0,1,2\}^n$ finding trits $y_1, \dots y_k$ satisfying

$$\sum_{i \in S} a_i |x_i| \equiv \sum_{i \in S} a_i y_i \pmod 3 \tag{5.50}$$

is at least as hard as solving the 3 Output Mod 3 Problem on the concatenated input $(a_i x_i)_{i \in S}$. This is true since any circuit solving the former can be converted into a circuit solving the latter by adding an depth-2 circuit with $\exp(|S|) \leq \exp(k) \leq \mathsf{poly}(n)$ gates that adds the $|S|$ trits $a_i y_i$ modulo 3. However, the concatenated input $(a_i x_i)_{i \in S}$ is a uniform vector in $\{0,1,2\}^{n|S|}$. Corollary 101 then gives that

$$\sum_{i \in S} a_i(|x_i| - y_i) \pmod 3 \tag{5.51}$$

has an ℓ_1 distance at most $n^{-\Omega(1)}$ from the uniform distribution over $\{0,1,2\}$. Thus, $|\mathsf{E}[\chi_a(X)]| \leq n^{\Omega(1)}$.

Applying Lemma 102 with $G = \mathbb{Z}_3^k$ and X the distribution of the random variables $\bigotimes_{i=1}^k |x_i| - y_i \pmod 3$ gives that X has ℓ_1 distance

$$n^{-\Omega(1)} \cdot \sqrt{3^k} \tag{5.52}$$

from the uniform distribution on $\{0,1,2\}^k$.

To finish the proof, we note that by Chernoff's bound, the probability of drawing a string from the uniform distribution over $\{0,1,2\}^k$ with more than a $\frac{1}{3}$+0.01 fraction of its outputs 0 is bounded by $\exp(-\Omega(k))$. Then we see the probability of drawing a string from X with more than $\frac{1}{3} + 0.01$ fraction of zeros is bounded above by

$$n^{-\Omega(1)} \cdot \sqrt{3^k} + e^{-\Omega(k)} = n^{-\Omega(1) + k/(2\log_3(n))} + e^{-\Omega(k)}. \tag{5.53}$$

To complete the proof, we note there is some $k \in \Theta(\log n)$ for which the above sum is bounded above by $n^{-\Omega(1)}$. □

Theorem 105. *There exists a $k \in \Theta(\log n)$ for which an $\mathsf{AC}^0[2]/\mathsf{rpoly}$ circuit succeeds on the k-Parallel Parity Bending Problem with probability at most $n^{-\Omega(1)}$.*

Proof. We show a circuit solving the Parallel Parity Bending Problem can be reduced to one solving Problem 103 with success probability close to $\frac{1}{2}$ the original success probability.

This reduction is straightforward : given a solution $y_1, \ldots y_k$ to Problem 97 we convert to a solution $y'_1, \ldots, y'_k$ to Problem 103 by setting $y'_i = 0$ if $|y_i| = 0 \pmod 2$, and y'_i equal to a random choice of 1 or 2 if $|y_i| = 1 \pmod 2$. The expected number of successes is at least half the number of successes in the original instance, since a success on input i with $|x_i| \equiv 0 \pmod 3$ remains a success with probability 1, and a success on an input i with $|x_i| \pmod 3 \in \{1, 2\}$ remains a success with probability $\frac{1}{2}$. Concentrating around this value completes the proof. □

This establishes the classical lower bound in Theorem 61.

Chapter 6

Low-Depth Quantum Circuits: Interactive Simulation

6.1 Introduction

This is the second of the two chapters on low-depth quantum circuits. Shallow circuits are a good place to look for a quantum advantage, since the current error rates of quantum devices cripple their ability to implement deep quantum circuits. Separately, low-depth quantum circuits are also studied for their connection to low-depth classical circuits, which have proven a fruitful area of study. All of our arguments for low-depth circuits can be found at the start of Chapter 5.

The two chapters share a connection to a recent result of Bravyi, Gosset, and König [26] which offers the first practical, unconditional quantum advantage, between RelQNC^0 and RelNC^0. The relation problem they introduce is called the "2D Hidden Linear Function Problem" (2D HLF), and it can be implemented by a quantum circuit of constant depth, using only classically controlled Clifford gates, between adjacent qubits on a grid (simultaneously hitting all of our themes). Furthermore, they show that a comparable classical circuit cannot solve 2D HLF, without making any assumptions.

This work spurred a number of follow-up papers [38, 76, 27, 120], and two in particular motivate this work: Bravyi, Gosset, König, and Tomamichel [27], and Bene Watts, Kothari, Schaeffer, Tal [120] (which is the subject of the previous chapter). In [27], the authors give a new problem solvable by a one-dimensional (i.e., the qubits are arranged in a line) constant-depth quantum circuit, but remains impossible for classical constant-depth circuits. Then they show how the problem (and others like it) can be transformed to one solvable by *noisy* quantum circuits with high probability, albeit on a 3D array of qubits. Bene Watts et al. show that 2D HLF is hard for a slightly more powerful family of circuits, AC^0, consisting of constant-depth circuits of polynomial size with *unbounded* fan-in AND and OR gates. This result is also unconditional.

The goal of this chapter is to further investigate the complexity of 2D HLF and related problems, and offer new hard problems which minimize the dimensionality, number of qubits, circuit depth, etc. as much as possible. The main result is an interactive task which can be solved by a constant-depth classically-controlled Clifford circuit, but which we argue is $\oplus\mathsf{L}$-hard to simulate classically[1]. Note that for our purposes, *simulation* means returning any string of measurement outcomes which occurs with nonzero probability in a genuine execution of the circuit, resulting in a relation problem (output any string satisfying the above relation) rather than a sampling problem. Unfortunately, we conjecture that there is no black-box hardness reduction to *prove* that these problems are classically hard. That is, we believe a quantum device executing the circuits does not help solve any classically hard problem, but a *classical simulation* of the circuit does require the power of NC^1 or $\oplus\mathsf{L}$.

To leverage the difference between classical and quantum simulation, we must slightly strengthen our notion of simulation to be *interactive*. Specifically, we are interested in a task where the quantum device (or classical simulator) is told to implement a constant-depth Clifford circuit and measure all but $O(1)$ qubits. Then in a separate second round of interaction, once the device has revealed (or at least committed to) the measurement outcomes, it is told how to measure the remaining $O(1)$ qubits. We believe this is a reasonable task for a quantum device, in the sense that if the device can execute constant-depth circuits then all we are asking is for a tiny portion of the qubits to be measured apart from the rest.[2] Under this interactive model, we use the fact that we can *rewind* the classical simulator to an earlier state, an idea common in security proofs for (interactive) cryptographic protocols. The ability to rewind is not *physical*, and is the key that separates a quantum device from a classical simulation, which allows us to prove the main hardness result.

6.1.1 Proof Outline and Main Ideas

We give two hardness results based on the same general framework:

1. Simulating a sequence of Clifford gates is computationally hard, in the sense that the final quantum state contains the solution to a hard decision problem.

2. Using ideas from measurement-based quantum computation, we can collapse a sequence of Clifford gates to a constant-depth circuit, but we introduce Pauli errors on the final quantum state.

[1] We also have an NC^1-hard interactive task, which is weaker than the $\oplus\mathsf{L}$ result since $\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \oplus\mathsf{L}$, but the problem has other advantages like error correction, and weaker topological requirements on the qubits/gates.

[2] A fair objection is that the qubits of the device could have a limited lifespan, and then a device capable of executing constant-depth circuits might fail if there is too much delay between the rounds of interaction. For example, if the second round input is determined by a lengthy computation on the first round output, then it might delay the second round long enough that the qubits expire and the quantum device fails. One can define the task such that the delay is short, say by requiring that the second round input only depends on the first round in a way that can be computed *very* efficiently, e.g., in constant depth. It turns out that in our hardness reductions, we *ignore* the first round output (i.e., it is only important that the simulator/device commits to a string, not the string itself) and choose the second round input at random.

3. Make the problem interactive, so that the classical simulator (but *not* the quantum device) gains the ability to rewind.

4. Show how the classical simulator can recover information about the quantum state by rewinding, even though a quantum device cannot learn anything.

5. Use random self-reducibility to repeat the process and accumulate enough information about the state to solve the decision problem.

The first result (in Section 6.4) proves NC^1-hardness for a classical simulating a quantum device as it measures a cluster state on $2 \times n$ qubits. This is the conceptually easier of the two results, so we put it first to introduce the techniques.[3] The second result (see Section 6.5) proves $\oplus\mathsf{L}$-hardness for a classical simulation of measuring a state on an $\mathsf{poly}(n) \times \mathsf{poly}(n)$ grid of qubits. Let us walk through the main proof ideas one at a time.

We build our hardness results on the difficulty of classically-controlled Clifford circuits, i.e., a sequence of classically-controlled Clifford gates as described in Chapter 2. That is, there is a fixed sequence of Clifford gates and a classical input which tells us whether or not to apply each gate in the sequence, much like a branching program. The problem is to simulate (i.e., output a string of measurement outcomes with nonzero probability) the sequence on a fixed input state, and clearly the difficulty of this problem varies depending on the number of qubits and the gate set (we may take only a subgroup of the Clifford gates). We show that simulating a sequence of Clifford gates on two qubits is NC^1-complete, by first characterizing the group of two-qubit Clifford gates and applying a result of Barrington and Thérien [13]. We also show that simulating a sequence of CNOT, CSIGN, and R_Z gates on n-qubits is $\oplus\mathsf{L}$-hard, by a simple reduction from known $\oplus\mathsf{L}$-hard problems, e.g., in Damm [39] (recall Section 2.4). On the other hand, we slightly extend a Clifford simulation result of Aaronson and Gottesman [4] to claim that classically controlled Clifford gates can be simulated in $\oplus\mathsf{L}$ (even sampled, assuming randomness is available).

However, the simulation problems outlined above are unsuitable because the natural quantum implementation (i.e., literally executing a sequence of classically-controlled Clifford gates, then measuring) is *not constant depth.* We need to adjust the problem to make it solvable in constant depth.[4] We use an observation of Raussendorf, Browne, and Briegel [94] that their procedure for measurement-based quantum computation works without adaptivity for Clifford circuits. Ordinarily their technique introduces errors in the form of Pauli operators (they call them *byproduct operators*), which must be fixed adaptively, with a layer of adaptivity (or more) for

[3] The NC^1 result is not just included for presentation; the result is also on a one-dimensional grid of qubits, uses far fewer qubits per simulated gate, and the hardness result carries over when the classical simulator can err with some small probability.

[4] Interestingly, it is known that any Clifford circuit has a constant depth implementation (using $\tilde{\Theta}(n^2)$ ancillas) using unbounded fan-in parity gates (or equivalently, fan-out gates). This is an easy consequence of Moore and Nilsson [84], where they show how to implement CNOT circuits in constant depth (actually $\Theta(\log n)$-depth, but only because they insist on expanding the unbounded fan-in gates as trees of constant fan-in gates), and the Aaronson-Gottesman decomposition of Clifford circuits into constantly many layers of single qubit gates and CNOT circuits.

each layer of the original circuit. When all the gates are Clifford, however, the Pauli errors can by pushed (via conjugation relations) through the gates to the end. In principle, the final Pauli errors can be computed (in $\oplus\mathsf{L}$), but since this is too expensive for our hardness reductions, the Pauli error is unknown to us.

Interactivity is a critical component of our result. Recall that we want to find some task which a constant-depth quantum circuit can solve, but such that a classical simulator for the task solves hard problems. One way to prove this would be to show that *any* oracle for the task, quantum or classical, can be used to solve computationally hard problems. Alas, we conjecture the exact opposite: a constant-depth quantum circuit cannot perform any computation that would help solve NC^1-complete problems, let alone $\oplus\mathsf{L}$-complete problems. Our hardness proof must therefore use the fact that the simulator is classical at some point. We achieve this by making the task interactive, which gives us the following key difference. In between rounds of the protocol, one can look inside the device, quantum or classical, at its internal state. Quantum state famously cannot be copied or cloned, but classical state can, meaning that, for classical devices only, one can save the internal state and restore it at a later time. This immediately gives any classical simulator the additional power to *rewind*, in our case from the end of the interaction to beginning of the second round. With this power, we can measure (in simulation) the same $O(1)$-qubit state many times, and even in different bases.

The power to measure a state repeatedly, under multiple different bases, would seem to be sufficient to learn the state by tomography. However, recall that we define simulation to be returning measurement outcomes that are *possible* (i.e., occur with nonzero probability), rather than sampling measurement outcomes. In particular, the classical simulator may be designed adversarially to thwart our attempt to learn the state, or prove a hardness result in general. Standard tomography results depend on measurement outcomes being random, not adversarial, so they break under this model. Instead, we show we can use measurements from the magic square game [81] to force the classical simulator measure both outcomes for some Pauli measurement. This proves the state is not stabilized by the Pauli operator (or its negation), which rules out a small number of Clifford states. In this way, we can learn a small piece of information about the state, even if the measurement outcomes are chosen adversarially.[5]

Finally, we have the ability to learn a non-stabilizing Pauli operation for the state, but it could be adversarially chosen. In particular, if we need to decide whether a state is, e.g., $|00\rangle$ or $|{+}{+}\rangle$, then there are many choices of Pauli operators which do not distinguish the two (e.g., $\mathtt{YI}$ stabilizes neither).

[5] Alternatively, one could define the task to simulate in the sampling sense and use conventional tomography to skip this step. However, the result would be weaker (any kind of error-free sampling would imply our relational simulation) and it is somewhat difficult to define sampling so that it interacts correctly with interactivity and rewinding. To make it work, we need the second round to take fresh bits of randomness (so they can be changed after rewinding) and sample a random measurement outcome each time. Weaker notions of sampling will not suffice.

6.2 Background

In this section, we introduce the necessary background that has not already been covered in Chapter 2. We will briefly discuss how the Clifford group interacts with phase, and the rest of the section is an introduction to measurement-based quantum computation, and a lengthy analysis via the ZX-calculus.

6.2.1 Clifford modulo Pauli, and Pauli modulo Phase

Due to the limitations of measurement based quantum computation (which we describe in more detail in the next section), our results will usually implement Clifford operations up to a Pauli correction. That is, instead of implementing $C \in \mathcal{C}_m$, we implement CP for some $P \in \mathcal{P}_m$. In principle, we can compute P from the measurement outcomes, and for a fixed circuit it is always an affine function of the measurements. However, in all the cases we are trying to prove hardness, the problem of computing P is just as hard as the problem we are trying to reduce to, so we are unable to compute P in the reduction. Instead, we must be content to perform any Clifford operation in the same coset as the one we asked for, and we must talk about Clifford operations, Clifford states, and Pauli stabilizers *modulo* $\mathcal{P}_m$. Naturally, this leads us to work with several quotient groups.

Since $\mathcal{P}_m$ is normal in $\mathcal{C}_m$, the quotient group $\mathcal{C}_m/\mathcal{P}_m$ is well-defined. When we assert that two Clifford operations $C_1, C_2 \in \mathcal{C}_m$ are *equivalent modulo Paulis*, or write $C_1 \equiv C_2 \pmod{\mathcal{P}_m}$, we really mean that the cosets $C_1\mathcal{P}_m$ and $C_2\mathcal{P}_m$ are equal, or equivalently, $C_1C_2^{-1} \in \mathcal{P}_m$. Another way to characterize equivalent Clifford operations is by their action (by conjugation) on the Paulis.

Lemma 106. *Define the homomorphism $\phi_C{:}\mathcal{P}_m \to \mathcal{P}_m$ where $\phi_C(Q) := C \bullet Q$ for $C \in \mathcal{C}_m$ and $Q \in \mathcal{P}_m$. Two Clifford operations $C_1, C_2 \in \mathcal{C}_m$ are equivalent modulo Paulis if and only if their action on $\mathcal{P}_m/\mathcal{Z}_m$, i.e., $\phi_{C_1}(Q) \equiv \phi_{C_2}(Q) \pmod{\mathcal{Z}_m}$ for all $Q \in \mathcal{P}_m$.*

Proof. Let $C \in \mathcal{C}_m$ be a Clifford operation. First, it is clear that the map $\phi_C{:}\mathcal{P}_m \to \mathcal{P}_m$ such that $\phi_C(Q) := C \bullet Q$ is a homomorphism. Since $\mathcal{Z}_m$ commutes with C, we have $\phi_C(\mathcal{Z}_m) = \mathcal{Z}_m$. Hence, modding out by $\mathcal{Z}_m$ gives $\tilde{\phi}_C{:}\mathcal{P}_m/\mathcal{Z}_m \to \mathcal{P}_m/\mathcal{Z}_m$. The formal claim is that C_1 and C_2 are equivalent modulo $\mathcal{P}_m$ if and only if $\tilde{\phi}_{C_1} = \tilde{\phi}_{C_2}$. This reduces to checking that $C = C_1C_2^{-1}$ is equivalent to $\mathtt{I}\cdots\mathtt{I}$ (i.e., C is in $\mathcal{P}_m$) if and only if $\tilde{\phi}_C = \tilde{\phi}_{\mathtt{I}\cdots\mathtt{I}}$ (i.e., if $\tilde{\phi}_C$ is the identity permutation).

It is a fact that $PQP^\dagger = \pm Q$ for any $P, Q \in \mathcal{P}_m$. Hence, if $C \in \mathcal{P}_m$ then $\phi_C(Q) = \pm Q$, so $\tilde{\phi}_C$ is the identity permutation. This means there are at least 4^m Clifford operations C (i.e., the Pauli operations) for which $\tilde{\phi}_C$ is the identity. In the tableau representation, these operations all have the same $2m \times 2m$ matrix, and only differ in the phase bits. Since we know $2m$ phase bits suffice, there are at most $2^{2m} = 4^m$ such Clifford operations, so they must be exactly the Pauli operations. □

Next, we want an analogous notion of equivalence for Clifford states modulo $\mathcal{P}_m$. We say $|\psi_1\rangle \equiv |\psi_2\rangle \pmod{\mathcal{P}_m}$ if there exists P such that $|\psi_2\rangle = P|\psi_1\rangle$. It is not hard to check that this is an equivalence relation, and that equivalence is preserved under equivalent Clifford operations.

Lemma 107. *Two Clifford states $|\psi_1\rangle$ and $|\psi_2\rangle$ are equivalent if and only if their stabilizer groups contain all the same Pauli operations up to sign. That is, if P is in* $\mathrm{Stab}_{|\psi_1\rangle}$ *then $\pm P$ is in* $\mathrm{Stab}_{|\psi_2\rangle}$.

Proof. The calculation

$$(C \bullet P)C|\psi\rangle = CPC^\dagger C|\psi\rangle = CP|\psi\rangle = C|\psi\rangle$$

shows that $C \bullet P$ is a stabilizer of $C|\psi\rangle$ if P is a stabilizer of $|\psi\rangle$, where $C \in \mathcal{C}_m$ and $P \in \mathcal{P}_m$. So, if $|\psi_2\rangle = Q|\psi_1\rangle$ and P is a stabilizer of $|\psi_1\rangle$ then $Q \bullet P = QPQ^\dagger = \pm P$ is a stabilizer of $|\psi_2\rangle$. □

Finally, let us discuss Pauli measurement. Given a Pauli string $P \in \mathcal{P}_m/\mathcal{Z}_m$, the associated measurement projects onto the two eigenspaces of P and reports the corresponding eigenvalue, +1 or −1. For the vast majority of our applications, the property of Pauli measurement we use is that any Pauli string appearing in the stabilizer group has a deterministic measurement outcome, and all other Pauli measurements have inherently random outcomes. Thus, we will bend terminology and say $P \in \mathcal{P}_2/\mathcal{Z}_2$ is a *stabilizer* of $|\psi\rangle$ if P has a deterministic outcome on $|\psi\rangle$. Any other element of $\mathcal{P}_2/\mathcal{Z}_2$ is a *non-stabilizer*, and will have a uniformly random outcomes. We will also abuse notation and write, e.g., $|\psi\rangle = U|{+}{+}\rangle$ for $U \in \mathcal{C}_2/\mathcal{P}_2$, even though $|\psi\rangle$ is not a state, since U is a coset of unitaries.

These notions of equivalence will be important throughout the paper, so let us recap. Clifford operations C_1, C_2 are equivalent modulo Paulis if $C_1C_2^{-1}$ is a Pauli operation, and equivalent Clifford operations induce the same permutation of $\mathcal{P}_m/\mathcal{Z}_m$. Clifford states $|\psi_1\rangle$ and $|\psi_2\rangle$ are equivalent if they have the same stabilizer group up to signs, i.e., for any Pauli P in $\mathrm{Stab}_{|\psi_1\rangle}$, $\pm P \in \mathrm{Stab}_{|\psi_2\rangle}$.

6.2.2 Measurement-based computation

One of the main techniques used throughout this paper is a model of computation developed by Raussendorf, Browne, and Briegel known as *one-way quantum computation* [93, 94]. For reasons that will soon be clear, this is also sometimes called *measurement-based computation on cluster states.* At a high level, measurement-based computation allows for the simulation of any quantum computation by performing a sequence of adaptive single-qubit measurements on a certain highly-entangled initial state.

First, let us describe the initial state. Let $G = (V, E)$ be an undirected graph with n vertices. Define the *graph state* for G as

$$|G\rangle \equiv \sum_{x \in \{0,1\}^n} \prod_{(u,v) \in E} (-1)^{x_u x_v} |x\rangle .$$

Notice that any graph state can be constructed from the all zeros state by applying a Hadamard gate to each qubit, and then applying a CSIGN gate between each pair of qubits representing an edge. Thus, the any graph state can be constructed in depth equal to the maximum degree of the graph plus one (by edge coloring arguments).

A *cluster state* is a special case of a graph state where the graph is a 2D grid. Importantly, the cluster state can be constructed in constant depth, using at most 4 layers of CSIGN.

Measurement-based computation consists of a sequence of measurement operations that result in a gate-by-gate simulation of a quantum circuit. For a given gate U and an initial state $|\psi\rangle$, there is a measurement procedure that applies U to $|\psi\rangle$ by consuming a constant-size graph state $|G\rangle$:

1. Prepare the state $|\psi\rangle \otimes |G\rangle$.
2. Apply CSIGN between $|\psi\rangle$ and the leftmost grid points of $|G\rangle$.
3. Measure each input qubit of $|\psi\rangle$ and all but the rightmost grid points of $|G\rangle$ in either the Z-basis or the $\{|0\rangle \pm e^{i\theta}|1\rangle\}$-basis.

The unmeasured qubits after the above procedure will be in the state $PU|\psi\rangle$ where P is some Pauli string that depends on the measurement outcomes. To apply multiple gates, we simple string together multiple instances of the above procedure. Fortunately, each application of the CSIGN gate commutes with all previous measurements, so we can view the simulation of the entire circuit as a sequence of single-qubit measurements on some sufficiently large cluster state. Unfortunately, there are Pauli errors between each of the gates. Therefore, in general one must *adaptively* apply the measurements for one gate to compensate for the Pauli errors made in the application of the previous gate.

In this paper, we will focus only on quantum circuits which are composed of Clifford gates. Therefore, no such adaptivity is needed since all Pauli errors can be pushed to the end. We will also use that fact that every Clifford operation can be applied using only X, Y, and Z-basis measurements. This yields the following theorem:

Theorem 108 (Raussendorf, Browne, and Briegel [94]). *For any m-qubit Clifford circuit C with n local gates, there exists a pattern of X, Y, and Z single-qubit measurements on an $O(m) \times O(n)$ cluster state such that the unmeasured rightmost qubits of the cluster are in the state $PC|+\rangle^m$, where Pauli $P \in \mathcal{P}_{O(m)}$ is a function of the measurement outcomes.*

In the following, we will modify the constructions of Raussendorf, Browne, and Briegel to minimize the number of qubits we require. Although such constructions only save a constant fraction of qubits, we feel as though it is desirable to get the simplest possible circuit. One of the easiest ways to verify and explain these constructions is to appeal to the ZX-calculus of Coecke and Duncan [34]. We will not attempt to give a formal introduction to this topic, and instead refer the reader to their paper as it is quite clear and thorough. However, we will lay out the basics in the following section.

6.2.3 ZX-calculus

The ZX-calculus is a graphical language for reasoning about linear maps between qubits. It consists of a small set of generators (Table 6.1), which are graphical descriptions of states, operations, isometries, and projections, and a set of local replacement rules (Table 6.2). We only describe those generators and rules that we will use throughout this paper (and these are insufficient to completely describe all of quantum computation).

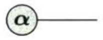
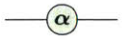
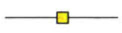
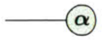
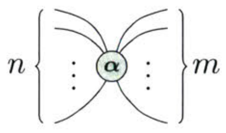
Type	Generator	Meaning
state		$\|0\rangle + e^{i\alpha}\|1\rangle$
unitary		$\|0\rangle\langle 0\| + e^{i\alpha}\|1\rangle\langle 1\|$
unitary		$\|+\rangle\langle 0\| + \|-\rangle\langle 1\|$
projection		$\langle 0\| + e^{i\alpha}\langle 1\|$
spider		$\|0\rangle^{\otimes m}\langle 0\|^{\otimes n} + e^{i\alpha}\|1\rangle^{\otimes m}\langle 1\|^{\otimes m}$

Table 6.1: Generators for the ZX-calculus. For each green generator listed above for the Z-basis, there is an analogous red generator for the X-basis. By convention, a solid green/red circle implies that α equals 0.

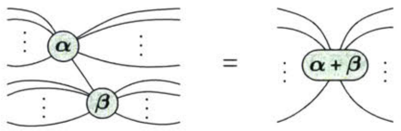
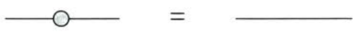
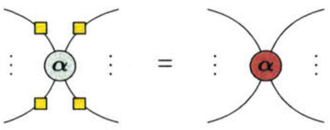
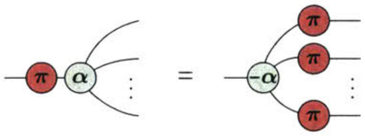
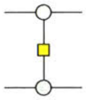
Rule name	Rule
Spider fusion	
Identity rule	
Color change	
π-copy	

Table 6.2: Rules for the ZX-calculus. In every rule, red and green nodes can be exchanged.

We will primarily use the ZX-calculus to determine the affect of the X and Y measurements on graph states.[6] First, one can verify that the following diagram represents a CSIGN gate:

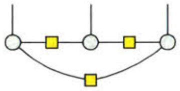
[6] Using the ZX-calculus to analyze measurement-based computation is not new. For instance, see references [34] and [43].



Using this fact and the spider fusion rule from Table 6.2, it is easy to prove the following lemma:

Lemma 109. *Every graph state $|G\rangle$ for $G = (V, E)$ is represented by a $|V|$-qubit ZX-calculus diagram where two qubits in the diagram are connected by edge with a Hadamard gate iff they share an edge in the graph G.*

For instance, the triangle graph on three vertices is represented by the diagram

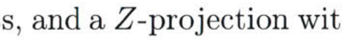


We now want to show the effect of X and Y measurements on the graph. In the subsequent discussion, we will actually refer to *projections*, rather than measurements. Let us briefly justify this choice. First, recall that Pauli measurement on a single qubit for Pauli $P \in \mathcal{P}_1$ projects the state onto either the +1 or −1 eigenspace of P. The key observation is that the net result of the measurement can always be described by applying a Pauli operator, projecting the qubit onto the +1 eigenspace, and applying another Pauli operator. There are two cases: if the Pauli measurement would have resulted in a projection into the +1 eigenspace, then do nothing; if it would have projected into the −1 eigenspace, then apply Pauli Q such that $Q \neq P$, project into the +1 eigenspace, and apply Pauli Q again. This is justified by the following equation:

$$\frac{\mathrm{I}-P}{2} = Q\left(\frac{\mathrm{I}+P}{2}\right)Q.$$

To be clear, in the following ZX-calculus diagrams we will make no explicit mention of the Pauli Q operation mentioned above, which is required to accurately describe the state if the measurement projects the state onto the −1 eigenspace. However, using the π-copy rule from Table 6.2, these phases can easily[7] be pushed to the end of the circuit. These phases determine the Pauli correction operation which appears in Theorem 108 (and, in fact, all measurement-based computation schemes).

With this out of the way, let us discuss measurements in the context of the ZX-calculus. To measure a qubit in the X-basis, we will apply the green projection generator in Table 6.1 to the corresponding qubit. To measure in the Y-basis, we first apply a $\pi/2$-phase gate to the corresponding qubit, and then project onto the X-basis. Since



by the spider fusion rule, we get that each measurement corresponds to a Z-projection with $\alpha = 0$ for X measurements, and a Z-projection with $\alpha = \pi/2$ for Y measurements.

[7] On the other hand, such calculations will not be "easy" for sufficiently weak machines, which in some sense is the basis for the hardness results of Section 6.4 and Section 6.5.

6.2.4 Single-qubit Clifford

This section will be devoted to proving the following theorem:

Theorem 110. *Let G be the line graph on $3n+2$ vertices. Given single-qubit Clifford gates $g_1, \ldots, g_n \in \mathcal{C}^1$, there exist projections $P_1, \ldots, P_{3n+1}$ such that*

- *P_i is either the projection $\frac{I+X}{2}$ or $\frac{I+Y}{2}$ onto the ith qubit,*
- *$P_1 \otimes \ldots \otimes P_{3n+1} \otimes I|G\rangle \propto |\psi\rangle \otimes \prod_{i=1}^{n} g_i |+\rangle$,*
- *P_1 is an X projection, and*
- *For $i \in [n]$, P_{3i-1}, P_{3i}, and P_{3i+1} depend only on $g_i \in \mathcal{C}_1$ and whether or not i is even.*

It is a standard result that every single-qubit gate can be decomposed into rotations $R_x(\theta_3)R_z(\theta_2)R_x(\theta_1)$, and for Clifford operations $\theta_i \in \{0, \pi/2, \pi, 3\pi/2\}$ for all i. Three single-qubit X and Y measurements on the line suffice to mimic such a decomposition using measurement-based computation (see e.g., [11]). We show this relationship in Figure 6-1.

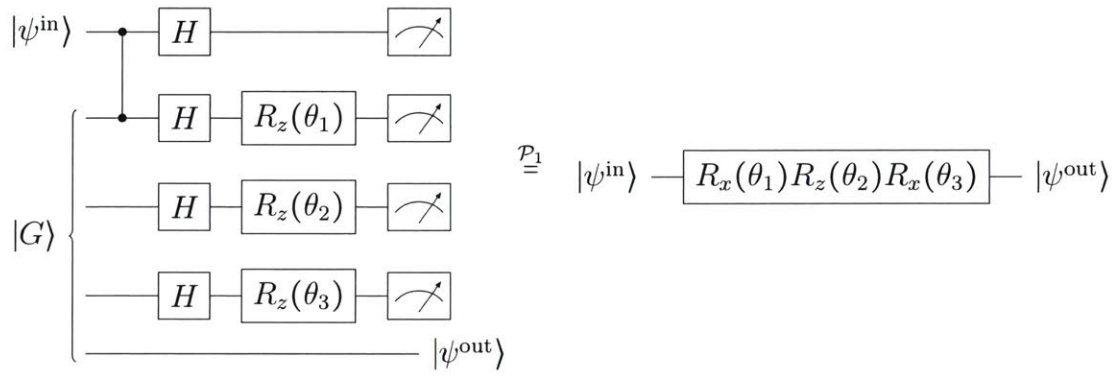
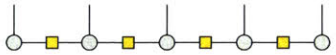
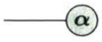
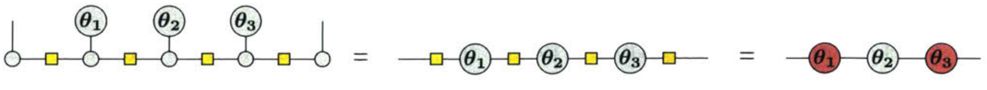

$|\psi^{\text{in}}\rangle$
H
H
$R_z(\theta_1)$
$|G\rangle$
H
$R_z(\theta_2)$
H
$R_z(\theta_3)$
$|\psi^{\text{out}}\rangle$
$\overset{\mathcal{P}_1}{=}$
$|\psi^{\text{in}}\rangle$
$R_x(\theta_1)R_z(\theta_2)R_x(\theta_3)$
$|\psi^{\text{out}}\rangle$

Figure 6-1: Generating any single-qubit Clifford by measuring graph state for line graph G on 4 vertices. The choice of basis in the middle three qubits, determines the gate applied to input $|\psi^{\text{in}}\rangle$ up to a Pauli correction.

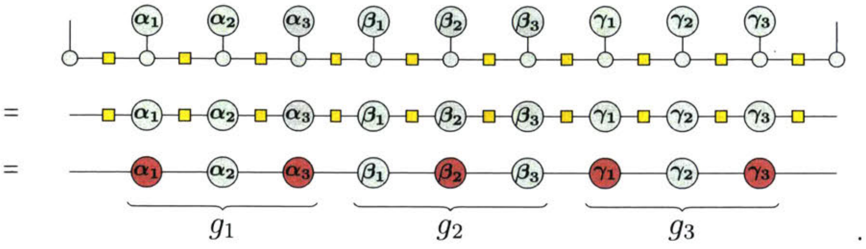
Let us prove this statement using the ZX-calculus. By Lemma 109, we can represent the 5-qubit line graph state as the following diagram in the ZX-calculus:



Recall that the generator for measurement is  , where $\alpha = 0$ for X measurement and $\alpha = \pi/2$ for Y measurement. Using this fact and the color changing rule from Table 6.2, we get



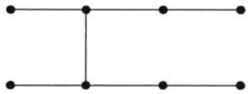
Suppose we wished to apply more than one single-qubit gate. We now only need to observe that any single-qubit gate can be decomposed into either $R_x(\theta_3)R_z(\theta_2)R_x(\theta_1)$ or $R_z(\theta_3)R_x(\theta_2)R_z(\theta_1)$ where θ_i depends on which type of decomposition is used. When using 3 qubits to generate a gate within a longer chain of qubits, the decompositions alternate. For example, the following diagram shows the application of gates $g_1 = R_x(\alpha_3)R_z(\alpha_2)R_x(\alpha_1)$, $g_2 = R_z(\beta_3)R_x(\beta_2)R_z(\beta_1)$, and $g_3 = R_x(\gamma_3)R_z(\gamma_2)R_x(\gamma_1)$ using a chain of length 11.



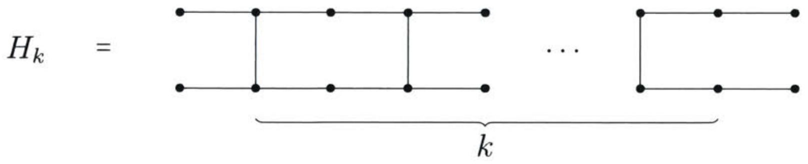
The generalization is clear, so this concludes the proof of Theorem 110.

6.2.5 Two-qubit Clifford

The analysis for the two-qubit case will be very similar to the single-qubit case, except that we will obviously require a more sophisticated initial graph state. The analogue to the line graph on five vertices will be the following graph on eight vertices:



We call this graph H_1 since it has one ⊏ graph as a subgraph. In general, we define the graph H_k to be the graph on $4k+4$ vertices with k such subgraphs:

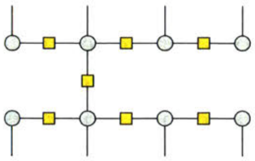


For convenience, we number the qubits of H_k from top to bottom, and then from left to right. We can now state the main theorem of this section:

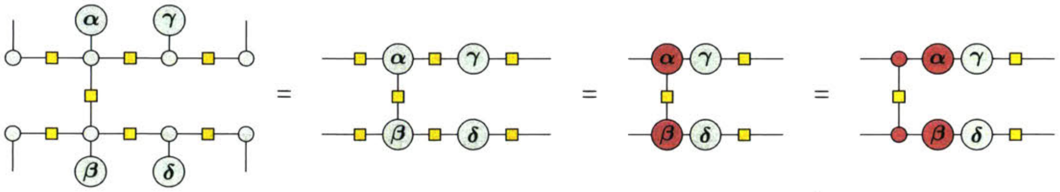
Theorem 111. *Given two-qubit Clifford gates* $g_1, \dots, g_n \in \mathcal{C}_2$, *there exist projections* $P_1, \dots, P_{16n+2}$ *such that*

- P_i *is either the projection* $\frac{I+X}{2}$ *or* $\frac{I+Y}{2}$ *onto the ith qubit,*
- $P_1 \otimes \dots \otimes P_{16n+2} \otimes II|H_{4n}\rangle \propto |\psi\rangle \otimes \prod_{i=1}^{n} g_i |{++}\rangle$,
- P_1 *and* P_2 *are* X *projections, and*
- *For* $i \in [n]$, $P_{16i-13}, \dots, P_{16i+2}$ *depend only on* $g_i \in \mathcal{C}_2$ *and whether or not* i *is equal to* n.

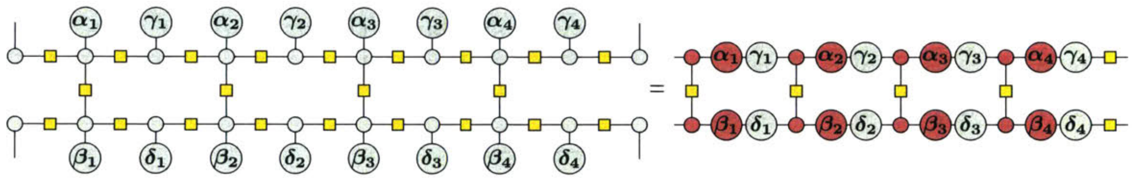
Once again, start by using the ZX-calculus to represent the graph state $|H_1\rangle$:



Using the same ideas as in the previous section, we get the following after measurements:



In other words, the measurements induce a circuit with $C(X,X)$, followed by single qubit gates $R_z(\gamma)R_x(\alpha) \otimes R_z(\delta)R_x(\beta)$, followed by $H \otimes H$. Recall that $C(X,X)$ is the analogue of the CSIGN gate for the X-basis. Notice that the extra $H \otimes H$ operation can be pushed into the next copy of the ⊏ gadget for larger $|H_k\rangle$ states. So, for example, the state $|H_4\rangle$ after measurement is the following:



This is example is particularly relevant given the following fact, which can be proved through computational brute force.

Fact 112. *Every two-qubit Clifford operation can be decomposed as*

$$\prod_{i=1}^{4}\left([R_z(\gamma_i)R_x(\alpha_i) \otimes R_z(\delta_i)R_x(\beta_i)]\mathrm{C}(\mathrm{X},\mathrm{X})\right).$$

for $\alpha_i, \beta_i, \gamma_i, \delta_i \in \{0, \pi/2, \pi, 3\pi/2\}$.

Therefore, if we have some large $|H_k\rangle$ state, we can measure 16 consecutive qubits to implement any two-qubit gate. The exception is the final 16 qubits which will have an extra $H \otimes H$ applied to the end. However, instead of setting the $\alpha_i, \beta_i, \gamma_i, \delta_i$ parameters to generate some gate g, we can set them to generate Hg. Combining the above observations completes the proof of Theorem 111.

6.2.6 Multi-qubit Clifford

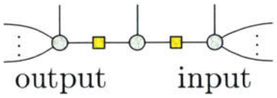
When reasoning about larger Clifford circuits, it will be more difficult to assess the effects measurements have on large states. Thus, it will be useful to split a large Clifford operation into smaller gates, and argue about the correctness of the implementations of the individual gates instead of the operation as a whole. Implicit in such an argument is a way to stitch together measurement-based computation gadgets so that two gates can be composed.

Let us first state what we mean by *measurement-based computation gadget* for a Clifford U. Each measurement gadget is defined by an m-qubit initial state $|\psi\rangle$, a graph state $|G\rangle$, and a set of measurements S on $|G\rangle$. The qubits of the graph state are partitioned into three sets: m input qubits, body qubits, and m output qubits. Starting with the state $|\psi\rangle \otimes |G\rangle \otimes |+\rangle^{\otimes m}$, apply a CSIGN gate applied between the ith qubit of $|\psi\rangle$ and the ith input qubit of $|G\rangle$, and between the ith output qubit of $|G\rangle$ and the ith qubit of $|+\rangle^{\otimes m}$. Measure each qubit of $|\psi\rangle$ in the X-basis and each qubit of $|G\rangle$ in the basis according to S. The guarantee of the measurement gadget is that the qubits that were initially in the state $|+\rangle^{\otimes m}$ are now in the state $PU|\psi\rangle$ for some Pauli P depending on the measurement outcomes.

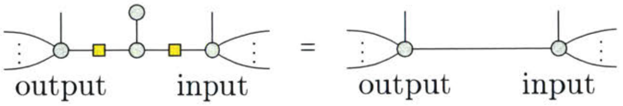
As it turns out, there is a simple procedure to compose measurement gadgets.

Lemma 113 (Raussendorf, Browne, and Briegel [94]). *Let $|\mathcal{G}_1\rangle$ be the graph state which implements the gate $g_1 \in \mathcal{C}_m$ via the set of measurements M_1, and let $|\mathcal{G}_2\rangle$ be the graph state implementing $g_2 \in \mathcal{C}_m$ via the measurements M_2. Let $\mathcal{G}$ be the composition graph: $\mathcal{G}_1$, followed by a layer of n auxiliary vertices, followed by $\mathcal{G}_2$. The ith auxiliary vertex has two edges: one to the ith output qubit of $\mathcal{G}_1$, and one to the ith input qubit of $\mathcal{G}_2$. Let M be the set of X-basis measurements on those auxiliary qubits. Then, the measurements $M_1 \cup M_2 \cup M$ on the state $|\mathcal{G}\rangle$ implement the transformation $g_2 g_1$.*

Sketch. This is easy to see from the ZX-calculus. Consider the qubit in $|\mathcal{G}\rangle$ which is at the interface of the input and output qubits of $\mathcal{G}_1$ and $\mathcal{G}_2$:



The X-measurement teleports the output qubit of the first gadget to the input qubit of the next.



□

6.3 Model

In this chapter, we are exploring the computational complexity of a classical simulation of an interactive protocol with a low-depth quantum computer. Before we discuss interactive protocols, let us explain why we do not expect relation problems to give us, e.g., a RelNC^1-hardness result for a problem in RelQNC^0.

First, although complexity theorists know how to prove specific problems are not in RelNC^0 (via light cones, for instance), AC^0 (e.g., using the switching lemma), or even $\mathsf{AC}^0[2]$ (see Razborov and Smolensky [97, 108]), proving unconditional circuit lower bounds for even slightly larger classes (e.g., TC^0) is beyond our current tools. The kind of hardness we hope to prove (i.e., NC^1-hardness or $\oplus\mathsf{L}$-hardness) should therefore be *conditional* on complexity theoretic assumptions. For example, if we could prove by a reduction that a constant-depth quantum circuit solves an NC^1-complete problem, then we could say $\mathsf{TC}^0 \subsetneq \mathsf{QNC}^0$ *conditional* on the (widely assumed) conjecture that $\mathsf{TC}^0 \subsetneq \mathsf{NC}^1$.

However, we think it is unlikely that oracle access to RelQNC^0 circuits can help solve NC^1-hard problems. In fact, we make the following precise conjecture.

Conjecture 114. *Let* $\mathsf{QNC}^0_f = \mathsf{QNC}^0[\mathsf{fanout}]$ *be the family of* QNC^0 *circuits augmented with a unbounded fan-out gate. Fan-out is the classical reversible gate which XORs a single control bit into any number of target bits (it can be constructed from a deep but straightforward network of CNOTs from the control to each target). Let* BQNC^0_f *be the class of decision problems solved by bounded-error* QNC^0_f *circuits.*

We conjecture that (as decision classes) $\mathsf{NC}^1 \not\subseteq \mathsf{BQNC}^0_f$.

We can think of this as an extension of the conjecture that TC^0 does not contain NC^1, because a result of Hoyer and Špalek [64] shows how to construct majority gates in QNC^0_f, and therefore TC^0 circuits can be implemented directly in QNC^0_f. We make the conjecture on the basis that the Hoyer and Špalek result has not been extended to NC^1 in the intervening 15 years, perhaps because their work depends on executing sequences of *commuting* gates, but we know NC^1 can compute products over non-abelian groups.

The informal consequence of the conjecture is that an oracle for some relation problem in RelQNC^0 cannot help solve NC^1-hard problems.

Corollary 115. *Suppose A is a relation problem in* RelQNC^0. *Under the conjecture,* $(\mathsf{TC}^0)^A$ *does not contain* NC^1.

Proof. Suppose for a contradiction that $(\mathsf{TC}^0)^A$ solves some NC^1-hard problem, and therefore NC^1 is contained in $(\mathsf{TC}^0)^A$. We argue that the $(\mathsf{TC}^0)^A$ circuit can be translated to a QNC^0_f circuit with bounded error, since majority gates can be constructed (with bounded error), fan-out gates are assumed, and QNC^0 gates are part of QNC^0_f by definition. It follows that

$$\mathsf{NC}^1 \subseteq (\mathsf{TC}^0)^A \subseteq \mathsf{BQNC}^0_f,$$

contradicting the conjecture. □

The key point is that this conjecture and the corollary above rule out a *black box* hardness reduction. We want a task in $A \in \mathsf{RelQNC}^0$ such that a classical implementation of A solves NC^1-hard problems, but the conjecture implies a quantum implementation of A does not solve NC^1-hard problems. The only way to have both results is to open up the oracle (i.e., black box) and have the proof depend on whether the internals are classical. For example, the sampling problems separating quantum and classical machines (e.g., Aaronson-Arkhipov [2]) use the fact that the classical machine may be assumed WLOG to be deterministic, with the random bits fed into it as a string. Taking out the randomness allows us to estimate probability amplitudes via approximate counting, where no similar idea is possible with a quantum machine.

However, we were unable to find an idea to separate classical and quantum devices for *relation* problems. Instead, we switch to *interactive* problems. Notice that this does not overcome the black box argument above; we do not expect an oracle solving the interactive task to assist in solving NC^1-hard problems. What interactivity gives is an easy way to extract more power (and thus solve hard problems) from a classical simulation of the task than from an actual quantum device performing the same task.

Let us imagine two parties: a prover (who is supposed to give answers that are consistent with a low-depth quantum computer), and a challenger. Suppose that the low-depth quantum computer starts the protocol with the graph state $|G\rangle$ where G is a subgraph of a 2D grid. Let $|\psi_0\rangle = |G\rangle$. The ith round of the protocol consists of the following:

- The challenger chooses a set of non-overlapping, local, Clifford gates on $|\psi_{i-1}\rangle$.
- The prover returns an outcome consistent with first applying the set of gates and then measuring those qubits in the Z-basis. Let $|\psi_i\rangle$ be the state on the unmeasured qubits consistent with the measurement results.

It is worth stressing here that the measurement results of the prover do not need to be uniformly chosen from the possible outcomes. There simply must be some positive probability that a quantum computer faithfully executing the protocol returns those answers.

We now ask what the computational complexity is for passing such a protocol. As a first observation, notice that if the protocol has polynomially many rounds, then the challenger can force the prover to simulate measurement-based computation on cluster states, and thus simulate an arbitrary Clifford circuit. The prover would then necessarily need to have the power of $\oplus\mathsf{L}$. At the other extreme, if we only ask for a single-round of measurements, then this problem reduces to the Hidden Linear Function (HLF) problem of Bravyi, Gosset, and König. As discussed above, since a constant-depth quantum circuit solves the HLF problem, we conjecture that an oracle solving the problem does not help solve NC^1-hard problems.

Naturally we want a protocol with constantly many rounds so that we may still claim the circuit is "constant depth", otherwise we would just implement measurement-based quantum computation in full. We focus on interactive protocols that have two rounds, the minimum non-trivial number of rounds. Our goal will be to show that any classical model of computation solving the two-round cluster statement measurement problem can also solve problems that are thought to be computationally harder (e.g., problems in $\oplus\mathsf{L}$).

Consider the CCC^0 circuit for the two round protocol in Figure 6-2. After the first round, the remaining unmeasured qubits of the graph state are in some state $|\psi_1\rangle$. However, if we were to rerun the protocol on the quantum computer, the remaining qubits are in some state $P|\psi_1\rangle$ where P is some Pauli operator depending on the measurement results. This corrupts the state so that measurements outputs from multiple invocations of the protocol are not easily compared.

There are a few ways one might try to circumvent this issue. First, one might hope to process the measurement results from the first round to recover P and therefore recover $|G_1\rangle$. Unfortunately, such a calculation is prohibitively computationally expensive, being as hard as our original problem (i.e., NC^1 hard or $\oplus\mathsf{L}$ hard). Alternatively, one might hope to duplicate the state after the first round to measure again. Of course, duplicating a state violates the no-cloning theorem, and running multiple copies of the procedure will produce different errors and thus different states.

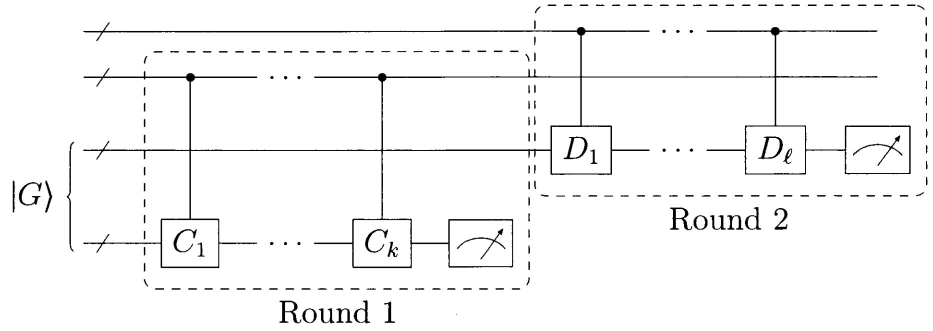


Figure 6-2: CCC^0 circuit for two round protocol for gates $C_1, \ldots, C_k$ in the first round, and gates $D_1, \ldots, D_\ell$ in the second round.

We model the interactive protocol with a quantum computer by an oracle $\mathcal{O}$. The oracle takes the set of gates for the first round and outputs a consistent measurement result, and then the oracle takes a set of gates for the second round and outputs measurement results consistent with both the first round measurements and the given set of gates. Once the oracle returns an answer for the second phase, you cannot ask it a different question for the second round.

Now consider a classical circuit which passes the two-round measurement protocol. Unlike the quantum model, the no-cloning theorem does not prevent us from copying the state of the classical machine after the first round. That is, after the second round of the machine, we can *rewind* the classical machine back to the beginning of the second round. We model this interaction with an oracle $\mathcal{O}_{\text{rev}}$. There are two ways to query the oracle:

1. Given a set of gates for the first round, the oracle returns consistent measurement results for the first round, or
2. Given a set of gates for the second round and a set of gates/measurement results in the first round, the oracle returns a consistent measurement result for the second round. *Such oracle queries are only guaranteed to be correct when the oracle has previously returned those first round measurements on that particular set of first round gates.*

Under almost any kind of classical circuit or classical Turing machine, the rewind oracle can be implemented by the same computational device as the original oracle.

Proposition 116. *Suppose $\mathcal{O}$ is an oracle for a 2-round interactive task. If $\mathcal{O} \in \mathsf{Inter}\mathcal{C}$ where $\mathcal{C} \in \{\mathsf{AC}^0, \mathsf{AC}^0[p], \mathsf{TC}^0, \mathsf{NC}^1, \mathsf{L/poly}, \oplus\mathsf{L/poly}\}$ is one of several types of circuit or Turing machine, then the rewind oracle, $\mathcal{O}_{\text{rev}}$, is in $\mathsf{Inter}\mathcal{C}$.*

Problem 117 (2-Round Clifford Simulation (CliffSim[2])). *Let* 2-Round Clifford Simulation *be the task of passing the interactive protocol above in the special case of 2 rounds. Later, we will further specialize this problem to be more specific about*

- *the geometry of the starting graph state,*
- *the precise encoding of the challenges, and*
- *the acceptable rate of error (if any).*

See Problem 121 and Problem 137 below for the specializations.

6.3.1 Upper bounds

Let us now address the question of the complexity of Clifford sampling problems. A theorem of Jozsa and Van den Nest shows that sampling Clifford circuits and calculating marginal probabilities are in polynomial time [67], but we would like to show that measurement statistics of Clifford circuits can be calculated in ⊕L. Also recall that the ⊕L procedure of Aaronson and Gottesman for simulating Clifford circuits only suffices for measurement on a single qubit [4]. Since our protocol involves sampling from many qubits, we must extend those arguments.

Problem 118 (Multi-qubit Clifford Measurement). *Given n generators for an n-qubit stabilizer state $|\psi\rangle$ and an integer $m \leq n$, output a measurement of the first m qubits of $|\psi\rangle$ in the Z-basis and a corresponding set of generators for the measured state.*

Problem 119 (Postselected Multi-qubit Clifford Measurement). *Given n generators for an n-qubit stabilizer state $|\psi\rangle$ and $p \in \{0,1\}^m$, output n generators for the projection of $|p\rangle\langle p| \otimes I^{m-n}$ onto $|\psi\rangle$, or report that projection is empty.*

Let us define some useful notation for the following theorem. Recall that we can write any Pauli string $g \in \mathcal{P}_n$ as a $(2n+1)$-length binary vector: the Pauli-X components, the Pauli-Z components, and the sign bit. For this reason, let us define the characteristic function for the X Paulis of g as $g^{(x)}$ where $g_j^{(x)} = 1$ if the jth Pauli is X or Y, and 0 otherwise. Similarly, define the characteristic function for the Z Paulis of g as $g^{(z)}$ where $g_j^{(z)} = 1$ if the jth Pauli is Z or Y. Let $Z_j \in \mathcal{P}_n$ be the Pauli operator which applies Pauli-Z to the jth qubit.

Theorem 120. *The Multi-qubit Clifford Measurement Problem and the Postselected Multi-qubit Clifford Measurement Problem are in relational ⊕L.*

Proof. We will use the fact that a problem is in ⊕L if it is computable by an L machine with access to a ⊕L oracle [63]. Let us assume that our initial state has generators $g_1, g_2, \ldots, g_n \in \mathcal{P}_n$. To keep track of the characteristic Pauli-X and Pauli-Z vectors of the generators, we define the matrix $G_{s,t}^{(x)} = \{g_{i,j}^{(x)}\}$ for $i \in [s]$ and $j \in [t]$, and $G_{s,t}^{(z)}$ similarly. We divide any multi-qubit measurement into two steps: measurement on a subset of the qubits where the outcomes will be random, and then measurement on the remaining qubits where the outcome is deterministic.

Step 1: Measurements which return a random outcome
When measuring multiple qubits at a time, we first identify a maximal set of qubits that yield a random outcome when measured.[8] To do this, let us first recall the procedure for measuring the a single qubit [4, 53]. If we measure qubit j and there exists a generator g_i such that $g_{i,j}^{(x)} = 1$, then the measurement outcome is random. We can construct a new set of generators as follows:

- For all $k \neq i$ such that $g_{k,j}^{(x)} = 1$, replace g_k with $g_k g_i$,
- Replace g_i with $\sigma_j Z_j$ where σ_j is a random sign in $\{-1, +1\}$.

Thus, the positions of the nondeterministic measurements are a function of the matrix $G_{n,n}^{(x)}$. In fact, the number of random measurements is equal to the rank of $G_{n,m}^{(x)}$ since the single-qubit measurement procedure described above row reduces on this submatrix. It may seem that simulating these row reductions is an inherently sequential process, and therefore not in $\oplus\mathsf{L}$. However, this is not true. We can break the row reduction into two $\oplus\mathsf{L}$ steps: choosing a basis, and selecting new generators.

First, we select a basis $B = \{b_1, \ldots, b_r\} \subseteq G$ such that $|B| = \operatorname{rank}(G_{n,m}^{(x)})$ and the row space of $B_{r,m}^{(x)}$ is equal to the row space of $G_{n,m}^{(x)}$, where the matrix $B_{s,t}^{(x)}$ is once again defined as $\{b_{i,j}^{(x)}\}$ for $i \in [s]$ and $j \in [t]$. We select this basis in the following way:

$$g_i \in B \Leftrightarrow \operatorname{rank}(G_{i,m}^{(x)}) > \operatorname{rank}(G_{i-1,m}^{(x)})$$

Notice that we can construct this set with L machine with access to an oracle for rank (which is in $\oplus\mathsf{L}$).

The next step is to compute a new set of generators such that their characteristic Pauli-X vectors are in row echelon form. First, define vectors $f_j = (0, \ldots, 0, 1)$ of length j. Notice that there are exactly r different f_j such that f_j is in the row space of $B_{r,j}^{(x)}$. This comes directly from the row echelon form of $B_{r,m}^{(x)}$. Once again, we can find these r vectors in $\oplus\mathsf{L}$ by comparing the rank of $B_{r,j}^{(x)}$ to the rank of f_j appended to $B_{r,j}^{(x)}$. Let $F = \{f_j : f_j \in \operatorname{rowsp}(B_{r,j}^{(x)})\}$ be this set.

For each $f_j \in F$, solve the linear equation $xB_{r,j}^{(x)} = f_j$ where $x \in \{0,1\}^{1\times r}$ is unknown, and add $\prod_{k=1}^{r} x_k b_k$ to the new list of generators. For each generator $g_i \in G \backslash B$ not in the basis, solve the linear equation $xB_{r,m}^{(x)} = (g_{i,1}^{(x)}, \ldots, g_{i,m}^{(x)})$, and add $g_i \prod_{k=1}^{r} x_k b_k$ to the new list of generators. Let us return to the issue of computing these products in $\oplus\mathsf{L}$ later.

The final stage of the random measurements procedure depends on whether or not there is postselection. For the Multi-qubit Clifford Measurement Problem, we replace each generator corresponding to a vector $f_j \in F$ by a new generator $\sigma_j Z_j$ where σ_j is a random sign in $\{-1, +1\}$. For the Postselected Multi-qubit Clifford Measurement Problem, we perform the same replacement, but set $\sigma_j = (-1)^{p_j}$ where p_j was the intended value of the postselected bit.

[8] It is worth noting here that this choice is not unique. Consider the state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Measuring either the first or second qubit results in a random outcome, but measuring one qubit after the other results in the same measurement outcome.

Step 2: Measurements with deterministic outcome
Now that the random measurement outcomes are fixed, we must compute the outcomes of the deterministic measurements. Since these measurements do not affect the state, we already know what the final generators for the state are (except in the postselected case, where the projection may be empty). Let G be the set of generators for the state after measurement in the first step. Supposing the outcome on qubit j is deterministic, it must be that some product of the generators in G yields Z_j or $-Z_j$. We only need to report which is true for each $j \in [m]$.

Let e_j to be the all-zeros vector of length n which has a 1 at position j. Solve the equations $xG^{(z)} = e_j$ and $xG^{(x)} = (0, \dots, 0)$ for $x \in \{0,1\}^{1\times n}$, and let $\sigma_j = \operatorname{sgn}(\prod_{k=1}^{n} x_k g_k)$. For the Multi-qubit Clifford Measurement Problem, output σ_j. For the Postselected Multi-qubit Clifford Measurement Problem, report that the projection is empty if $\sigma_j \neq (-1)^{p_j}$.

All that remains to complete the theorem is to show that we can compute products of generators in $\oplus\mathsf{L}$. The first thing to note is that we cannot keep an entire generator $g \in \mathcal{P}_n$ in memory. Each time we need the jth Pauli of g, we recompute it from scratch. Notice that calculating the product of polynomially many elements of $\mathcal{P}_1$ can performed in L. Since we are computing each Pauli locally, we must also keep track of a local sign bit in $\{\pm 1, \pm i\}$. To compute the sign of a generator (which will necessarily be either +1 or −1), we compute the product of each of the local sign bits, which once again is an operation in L. This completes the proof. □

6.4 NC^1-hardness

Recall the 2-Round Clifford Simulation problem, and specialize it as follows.

Problem 121 (Narrow Cluster Clifford Simulation (narrow-CliffSim[2])). *Let $A \in \{0,1\}^{2\times(8n+1)}$ and $B \in \{0,1\}^{2\times 9}$ be binary matrices. Let* Narrow Cluster Clifford Simulation *be the problem of passing the* CliffSim[2] *protocol with initial state $|H_{4n+4}\rangle$ and the following two rounds of challenges:*

- *Round 1 Challenges: prover measures qubit (i,j) in X-basis if $A_{i,j} = 0$; otherwise, prover measures qubit (i,j) in Y-basis.*
- *Round 2 Challenges: prover measures qubit $(i, j + 8n + 1)$ in X-basis if $B_{i,j} = 0$; otherwise, prover measures qubit $(i, j + 8n + 1)$ in Y-basis.*

Our main result for this section will be as follows.

Theorem 133. *Let $\mathcal{O}_{\mathrm{rev}}$ be the rewind oracle for the 2-round Clifford simulation described above (Problem 121). Then*

$$\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathcal{O}_{\mathrm{rev}}}.$$

As a consequence, if there is, e.g., a TC^0 circuit implementation of $\mathcal{O}$, then by Proposition 116 there is also a TC^0 circuit for $\mathcal{O}_{\mathrm{rev}}$, implying $\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathsf{InterTC}^0} = \mathsf{TC}^0$, contradicting a standard complexity conjecture: $\mathsf{TC}^0 \subsetneq \mathsf{NC}^1$. We take this as evidence that there is no classical implementation of $\mathcal{O}$ with TC^0 circuits.

The high level outline of the proof is as follows. First, we establish that it is NC^1-complete to compute the product of a sequence of 2-qubit Clifford gates, even up to unknown Pauli corrections. Problem 121 is precisely the task of simulating (via MBQC) a sequence of 2-qubit Clifford gates (modulo Paulis) on the state $|{+}{+}\rangle$. In round two, we can use rewinding to measure the state in several bases to perform a kind of tomography. In particular, through the use of non-contextuality, we must learn at least one Pauli string which is not a stabilizer of the state. Finally, by randomizing the reduction we can obfuscate the state and learn a *random* non-stabilizer Pauli string each time, and repetition allows us to learn the state.

6.4.1 Hardness and 2-Qubit Clifford Gates

Recall that the Clifford gates form a discrete group under composition, and the two-qubit Clifford gates are a finite subgroup. Computing the product of a sequence of gates is therefore a special case of the group product problem considered by Barrington and Thérien [13], so we will use their results to prove hardness as soon as we identify $\mathcal{C}_2$, the group of two-qubit Clifford gates.

It turns out that there are exactly 11520 two-qubit Clifford gates (i.e., $|\mathcal{C}_2| = 11520$) in $\mathcal{C}_2 \subseteq \mathrm{SU}(2)$ [4]. Among these are the 16 (again, up to phase) elements of the Pauli group, $\mathcal{P}_2$. As discussed in Section 6.2, we can only implement a sequence of these Clifford operations modulo the Pauli group, so we are actually interested in the group $\mathcal{C}_2/\mathcal{P}_2$ of order 720. Recall from Section 6.2 that the Clifford operations modulo Paulis can be represented by 4×4 symplectic matrices over $\mathbb{F}_2$. It is known [44] that the symmetric group S_{4k+2} is contained in the symplectic group of $4k \times 4k$ matrices over $\mathbb{F}_2$, and by counting we see that it must be the whole group when $k = 1$, so we have the isomorphism $\mathcal{C}_2/\mathcal{P}_2 \cong \mathrm{S}_6$. However, to keep this result as self-contained as possible, we give a proof below, with an explicit description of the isomorphism.

Lemma 122. *The group of two-qubit Clifford gates, up to Pauli corrections, is isomorphic to S_6. That is, $\mathcal{C}_2/\mathcal{P}_2 \cong \mathrm{S}_6$. Figure 6-3 shows how a permutation of six vertices induces a permutation of the edge labels (in $\mathcal{P}_2/\mathcal{Z}_2$) and thus specifies an element of $\mathcal{C}_2/\mathcal{P}_2$.*

Proof. Recall from Section 6.2 that Clifford gates act on the Pauli group by conjugation $U \bullet P = UPU^\dagger$ for any $U \in \mathcal{C}_m$ and $P \in \mathcal{P}_m$. Furthermore, each Clifford operation induces a permutation of $\mathcal{P}_m/\mathcal{Z}_m$, namely $\tilde{\phi}_U : \mathcal{P}_m/\mathcal{Z}_m \to \mathcal{P}_m/\mathcal{Z}_m$, where

$$\tilde{\phi}_U(P\mathcal{Z}_m) = (U \bullet P)\mathcal{Z}_m.$$

Moreover, recall that the kernel of this homomorphism is $\mathcal{P}_m$, so there is an injective homomorphism from $\mathcal{C}_m/\mathcal{P}_m$ into the symmetric group on $\mathcal{P}_m/\mathcal{Z}_m$. Unfortunately, this symmetric group is much too big (even for the special case $m = 2$) since $|\mathcal{P}_2/\mathcal{Z}_2| = 16$.

Clearly not all permutations of 16 elements are in the image of $\tilde{\phi}$. For instance, every unitary commutes with $\mathrm{I}^{\otimes m}$ so $\mathrm{I}^{\otimes m}\mathcal{Z}_m$ is fixed by all permutations. Another

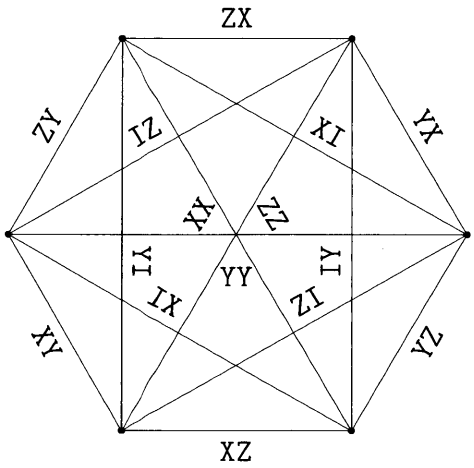


Figure 6-3: $S_6 \cong \mathcal{C}_2/\mathcal{P}_2$ isomorphism

constraint is that two Pauli operations $P, Q \in \mathcal{P}_m$ either commute or anti-commute, i.e., $[P, Q] = \pm \mathtt{I}^{\otimes m}$, and this is preserved by conjugation:

$$[U \bullet P, U \bullet Q] = U \bullet [P, Q] = [P, Q].$$

Also note that the commutator does not depend on the sign of P or Q, i.e., $[P, Q] = [\alpha P, \beta Q]$ for $\alpha, \beta \in \{\pm 1, \pm i\}$. These commutation constraints drastically limit the set of possible permutations of $\mathcal{P}_m/\mathcal{Z}_m$. To understand the permutation we define the following six sets of pairwise anti-commuting Pauli operations.

$$M_1 = \{\mathtt{XI}, \mathtt{YI}, \mathtt{ZX}, \mathtt{ZY}, \mathtt{ZZ}\}, \quad M_2 = \{\mathtt{XI}, \mathtt{ZI}, \mathtt{YX}, \mathtt{YY}, \mathtt{YZ}\}, \quad M_3 = \{\mathtt{YI}, \mathtt{ZI}, \mathtt{XX}, \mathtt{XY}, \mathtt{XZ}\},$$
$$M_4 = \{\mathtt{IX}, \mathtt{IY}, \mathtt{XZ}, \mathtt{YZ}, \mathtt{ZZ}\}, \quad M_5 = \{\mathtt{IX}, \mathtt{IZ}, \mathtt{XY}, \mathtt{YY}, \mathtt{ZY}\}, \quad M_6 = \{\mathtt{IY}, \mathtt{IZ}, \mathtt{XX}, \mathtt{YX}, \mathtt{ZX}\}.$$

In Figure 6-3, each vertex corresponds to an M_i. The edges incident to that vertex are labeled with the Pauli strings of M_i. Not only are the elements in each set pairwise anti-commuting, but they are the *only* pairwise anti-commuting subsets of size five. Since any Clifford operation U preserves commutation relations, $\tilde{\phi}_U$ must permute these six sets/vertices.

In this way, each element of $\mathcal{C}_2/\mathcal{P}_2$ maps to a permutation of $\mathcal{P}_2/\mathcal{Z}_2$, which has an associated permutation in S_6. This map injective because of Lemma 106 and the fact that each Pauli string occurs on exactly one edge (so we can recover the permutation of $\mathcal{P}_2/\mathcal{Z}_2$ from a permutation of vertices). Since $|\mathcal{C}_2/\mathcal{P}_2| = 720 = |S_6|$, the homomorphism is an isomorphism. □

Since $\mathcal{C}_2/\mathcal{P}_2$ is isomorphic to S_6, there is a normal subgroup within it corresponding to A_6. We call the elements of this subgroup *even* since they are isomorphic to *even* permutations within S_6.

It follows immediately that it is hard to compute the product of many two-qubit Clifford gates.

Corollary 123. *Given $U_1, \ldots, U_n \in \mathcal{C}_2/\mathcal{P}_2$, the problem of computing $U_1 \cdots U_n |{+}{+}\rangle$ (modulo Pauli operations) is NC^1-complete. The problem remains hard if $U_1, \ldots, U_n$ are promised to be even and if $U_1 \cdots U_n$ is promised to be either II or $H \otimes H \in \mathcal{C}_2/\mathcal{P}_2$, so $U_1 \cdots U_n |{+}{+}\rangle$ is either $|{+}{+}\rangle$ or $|00\rangle$ (modulo Pauli operations).*

Proof. By definition, the even elements in $\mathcal{C}_2/\mathcal{P}_2$ form a subgroup isomorphic to A_6. Since A_6 is not solvable, a result of Barrington and Thérien [13] says that computing products in the group is NC^1-complete. That is, it is NC^1-complete to compute the product $U_1 \cdots U_n$. In fact, the problem remains hard if the product is promised to be either $\mathtt{I} \otimes \mathtt{I}$ or U, for any choice of U. Let us take $U = H \otimes H \in \mathcal{C}_2/\mathcal{P}_2$. Note that we can express $H \otimes H$ as

$$H \otimes H = (H \otimes \mathtt{I}) \circ \mathrm{SWAP} \circ (H \otimes \mathtt{I}) \circ \mathrm{SWAP},$$

using two SWAP gates and two $H \otimes \mathtt{I}$ gates, so it must be an even gate.

Since the product of the elements is either $\mathtt{I} \otimes \mathtt{I}$ or $H \otimes H$, the state will be either $|{+}{+}\rangle$ or $|00\rangle$, and it follows that distinguishing these states is NC^1-complete. □

6.4.2 Tomography and the Magic Square Game

Recall that in the context of our task, Problem 121, the first round essentially computes the state $U_1 \cdots U_n |{+}{+}\rangle$ with Pauli corrections, in the sense that the state is in the last two unmeasured qubits.[9] The obvious next step is to measure the two-qubit state repeatedly (i.e., rewinding and applying a different Clifford circuit each time) in different bases, and infer the state from the measurements. Under a slightly different definition of Problem 121, this would be a straightforward application of quantum tomography (in our case, Clifford state tomography [82]). Unfortunately, traditional tomography depends on getting random samples, but a classical algorithm solving Problem 121 is allowed to answer challenges with *any* outcome that occurs with nonzero probability. This makes it impossible to infer even one stabilizer of the state.

For example, suppose we are promised that the state is either $|00\rangle$ or $|{+}{+}\rangle$. They have disjoint stabilizers (ZI, IZ, ZZ vs. XI, IX, XX), so learning even one would determine the entire state. However, the states are not orthogonal, so no measurement can perfectly distinguish the states. For any measurement we make, there must be some outcome that could be observed for both $|00\rangle$ and $|{+}{+}\rangle$, and an adversarial classical algorithm may choose to answer with that outcome for every measurement.

Nonetheless, it is possible to learn *something* about the state from measurements. We claim that it is not possible to answer all 2-qubit Clifford measurements in a way which is consistent with all 2-qubit stabilizer states. In fact, we have something stronger from the theory of quantum contextuality: there is a collection of 2-qubit Pauli measurements such that it is impossible to give a consistent answer to all of them that does not depend on *context*, i.e., which other commuting Pauli measurements we

[9] Recall that the Pauli corrections can be computed from the measurement outcomes, but that computation is already NC^1-complete (exercise to the reader), so we will not be able to compute the corrections as part of the NC^1-hardness reduction.

XX	YY	ZZ
YZ	ZX	XY
ZY	XZ	YX

Figure 6-4: The magic square game

apply in the same measurement set. We can use this to force the classical simulator to answer inconsistently on some Pauli measurement, and thus learn that the Pauli string is not a stabilizer.

Before we go further, let us formalize what kind of measurements we perform. For any stabilizer state $|\psi\rangle$ (which are the only states we consider in this chapter), and any $P \in \mathcal{P}_m/\mathcal{Z}_m$, the Pauli measurement associated to P projects onto the $+1$ or -1 eigenspace of the canonical (i.e., positive) Pauli operation in P, and gives a corresponding $+1$ or -1 outcome. As previously discussed, we call P a stabilizer of $|\psi\rangle$ if the outcome of this measurement is deterministic. Given pairwise commuting measurements $P_1, \ldots, P_n \in \mathcal{P}_m/\mathcal{Z}_m$, the order of measurement does not affect the outcomes. Furthermore, the outcome of measuring any product (which commutes with all the individual measurements) will be the product of the outcomes for the individual measurements.

To measure Paulis P and Q we apply a Clifford operation which maps $P \mapsto \texttt{ZI}$, $Q \mapsto \texttt{IZ}$ and then measure both qubits in the Z basis. Although we cannot directly measure Pauli PQ, we can infer its measurement outcome by multiplying the two measurement outcomes from P and Q. We will call any set of three pairwise-commuting Pauli strings $\{P, Q, R\}$ multiplying to $\pm\texttt{II}$ a *Pauli line.* By the above, a two-qubit Clifford measurement gives us outcomes for P, Q, and R on some Pauli line $\{P, Q, R\}$.

The geometry of Pauli lines on two-qubits is important in what follows, so let us list a few facts.

Fact 124. *There are 15 Pauli lines, and they correspond to the perfect matchings in Figure 6-3. I.e., the three Pauli strings in the line correspond to the labels of the three edges in the matching. Symmetrically, there are 15 Pauli strings and each one is contained in 3 Pauli lines. It is possible to partition the Pauli strings across* 5 *non-intersecting Pauli lines, e.g., as*

$$\{XI, IX, XX\}, \{YI, IY, YY\}, \{ZI, IZ, ZZ\}, \{XY, YZ, ZX\}, \{YX, ZY, XZ\}.$$

A particularly nice structure of Pauli lines and Pauli strings is the *magic square game*, given below.

Definition 125. The *magic square game*, independently discovered by Mermin [81] and Peres [89] defines a 3×3 grid (see Figure 6-4) of Pauli measurements on two qubits. Each row and column is a Pauli line. Moreover, the product of each column is $+\texttt{II}$ and the product of each row is $-\texttt{II}$, so the measurement outcomes multiply to 1 for a column or -1 for a row.

The reader may be familiar with a different grid under the name "magic square". There are many choices for the nine Pauli measurements which work (e.g., simply apply any Clifford operation to all Pauli strings, and the new ones are guaranteed to satisfy the same commutation relations) and most references, including Mermin [81] and Peres [89], opt for a different magic square. The reason we choose this one is that there is a particularly nice form for row or column measurements.

Lemma 126. *The first row of the magic square above (Figure 6-4) corresponds to measurement of the qubits in the Bell basis. Measuring any other row or column is done by applying a single qubit Clifford gate to one of the qubits and then measuring in the Bell basis.*

We have already alluded to how we will use the magic square, but let us state it formally:

Theorem 127. *There is a procedure to make six measurements (on six copies) of an unknown two-qubit quantum state $|\psi\rangle$ and learn, with certainty, some Pauli string which does not stabilize $|\psi\rangle$.*

Proof. As one might guess, we measure each copy of the state with a different row or column of the magic square. That is, apply appropriate single qubit gates for the row/column (as in Lemma 126), then measure in the Bell basis.

Each Pauli string is measured exactly twice: once in a row and once in a column. We can construct two tables of outcomes, one from the row measurements and one from the column measurements. In the column table, the product of all elements is +1 (since each column multiplies to 1) and in the row table, the product of all elements is -1. We conclude that the tables must be different, so there exists some Pauli string in the magic square for which we have contradictory measurement outcomes. This Pauli string does not stabilize $|\psi\rangle$, otherwise it would have to measure consistently. □

It is clear that the above procedure (in Theorem 127) can only output Pauli strings that appear in the magic square. Even if we learn whether each Pauli string of the magic square stabilizes $|\psi\rangle$, it may not be enough to learn $|\psi\rangle$ itself. As discussed, there are several magic squares with the properties we need (e.g., conjugate each Pauli by any $U \in \mathcal{C}_2$), so we can ask about a random magic square. However, we run into the same problem as before: the prover's answers may be adversarial, and it is possible to answer in such a way that we cannot determine $|\psi\rangle$. Consider the following fact:

Fact 128. *Every magic square grid intersects every Pauli line, i.e., there exists some Pauli string in both the line and the grid.*

No matter what magic square we measure, the classical algorithm can arrange for `YI`, `IY`, or `YY` to be the only Pauli strings with inconsistent answers, and since all of these are non-stabilizers of both $|00\rangle$ and $|{+}{+}\rangle$ (among other stabilizer states), we cannot deduce $|\psi\rangle$.

6.4.3 Randomization and Self-Reduction

We have argued that by repeated measurement in round two of Problem 121, we can force the prover to reveal a non-stabilizer of the two-qubit state $|\psi\rangle$ being measured. Of course, this is not enough to determine the state.[10] The only way to make progress is to start over in round one with a different instance of the task, constructing a new state $|\psi'\rangle$ which is *related* to $|\psi\rangle$, so that we can carry over what we learn about $|\psi'\rangle$ to $|\psi\rangle$.

Let us first consider a naïve approach on input $U_1, \dots, U_n$. By the standard approach, the prover returns a non-stabilizer P of $U_1 \cdots U_n |{++}\rangle$. To generate a different non-stabilizer Q, we could give the prover the input $U_0, U_1, \dots, U_n$ for random $U_0 \in \mathcal{C}_2/\mathcal{P}_2$. That is, $U_0^{-1} \bullet Q$ is a non-stabilizer $U_1 \cdots U_n |{++}\rangle$. If $U_0^{-1} \bullet Q \neq P$, we have learned new information. Unfortunately, an adversarial prover may anticipate this approach and return $Q = U_0 \bullet P$, so we learn no new information. To force the prover to reveal something new, we need a more thorough randomization procedure, which begins with an idea of Kilian [69].

Theorem 129 (Kilian Randomization). *Let G be a group. Given $g_1, \dots, g_n \in G$, there is a procedure to sample uniformly random $g'_1, \dots, g'_n \in G$ subject to $g_1 \cdots g_n = g'_1 \cdots g'_n$.*

We will need to extend Kilian's idea slightly below.

Corollary 130. *Let G be a group and let $H \trianglelefteq G$ be a normal subgroup. Given $g_1, \dots, g_n \in G$, there is a procedure to sample uniformly random $g'_1, \dots, g'_n \in G$ subject to $g_1 \cdots g_n = g'_1 \cdots g'_n$ and $g_i H = g'_i H$ for all i as follows:*

```
function KILIAN_H($(g_1, \dots, g_n)$)
    $h_1, \dots, h_{n-1} \sim Unif(H)$
    $g'_1 \leftarrow g_1 h_1$
    $g'_i \leftarrow h_{i-1}^{-1} g_i h_i$ for $i = 2, \dots, n-1$
    $g'_n \leftarrow h_{n-1}^{-1} g_n$
    return $(g'_1, \dots, g'_n)$
```

Note that Kilian randomization is the special case $H = G$.

Proof. It is clear that cancellation gives $g_1 \cdots g_n = g'_1 \cdots g'_n$. It is also easy to see that $g'_i H = g_i H$ using the fact that

$$g'_i = h_{i-1}^{-1} g_i h_i = g_i (g_i^{-1} h_{i-1}^{-1} g_i) h_i$$

and $g_i^{-1} h_{i-1}^{-1} g_i \in g_i^{-1} H g_i = H$.

It remains to show that $g'_1, \dots, g'_n$ are uniformly random subject to the constraints. By definition, h_1 is a uniformly random element of H, so g'_1 is a uniformly random element of the coset $g_1 H$. For $i = 2, \dots, n-1$, we see that g'_i is uniformly random conditioned on $g'_1, \dots, g'_{i-1}$ since h_i is uniformly random and independent of $g'_1, \dots, g'_{i-1}$.

[10] Even if we were to make all possible Pauli measurements, an adversarial prover need only reveal three non-stabilizers of the state. By Fact 128, the prover can return non-stabilizers YI, IY, or YY. However, this information still does not distinguish $|00\rangle$ and $|{++}\rangle$.

Finally, given $g'_1, \ldots, g'_{n-1}$, there is a unique choice of g'_n satisfying the constraints, namely $g'_n := (g'_1 \cdots g'_{n-1})^{-1} g_1 \cdots g_n$. □

We can use this technique to randomly self-reduce algorithms that take a list of group elements. After applying this randomization step, the naïve idea from earlier (i.e., multiplying by a random element and conjugating the result by the inverse) actually works. We will state the theorem in the abstract (with a finite group $(G, \cdot)$ acting on a set S as $\bullet : G \times S \to S$ where $(gh) \bullet x = g \bullet (h \bullet x)$), but it is a complicated theorem and it may help to keep an example in mind. For this section, the relevant setting of parameters[11] is $G = H = F = \mathcal{C}_2/\mathcal{P}_2$, acting on the set of Pauli strings $S = \mathcal{P}_2/\mathcal{Z}_2$ by conjugation, $U \bullet P := UPU^\dagger$.

Theorem 131. *Let $A : G^* \to S$ be a randomized algorithm which takes lists of group elements as input and outputs an element of some set S. Suppose G acts on S by $\bullet : G \times S \to S$. Let G have subgroups F, H such that $F \leq H \trianglelefteq G$. Consider the following randomized algorithm:*

```
function $B(g_1, \cdots, g_n)$
    $f \sim Unif(F)$
    $g'_1, \ldots, g'_n \leftarrow \textsc{Kilian}_H(fg_1, g_2, \ldots, g_n)$
    return $f^{-1} \bullet A(g'_1, \ldots, g'_n)$.
```

The output distribution of $B(g_1, \ldots, g_n)$ is $(g_1 \cdots g_n) \bullet \mathcal{D}(Fg_1 \cdots g_n, g_1H, \ldots, g_nH)$ where $\mathcal{D}(Fg_1 \cdots g_n, g_1H, \ldots, g_nH)$ is the average of $(g'_1 \cdots g'_n)^{-1} \bullet A(g'_1, \ldots, g'_n)$ over all $g'_1, \ldots, g'_n$ such that $g'_1 \cdots g'_n \in Fg_1 \cdots g_n$ and $g_iH = g'_iH$ for all i.

Proof. Kilian randomization (in the form of Corollary 130) samples a uniformly random $g'_1, \ldots, g'_n$ subject to $fg_1g_2 \cdots g_n = g'_1 \cdots g'_n$ and $fg_1H = g_1H = g'_1H$, $g_2H = g'_2H$, $\ldots$, $g_nH = g'_nH$. Since f is uniformly random in F, $fg_1 \cdots g_n$ is a uniformly random element of $Fg_1 \cdots g_n$, so we are actually sampling a uniformly random $g'_1, \ldots, g'_n$ such that $g'_1 \cdots g'_n \in Fg_1 \cdots g_n$ and $g'_iH = g_iH$ for all i. Instead of returning the result of $A(g'_1, \ldots, g'_n)$ directly, Algorithm B outputs,

$$\begin{aligned} f^{-1} \bullet A(g'_1, \ldots, g'_n) &= (f^{-1} g'_1 \cdots g'_n) \bullet (g'_1 \cdots g'_n)^{-1} \bullet A(g'_1, \ldots, g'_n) \\ &= (g_1 \cdots g_n) \bullet (g'_1 \cdots g'_n)^{-1} \bullet A(g'_1, \ldots, g'_n) \\ &= (g_1 \cdots g_n) \bullet P \end{aligned}$$

where P is a sample from $(g'_1 \cdots g'_n)^{-1} \bullet A(g'_1, \ldots, g'_n)$ where $g'_1, \ldots, g'_n$ are uniformly random subject to $g_iH = g'_iH$ and $g'_1 \cdots g'_n \in Fg_1 \cdots g_n$. That is, P is drawn from the distribution $\mathcal{D}(Fg_1 \cdots g_n, g_1H, \ldots, g_nH)$. □

Let us interpret the result in the concrete setting we need for this section: $G = H = F = \mathcal{C}_2/\mathcal{P}_2$ and $S = \mathcal{P}_2/\mathcal{Z}_2$ under conjugation, $U \bullet P := UPU^\dagger$. We imagine Algorithm A as adversarially outputting $P \in \mathcal{P}_2/\mathcal{Z}_2$ which does not stabilize $|\psi\rangle := U_1 \cdots U_n |{+}{+}\rangle$.

[11] In Section 6.5 we use Theorem 131 in full generality to prove the ⊕L result.

Recall that P stabilizes $U_1 \cdots U_n |{+}{+}\rangle$ if and only if $(U_1 \cdots U_n)^{-1} \bullet P$ stabilizes $|{+}{+}\rangle$ because

$$\begin{aligned} U_1 \cdots U_n |{+}{+}\rangle &= P U_1 \cdots U_n |{+}{+}\rangle \\ |{+}{+}\rangle &= (U_1 \cdots U_n)^{-1} P U_1 \cdots U_n |{+}{+}\rangle \\ &= ((U_1 \cdots U_n)^{-1} \bullet P) |{+}{+}\rangle . \end{aligned}$$

This fact gives meaning to the expression $(U_1' \cdots U_n')^{-1} \bullet A(U_1', \cdots, U_n')$: it is the non-stabilizer for $|{+}{+}\rangle$ we get by rolling back the unitaries $U_1' \cdots U_n'$ on whichever non-stabilizer A returns for $U_1' \cdots U_n' |{+}{+}\rangle$. It makes sense to average these distributions (as in, e.g., $\mathcal{D}(F g_1 \cdots g_n, g_1 H, \cdots, g_n H)$), since they are all non-stabilizers of the same state. Since $G = H = F$, all the parameters of this distribution have only one value, G. Hence, Theorem 131 defines only one distribution, $\mathcal{D}_n := \mathcal{D}(F g_1 \cdots g_n, g_1 H, \cdots, g_n H)$, which is an average over all lists of n two-qubit Clifford operations. Hence, in our example, Algorithm B is equivalent to sampling P from $\mathcal{D}_n$ and conjugating it by $U_1 \cdots U_n$ to make it a non-stabilizer of $U_1 \cdots U_n |{+}{+}\rangle$.

All that being said, we have not attained our original goal of sampling a *random* non-stabilizer $P \in \mathcal{P}_2 / \mathcal{Z}_2$. If Algorithm A is adversarial, it may be that Algorithm B always returns the same answer, e.g., $(U_1 \cdots U_n) \bullet \mathtt{YY}$, because the distribution $\mathcal{D}_n$ may have support on only one element (e.g., $\mathtt{YY}$). Information theoretically, this kind of distribution actually gives us *more* information about $U_1 \cdots U_n$ than a random non-stabilizer, but since it is more difficult to analyze, we will instead apply another layer of random self-reduction.

Theorem 132. *Let $|\psi\rangle$ be an m-qubit Clifford state. Suppose Algorithm B takes a list of Clifford operations $U_1, \ldots, U_n \in \mathcal{C}_m / \mathcal{P}_m$ and outputs a Pauli string from a distribution $(U_1 \cdots U_n) \bullet \mathcal{D}_n$ where $\mathcal{D}_n$ is a distribution over Pauli strings depending only on n.*

Define Algorithm C as below.

```
function C($U_1, \ldots, U_n$)
    $V \sim \mathrm{Unif}(\{V \in \mathcal{C}_m : V|\psi\rangle = |\psi\rangle\})$
    return $B(U_1, \ldots, U_{n-1}, U_n V)$
```

Algorithm C outputs an element of $\mathcal{P}_m / \mathcal{Z}_m$ at random from $(U_1 \cdots U_n) \bullet \mathcal{D}_n'$ where $\mathcal{D}_n'$ is a distribution such that

$$\Pr[P \textit{ stabilizes } |\psi\rangle | P \sim \mathcal{D}_n'] = \Pr[P \textit{ stabilizes } |\psi\rangle | P \sim \mathcal{D}_n],$$

and $\mathcal{D}_n'$ is uniform over the stabilizers of $|\psi\rangle$ and uniform over non-stabilizers of $|\psi\rangle$.

Proof. By symmetry, it suffices to prove the result for any Clifford state $|\psi\rangle$; let us set $|\psi\rangle = |0^m\rangle$. Any Clifford operation stabilizing $|0^m\rangle$ fixes the Pauli subgroup $\{\mathtt{I}, \mathtt{Z}\}^m$, and therefore the tableau must be of the form

$$\begin{bmatrix} A & B \\ 0 & A^{-T} \end{bmatrix}$$

for arbitrary $A, B \in \{0,1\}^{m \times m}$ subject to the conditions that A is full rank and BA^T is symmetric. Given an arbitrary Pauli string $P = X^x Z^z$ expressed as bits $x, z \in \{0,1\}^m$ for the X and Z components respectively, the tableau above will map it to the Pauli string $VPV^\dagger = X^{x'} Z^{z'}$ where

$$\begin{bmatrix} x' & z' \end{bmatrix} = \begin{bmatrix} x & z \end{bmatrix} \begin{bmatrix} A & B \\ 0 & A^{-T} \end{bmatrix} = \begin{bmatrix} xA & xB + zA^{-T}. \end{bmatrix}$$

Clearly $|0^m\rangle$ is stabilized by strings with only Z component, i.e., where $x = 0$. If P is a stabilizer then it follows that $x = 0$ and thus $x' = xA = 0$. Clearly a random invertible transformation A (or A^{-T}) will map a non-zero $z \neq 0$ to a uniformly random non-zero z', so $VPV^\dagger$ is a uniformly random stabilizer of $|0^m\rangle$. On the other hand, if P is not a stabilizer of $|0^m\rangle$ then $x \neq 0$, and by the same argument as above, $x' = xA$ is a uniformly random non-zero vector. Then zA^{-T} is *not* uniformly random conditioned on x', but we assert that xB is uniformly random. Recall that $S := BA^T$ is a uniformly random symmetric matrix, and by rearranging, $B = SA^{-T}$. Note that if x is non-zero, then xS is a uniformly random vector, and therefore $xB = xSA^{-T}$ is uniformly random because invertible linear transformations preserve the uniform distribution. We conclude that z' is a uniformly random vector, and hence $VPV^\dagger$ is uniformly random non-stabilizer of $|0^m\rangle$. □

We now have all the pieces for this section's main result:

Theorem 133. *Let $\mathcal{O}_{\text{rev}}$ be the rewind oracle for the 2-round Clifford simulation described above (Problem 121). Then*

$$\mathsf{NC}^1/\mathsf{rpoly} \subseteq (\mathsf{AC}^0)^{\mathcal{O}_{\text{rev}}}/\mathsf{rpoly},$$

where rpoly *denotes randomized advice,*[12] *that is, a bit string sampled from a distribution dependent only on the input length.*

Proof. Our goal is to use the rewind oracle to determine the state $U_1 \cdots U_n |{+}{+}\rangle$, modulo Pauli operations, given unitaries $U_1, \ldots, U_n$.

First, we construct Algorithm A from the rewind oracle. Algorithm A applies the oracle to $U_1, \ldots, U_n$ in the first round, then measures magic square rows and columns (all of them, by rewinding) in the second round. From these measurements, Theorem 127 says we can identify a non-stabilizer of the state $|\psi\rangle$ being measured. If there is more than one measurement with inconsistent results, choose one arbitrarily to return. By Theorem 111, the initial state is $|{+}{+}\rangle$ and then $U_1, \ldots, U_n$ are applied (U_n first) to the state by measuring appropriately, so the final two qubits are in state $|\psi\rangle := U_1 \cdots U_n |{+}{+}\rangle$, modulo the Pauli group.

Next, we use Kilian randomization and related ideas in Theorem 131 and Theorem 132 to construct Algorithm C, which makes calls to Algorithm A and returns a uniformly random Pauli string P not stabilizing $|\psi\rangle$. Although it is not stated explicitly, Algorithm C makes exactly one call to Algorithm A (which is in turn making

[12] Really the advice is a distraction from the core theorem, but technically the reduction is randomized so we must include it.

constantly many calls to the rewind oracle), and the reduction can be computed in AC^0 since it only has to sample from the group, multiply constantly many in the group, and translate the two-qubit Clifford elements to measurements.

Finally, we run Algorithm C sufficiently many times (i.e., $O(\log n)$) to collect *all* non-stabilizers of $|\psi\rangle$ with high probability, and thereby deduce $|\psi\rangle$. This solves an NC^1 decision problem in Corollary 123. □

This is evidence for the hardness of computing $\mathcal{O}$, although it takes some work to see. Suppose $\mathcal{O}$ is in $\mathsf{InterTC}^0$ for this problem. That is, there exist TC^0 circuits which implement the interactive task in Problem 121. By Proposition 116, the rewind oracle $\mathcal{O}_{\text{rev}}$ is also in $\mathsf{InterTC}^0$. The AC^0 and TC^0 circuits can all be simulated in TC^0, so the theorem says $\mathsf{NC}^1/\mathsf{rpoly} \subseteq \mathsf{TC}^0/\mathsf{rpoly}$. However, Bennett and Gill [16] show that (among other containments) $\mathsf{TC}^0/\mathsf{rpoly} = \mathsf{TC}^0/\mathsf{poly}$ and $\mathsf{NC}^1/\mathsf{rpoly} = \mathsf{NC}^1/\mathsf{poly}$, so $\mathsf{NC}^1/\mathsf{poly} \subseteq \mathsf{TC}^0/\mathsf{poly}$. We do not believe TC^0 contains NC^1, non-uniformly or otherwise, so we conclude that $\mathcal{O}$ is probably not in $\mathsf{InterTC}^0$.

The remainder of this section is devoted adding error tolerance to the result, which unfortunately complicates the proof somewhat.

6.4.4 Error Tolerance

Suppose the classical simulation is faulty and outputs incorrect (possibly adversarial) responses for some fraction of interactions. That is, for uniformly random inputs (i.e., unitaries $U_1, \ldots, U_n$ chosen randomly from $\mathcal{C}_2$ in the first round, and a uniformly random Clifford measurement in the second round), the classical simulation fails the task with probability $\epsilon > 0$. It turns out that for sufficiently small ϵ, we can still solve an NC^1-hard problem, given access to a rewind oracle. Let us give a simple version of the result.

Theorem 134. *Let $\mathcal{O}_{\text{rev}}$ be the rewind oracle for a 2-round Clifford simulation which performs the task in Problem 121 and fails on a uniformly random input with probability less than ϵ for some $\epsilon < \frac{1}{75}$. Then*

$$\mathsf{NC}^1/\mathsf{rpoly} \subseteq (\mathsf{AC}^0)^{\mathcal{O}_{\text{rev}}}/\mathsf{rpoly}.$$

Proof. Construct Algorithm A from the rewind oracle as before, but make all possible Pauli line measurements in round two. Let δ be the probability that Algorithm A fails, by outputting a stabilizer of $|\psi\rangle := U_1 \cdots U_n |{+}{+}\rangle$ instead of a non-stabilizer, on a uniformly random input. That is, δ is the probability that any of the second round measurements made by Algorithm A fails for a uniformly random first-round input.

Algorithm A queries all 15 possible Pauli lines (recall Fact 124), so if Algorithm A fails, it is because one of the 15 queries failed. Since that query has a $\frac{1}{15}$ chance of being chosen uniformly at random, we conclude that $\frac{\delta}{15} < \epsilon$.

Now construct Algorithm C from Algorithm A using Theorem 131 and Theorem 132, as above. These theorems preserve the error rate; the probability of outputting a stabilizer for a uniformly random input (δ) is the same for Algorithms A, B, and C. Note that if Algorithm C outputs a non-stabilizer then it is still a uniformly random non-stabilizer, and if it outputs a stabilizer then it is also uniformly random.

Algorithm C outputs one of three stabilizers with probability δ, and one of twelve non-stabilizers with probability $1 - \delta$. That is, we see a particular Pauli string with probability $\frac{\delta}{3}$ if it is a stabilizer and probability $\frac{1-\delta}{12}$ if it is a non-stabilizer. As long as $\frac{\delta}{3}$ is less than $\frac{1-\delta}{12}$ (or equivalently, $\delta < \frac{1}{5}$), we can discern the difference in frequency over $\mathsf{poly}(n)$ samples with high probability. That is, the three least-frequently reported Pauli strings are, with high probability, the stabilizers.

Finally, we take $\epsilon < \frac{1}{75}$ to guarantee that $\delta < 15\epsilon < \frac{1}{5}$. □

However, with some deliberate effort to correct errors, we can do better.

Lemma 135. *Let $\nu : \mathcal{P}_2/\mathcal{Z}_2 \to \{+1, -1\}$ be an assignment of measurement outcomes to the canonical (i.e., positive) Pauli string in each coset. For any such assignment, at least* 3 *of the* 15 *Pauli line constraints are violated.*

Proof. This may be verified by brute force enumeration of the $2^{15} = 32768$ different assignments. It is tight because, e.g., assigning *all* Pauli strings to +1 only violates three lines: $\{\mathtt{XX}, \mathtt{YY}, \mathtt{ZZ}\}$, $\{\mathtt{XY}, \mathtt{YZ}, \mathtt{ZX}\}$, and $\{\mathtt{XZ}, \mathtt{YX}, \mathtt{ZY}\}$. □

Theorem 136. *Let $\mathcal{O}_{\text{rev}}$ be the rewind oracle for a* 2*-round Clifford simulation which performs the task in Problem 121, and fails on a uniformly random input with probability less than ϵ for some $\epsilon < \frac{2}{75}$. Then*

$$\mathsf{NC}^1 \subseteq (\mathsf{AC}^0/\mathsf{rpoly})^{\mathcal{O}_{\text{rev}}}.$$

Proof. We construct an Algorithm A which uses the rewind oracle to return a *Pauli line*, rather than an individual Pauli. In the first round, the algorithm sets up state $|\psi\rangle := U_1 \cdots U_n |{++}\rangle$ in the last two qubits, then makes all possible Clifford measurements in the second round, using the rewind oracle. Recall that these measurements give three outcomes for each Pauli string, so we may construct $\nu : \mathcal{P}_2/\mathcal{Z}_2 \to \{+1, -1\}$ where $\nu(P)$ is the majority of the three measurement outcomes for P. By Lemma 135, this assignment of outcomes violates at least three Pauli line constraints, e.g., $PQR = +\mathtt{II}$ but $\nu(P)\nu(Q)\nu(R) = -1$. Return one violated Pauli line at random.

Let ℓ be the unique Pauli line such that all Pauli strings stabilize $|\psi\rangle$. If there are no errors, then the outcomes for any $P \in \ell$ are all the same, and the Pauli line constraint for ℓ is satisfied, so Algorithm A will output some other line. Moreover, it would require at least two measurement errors involving a stabilizer $P \in \ell$ to change the value of $\nu(P)$, so Algorithm A will not output ℓ in case of one error.

When there are two or more errors, we assume the adversary may arbitrarily change ν so that the constraint for ℓ is violated, in addition to at least two other lines. Hence, the probability of outputting ℓ is still at most $\frac{1}{3}$.

We randomly self-reduce Algorithm A (by Theorem 131 and Theorem 132) to produce an Algorithm C. The self-reductions map stabilizers to stabilizers and non-stabilizers to non-stabilizers, so Algorithm C outputs the line ℓ for a particular input with the same average probability as Algorithm A.

Finally, we run Algorithm C many times and count how many times each line is reported. There are three classes of Pauli line:

$$\begin{aligned}C_1 &= \{\ell\},\\ C_2 &= \{\text{lines incident to } \ell\}\backslash\{\ell\},\\ C_3 &= \{\text{lines not incident to } \ell\}.\end{aligned}$$

By the random self-reduction, the probability Algorithm C returns a line is uniform within each class, so the distribution of lines it returns is uniform on three subsets of size $|C_1| = 1$, $|C_2| = 6$, and $|C_3| = 8$. With $O(n)$ samples, we recover the distribution to accuracy $O(1/\sqrt{n})$, and thus also recover the three uniform subsets *unless* elements from two or more classes occur with the same frequency (to within $O(1/\sqrt{n})$). However, even if *two* of the classes are indistinguishable, the third will give us ℓ, since either the class is ℓ itself, or ℓ is the unique Pauli line incident to all lines in the class, or ℓ is the unique Pauli line *not* incident to all lines in the class. Thus, the only way $O(n)$ samples fail to recover ℓ is if Algorithm C outputs Pauli lines uniformly. We argue this is not possible when $\epsilon < \frac{2}{75}$.

Let δ be the probability the rewind oracle makes more than 2 errors among the 15 second round measurements, for a uniformly random first round input. Immediately we have $\epsilon > \frac{2}{15}\delta$, so if $\epsilon < \frac{2}{75}$ then $\delta < \frac{1}{5}$. Observe that Algorithm C outputs ℓ with probability at most $\frac{\delta}{3}$. Since $\frac{\delta}{3}$ is a constant less than $\frac{1}{15}$, the distribution is noticeably different from uniform with $O(n)$ samples. □

6.5 ⊕L-hardness

The purpose of this section is to obtain a ⊕L hardness result for the CliffSim[2] problem. First, we will necessarily need to modify the CliffSim[2] protocol from the previous section to make it more computationally challenging. Since an NC^1 circuit can pass the CliffSim[2] protocol on any grid of constant width, we will use larger graph states in this section in order to apply more complicated gates. For now, let us only roughly describe the initial state as some graph state $|\mathcal{G}_{n,m}\rangle$ where $\mathcal{G}_{n,m}$ is some subgraph of an $m \times \Theta(mn)$ grid. The exact details of $\mathcal{G}_{n,m}$ will be given in Section 6.5.4. The formal statement is as follows:

Problem 137 (Wide Cluster Clifford Simulation (wide-CliffSim[2])). *An instance of* Wide Cluster Clifford Simulation *is defined by two binary matrices $A \in \{0,1\}^{m\times c_1 mn}$ and $B \in \{0,1\}^{m\times c_2}$. Let the problem be to pass the* CliffSim[2] *protocol with initial state $|\mathcal{G}_{n,m}\rangle$ and the following two rounds of challenges:*

- *Round 1 Challenges: prover measures qubit (i,j) in X-basis if $A_{i,j} = 0$; otherwise, prover measures qubit (i,j) in Y-basis.*
- *Round 2 Challenges: prover measures qubit $(i,j+c_1mn)$ in X-basis if $B_{i,j} = 0$; otherwise, prover measures qubit $(i,j+c_1mn)$ in Y-basis.*

Theorem 142. *Let $\mathcal{O}_{\text{rev}}$ be the rewind oracle for* wide-CliffSim[2]. *Then*

$$\oplus\mathsf{L} \subseteq \mathsf{BPL}^{\mathcal{O}_{\text{rev}}}.$$

Once again, this implies strong complexity theoretic evidence that certain highly-parallel and low-depth classical models of computation cannot solve wide-CliffSim[2]. For example, if the oracle is in InterL/poly, then there is an L/poly classically controlled Clifford circuit computing it each round of interaction, and similarly for $\mathcal{O}_{\text{rev}}$, since it is a classical class. The oracle is non-uniform, so we can make $\oplus$L non-uniform as well:

$$\oplus\mathsf{L/poly} \subseteq \mathsf{BPL}^{\mathsf{L/poly}}$$

But BPL is contained in L/poly as a consequence of Bennett and Gill [16], so $\oplus$L/poly $\subseteq$ L/poly, which would be unexpected.

The proof of $\oplus$L-hardness in Theorem 142 follows the same general outline as the proof of NC^1-hardness in Theorem 133. We show that outputs from a conditional distribution of a QNC^0 circuit can be combined by a logarithmic-space machine to solve a $\oplus$L-hard problem. The QNC^0 circuit will be nothing more than a straightforward implementation of measurement based-computation of some Clifford computation. However, because we must choose a larger and more computationally challenging group of Clifford operations to implement, we will need a few key new ideas, especially with respect to the randomization procedure.

6.5.1 Formal statement of problem

The starting point for Theorem 142 is the problem of computing the product of matrices $A_1, \ldots, A_n \in \mathrm{GL}(m, 2)$, which is a well-known $\oplus$L-complete problem for $m = \mathsf{poly}(n)$ (see e.g., [39]). The problem is known to remain $\oplus$L-hard when each matrix represents some $\mathrm{CNOT}(i,j)$ gate between bits $i, j \in [m]$ (see e.g., Lemma 15 in Section 2.4). In fact, we will need yet another modification to this problem for the purpose of randomization.

To see this, let us recall the randomization procedure required for the NC^1-hardness result and show why it is insufficient here: given a product of group elements $g_1 \cdots g_n \in \mathcal{C}_2$, we construct the product $(h_0 g_1 h_1)(h_1^{-1} g_2 h_2) \ldots (h_{n-1}^{-1} g_n)$ with random $h_i \in \mathcal{C}_2$. To randomize this product for $\oplus$L-hardness, each h_i term would be an arbitrary element of $\mathrm{GL}(m, 2)$. There are two main issues:

1. Local measurement statistics such as those obtained from the magic square game are insufficient to reconstruct $(h_0 g_1 h_1)(h_1^{-1} g_2 h_2) \ldots (h_{n-1}^{-1} g_n) \left|+\right\rangle^{\otimes m}$, a (potentially) highly entangled state.
2. Inverting a matrix $A \in \mathrm{GL}(m, 2)$ is $\oplus$L-complete, so even implementing the randomization seems to make the reduction too powerful. The simpler procedure of generating a random pair of matrices $A, A^{-1} \in \mathrm{GL}(m, 2)$ is also not know to be in L to the authors' knowledge.

Because of these issues, we choose a different approach—namely, we perform the randomization with group that is normalized by $\mathrm{GL}(m, 2)$. Let us now carefully define the groups we will be concerned with in this section. The largest group we will need is the group $\langle \text{CNOT}, \text{CZ}, R_z \rangle_m$, which are those transformations generated by CNOT, CZ, and R_z on m-qubits. More formally, the group contains those operations obtained by composing finitely many transformations on m qubits from the set

$$\{\text{CNOT}(i,j) : i \neq j \in [m]\} \cup \{\text{CZ}(i,j) : i \neq j \in [m]\} \cup \{R_z(i) : i \in [m]\}$$

where $\text{CNOT}(i,j)$ denotes the CNOT gate from qubit i to qubit j, $\text{CZ}(i,j)$ denotes a controlled-Z gate from qubit i to qubit j, and $R_z(i)$ denotes a R_z gate on qubit i. We will only be concerned with transformations modulo the Pauli operations. For this reason, we define the group $G_m := \langle \text{CNOT}, \text{CZ}, R_z \rangle_m / \mathcal{P}_m$. Notationally, when we refer to an element $g \in G_m$, we are referring to an actual transformation generated by CNOT,CZ, and R_z. However, when discussing equality,[13] we say that $g = h$ for some $h \in G_m$ if the coset $g\mathcal{P}_m$ equals the coset $h\mathcal{P}_m$.

Let H_m be the normal subgroup[14] of G_m consisting of those transformations generated by CZ and R_z only. That is, $H_m := \langle \text{CZ}, R_z \rangle_m / \mathcal{P}_m$. The fact that H_m is a normal subgroup of G_m can be checked by straightforward calculation using the identities in Figure 6-5.

Another useful property of H_m is that it is abelian, so an arbitrary element of H_m can be represented by a layer of R_z gates followed by a layer of CZ gates, where no $\text{CZ}(i,j)$ gate appears more than once. In fact, since $R_z^2 = \mathbb{I}$ modulo Pauli operations, we can assume that there is at most one R_z gate per qubit as well. Thus, for each $h \in H_m$, we associate a symmetric $m \times m$ binary matrix A such that

$$h = \prod_{i<j} \text{CZ}(i,j)^{A_{i,j}} \prod_{i} R_z(i)^{A_{i,i}}$$

where the equality is modulo Pauli operations. We will call this representation its *canonical decomposition.*

Notationally, we will be somewhat sloppy when writing out products of elements in G_m and H_m. We write gh for $g \in G_m$ and $h \in H_m$ to refer to the product of g and h in the group G_m. We will write CNOT_m for the set consisting of single CNOT gates on m qubits—i.e., $\text{CNOT}_m = \{\text{CNOT}(i,j) : i, j \in [m] \text{ and } i \neq j\}$.

Theorem 16. *Given $g_1, \dots, g_n \in \text{CNOT}_m$, it is $\oplus\mathsf{L}$-hard to compute $g_1 g_2 \cdots g_n$ under $\mathsf{DLOGTIME}$ reductions. The problem remains $\oplus\mathsf{L}$-hard if $g_1 \cdots g_n$ is promised to be either a cycle on the first three qubits (C_3) or the identity transformation (*id*).*

[13] These equivalence classes are well defined since G_m normalizes $\mathcal{P}_m$.

[14] Formally, it may be easier to verify that $\mathcal{P}_m \trianglelefteq \langle \text{CZ}, R_z \rangle_m \trianglelefteq \langle \text{CNOT}, \text{CZ}, R_z \rangle_m$. Then, by the third isomorphism theorem, we have that $H_m \trianglelefteq G_m$ and

$$\frac{G_m}{H_m} = \frac{\langle \text{CNOT}, \text{CZ}, R_z \rangle_m / \mathcal{P}_m}{\langle \text{CZ}, R_z \rangle_m / \mathcal{P}_m} \cong \frac{\langle \text{CNOT}, \text{CZ}, R_z \rangle_m}{\langle \text{CZ}, R_z \rangle_m}.$$

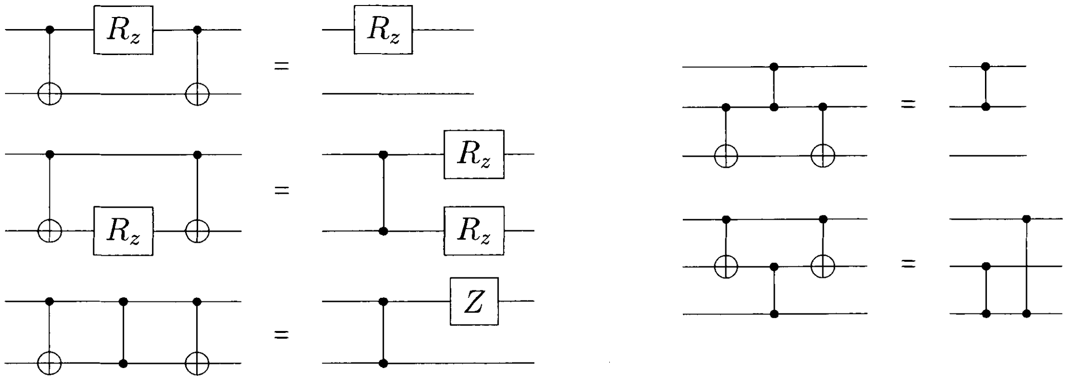


Figure 6-5: Conjugating CZ and R_z by CNOT.

6.5.2 Tomography and the Magic Pentagram Game

Our goal is to distinguish two states based on the product of $g_1, \ldots, g_n$. Since $g_1 \cdots g_n$ is promised to be a permutation, we can commute the product through to such that the comparison will come down to distinguishing $h_1 |+\rangle^{\otimes m}$ and $h_2 |+\rangle^{\otimes m}$ for some $h_1, h_2 \in H_m$. For concreteness, suppose $h_1 = R_z(1)$ and $h_2 = R_z(2)$. Since the resulting states are both product states, it suffices to learn the state of the first two qubits—that is, are the stabilizer generators $\{\mathtt{XI}, \mathtt{IY}\}$ or $\{\mathtt{YI}, \mathtt{IX}\}$?

One might then be tempted to try to repeat the magic square measurements from the previous section to eventually learn this state. Suppose, however, that the magic square measurements always reveal that some element from the Pauli line $\{\mathtt{ZI}, \mathtt{IZ}, \mathtt{ZZ}\}$. Since each element of the Pauli line is a non-stabilizer of both possible states, we learn nothing about our state. On the other hand, we gain nothing from our randomization procedure since the Pauli line is fixed under conjugation by CZ and R_z gates. Finally, every magic square game must contain at least one of ZI, IZ, or ZZ, ensuring that every magic square may return a result which is useless for distinguishing our two states.

This argument generalizes, and for this reason, we switch to a different set of contextual measurements shown in Figure 6-6 known as the magic pentagram [81]. Each vertex in the magic pentagram is labeled by a Pauli operator in the set

$$\bigstar := \{\mathtt{YII}, \mathtt{IYI}, \mathtt{IIY}, \mathtt{XII}, \mathtt{IXI}, \mathtt{IIX}, \mathtt{YYX}, \mathtt{YXY}, \mathtt{XYY}, \mathtt{XXX}\}.$$

Each line consists of 4 commuting operators which multiply to the identity, except for the dashed red line which multiplies to minus identity. One can easily verify that there is no ±1 assignment to the vertices that multiply to 1 along the solid black lines and to −1 along the dashed red line. Thus, we have the following:

Theorem 138. *There is a procedure to make five measurements (on five copies) of an unknown three-qubit quantum state $|\psi\rangle$ and learn, with certainty, some Pauli string which does not stabilize $|\psi\rangle$. Furthermore, the Pauli string is in the set $\bigstar$.*

The proof is virtually identical to that of Theorem 127, so we omit it.

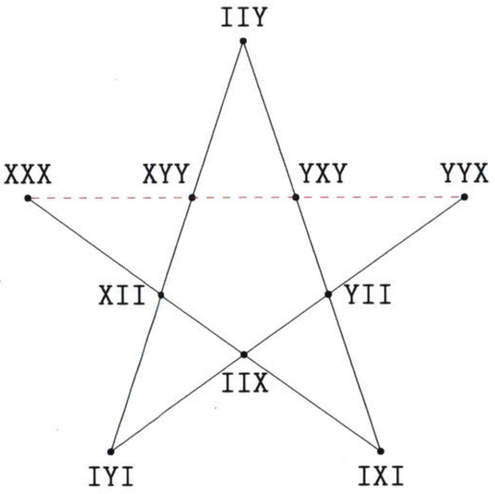


Figure 6-6: Magic pentagram.

6.5.3 Randomization and Self-Reducibility

In this section, we will slightly abuse notation and let $H_3 \leq H_m$ be the subgroup of H_m of CZ and R_z gates on the first three qubits. Let $H_3^{\oplus} < H_3$ be the subgroup of H_3 that only contain those transformations with an even number of R_z and CZ gates. Let $S := H_3^{\oplus} \bullet ✮ = \{h \bullet P : h \in H_3^{\oplus} \text{ and } P \in ✮\}$ represent the set of possible Pauli strings obtained by measuring the first three qubits using magic pentagram measurements (conjugated by some element in $H_3^{\oplus}$).

Suppose we have some randomized algorithm A which takes $g_1, \dots, g_n \in G_m$ as input, and outputs a Pauli string $P \in S$ such that P does not stabilize $|\psi\rangle = g_1 \cdots g_n |+\rangle^{\otimes m}$. Let algorithm B be obtained from Theorem 131 where $F = H_3^{\oplus}$, $H = H_m$, and A is the algorithm defined above. That is, on input $g_1, \dots, g_m \in G_m$, Algorithm B returns a Pauli $(g_1 \cdots g_n) \bullet P$, where $P \in \mathcal{P}_m$ is drawn from a distribution of Pauli strings which do not stabilize $|+\rangle^m$.

We would now like to use B to solve the CNOT multiplication problem (see Theorem 16). Recall that the product of CNOT gates is either the three-cycle or the identity. Since any Pauli P which non-stabilizes $|+\rangle^{\otimes 3}$ also non-stabilizes $C_3 |+\rangle^{\otimes 3}$, the output of B is not very meaningful. On the other hand, in the following theorem, we will show how an L machine can smuggle an element of $H_3^{\oplus}$ into the product of the CNOT gates. Since the two permutations act differently on elements of $H_3^{\oplus}$, the L machine will be able to extract meaningful statistics.

Theorem 139. *Let $g_1, \dots, g_n \in \mathrm{CNOT}_m$ for $m \geq 3$ be such that $g_1 \cdots g_n = \pi$ is some permutation on the first three qubits. Let $f \in H_3^{\oplus}$. Define Algorithm C as below.*

```
function $C(f, g_1, \dots, g_n)$
    return $B(g_1, \dots, g_n, f, g_n, \dots, g_1)$
```

Then the output distribution of $C(f, g_1, \dots, g_n)$ is $(\pi f \pi^{-1}) \bullet \mathcal{D}(g_1 H_m, \dots, g_n H_m)$ where $\mathcal{D}(g_1 H_m, \dots, g_n H_m)$ is the average of $(g_1' \cdots g_{2n}')^{-1} \bullet A(g_1', \dots, g_{2n}')$ over all $g_1', \dots, g_{2n}'$ such that $g_i' H_m = g_i H_m$ and $g_{n+i}' H_m = g_{n-i+1} H_m$ for all $i \in [n]$ and $g_1' \cdots g_{2n}' \in H_3^{\oplus}$.

Proof. By Theorem 131, we have that C will sample from the distribution

$$(g_1 \cdots g_n f g_n \cdots g_1) \bullet \mathcal{D}(H_3^{\oplus} g_1 \cdots g_n g_n \cdots g_1, g_1 H_m, \dots, g_n H_m, g_n H_m, \dots, g_1 H_m)$$

where $\mathcal{D}(H_3^{\oplus} g_1 \cdots g_n g_n \cdots g_1, g_1 H_m, \dots, g_n H_m, g_n H_m, \dots, g_1 H_m)$ is the average of the conjugation $(g_1', \dots g_{2n}')^{-1} \bullet A(g_1', \dots, g_{2n}')$ over all $g_1', \dots, g_{2n}'$ subject to the condition $g_1' \cdots g_{2n}' \in H_3^{\oplus} g_1 \cdots g_n f g_n \cdots g_1$ and $g_i' H_m = g_i H_m$ and $g_{n+i}' H_m = g_{n-i+1} H_m$ for all $i \in \{1, \dots, n\}$. Observe that $g = g^{-1}$ for all $g \in \mathrm{CNOT}_m$, so

$$H_3^{\oplus} g_1 \cdots g_n f g_n \cdots g_1 = H_3^{\oplus} g_1 \cdots g_n f (g_1^{-1} \cdots g_n^{-1})^{-1} = H_3^{\oplus} \pi f \pi^{-1} = H_3^{\oplus}$$

for any $f \in H_3^{\oplus}$. That is, for fixed $g_1, \dots, g_n$, Algorithm C samples from the same distribution $(\pi f \pi^{-1}) \bullet \mathcal{D}$ for some fixed distribution $\mathcal{D}$ *regardless of the choice of* $f \in H_3^{\oplus}$. □

Theorem 140. *Let A be a randomized algorithm which takes $g_1, \dots, g_n \in G_m$ as input, and outputs a Pauli string $P \in S$ such that P does not stabilize $g_1 \cdots g_n |+\rangle^{\otimes m}$. Then, given $g_1, \dots, g_n$, there exists a randomized algorithm in BPL^A which computes $g_1 \cdots g_n$, promised that it is either the three-cycle on the first three qubits or the identity transformation.*

Proof. Fix $g_1, \dots, g_n$ such that $g_1 \cdots g_n = \pi$, and let Algorithm C be obtained by composing Theorem 131 and Theorem 139. By construction, C outputs a non-stabilizer Pauli $(\pi f \pi^{-1}) \bullet P \in S$ of $\pi f \pi^{-1} |+\rangle^{\otimes m}$ where $P \sim \mathcal{D} := \mathcal{D}(g_1 H_m, \dots, g_n H_m)$. In other words, $\mathcal{D}(P)$ is a probability distribution over non-stabilizers of $|+\rangle^{\otimes m}$, and $\mathcal{D}_{\pi,f} := (\pi f \pi^{-1}) \bullet \mathcal{D}$ is the conjugated probability distribution over non-stabilizers of $\pi f \pi^{-1} |+\rangle^{\otimes m}$. The algorithm works in two steps:

1. Query algorithm $C(\mathtt{III}, g_1, \dots, g_n)$ to draw $O(\log n)$-many samples from $\mathcal{D}$.
2. Choose $f \in H_3^{\oplus}$ based on query results, and query $C(f, g_1, \dots, g_n)$ to draw $O(\log n)$-many samples from $\mathcal{D}_{\pi,f}$. This will reveal π with overwhelming probability.

Let us now show how we should choose f in the second step. In particular, we will show that there exists $f \in H_3^{\oplus}$ such that the distribution $\mathcal{D}_{\mathrm{id},f}$ is not equal to $\mathcal{D}_{C_3,f}$. We will argue by contradiction: if there is no such f, then the support of $\mathcal{D}$ is empty. Our starting point will be the fact that $\mathcal{D}$ does not have any support on the stabilizer group of $|+\rangle^{\otimes 3}$, which generated by $\mathtt{XII}$, $\mathtt{IXI}$, and $\mathtt{IIX}$. That is, we know a priori that $\mathcal{D}(\mathtt{XII}) = \mathcal{D}(\mathtt{IXI}) = \mathcal{D}(\mathtt{IIX}) = 0$.

Suppose we have some $f \in H_3^{\oplus}$ which does not distinguish the two permutations. To be concrete, let $f = R_z \otimes R_z \otimes \mathrm{I}$. We have $\mathcal{D}_{\mathrm{id},f}(\mathtt{YII}) = \mathcal{D}(\mathtt{XII}) = 0$ and $\mathcal{D}_{C_3,f}(\mathtt{YII}) = \mathcal{D}(\mathtt{YII})$. Since f does not distinguish the two permutations, it must be that $\mathcal{D}_{\mathrm{id},f}(\mathtt{YII}) = \mathcal{D}_{C_3,f}(\mathtt{YII})$, so

$$0 = \mathcal{D}_{\mathrm{id},f}(\mathtt{YII}) = \mathcal{D}_{C_3,f}(\mathtt{YII}) = \mathcal{D}(\mathtt{YII}).$$

That is, $\mathcal{D}$ cannot have any support on Pauli $\mathtt{YII}$. More generally, consider any Pauli $P \in \mathcal{S}$. If there is no $f \in H_3^{\oplus}$ distinguishing $\mathcal{D}_{\mathrm{id},f}$ from $\mathcal{D}_{C_3,f}$, then $\mathcal{D}_{\mathrm{id},f}(P) = \mathcal{D}_{C_3,f}(P)$

$P \in S$	$f \in H_3^{\oplus}$	$C_3 f^{-1} C_3^{-1} f$	$C_3 f C_3^{-1} f \bullet P$
YII	$R_z(1) \cdot R_z(2)$	$R_z(1) \cdot R_z(3)$	XII
XZI	$\mathrm{CZ}(1,3) \cdot \mathrm{CZ}(2,3)$	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(2,3)$	XII
YZI	$\mathrm{CZ}(1,3) \cdot \mathrm{CZ}(2,3) \cdot R_z(1) \cdot R_z(2)$	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(2,3) \cdot R_z(1) \cdot R_z(3)$	XII
XZZ	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(2,3)$	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(1,3)$	XII
YZZ	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(2,3) \cdot R_z(1) \cdot R_z(2)$	$\mathrm{CZ}(1,2) \cdot \mathrm{CZ}(1,3) \cdot R_z(1) \cdot R_z(3)$	XII
YYX	$R_z(2) \cdot R_z(3)$	$R_z(1) \cdot R_z(2)$	XXX

Table 6.3: Operations $f \in H_3^{\oplus}$ which distinguish $\mathcal{D}_{\mathrm{id},f}$ from $\mathcal{D}_{C_3,f}$. We omit $P \in S$ that are equivalent up to permutation.

and so

$$\mathcal{D}(f^{-1} \bullet P) = \mathcal{D}_{\mathrm{id},f}(P) = \mathcal{D}_{C_3,f}(P) = \mathcal{D}(C_3 f^{-1} C_3^{-1} \bullet P).$$

In other words, $\mathcal{D}(P) = \mathcal{D}(C_3 f C_3^{-1} f \bullet P)$ for all $P \in S$ (recall that $f = f^{-1}$ for all $f \in H_m$). We can now systematically show that $\mathcal{D}(P) = 0$ for every $P \in S$. More precisely, for every $P \in S$, there exists a function $f \in H_3^{\oplus}$ such that $D(C_3 f C_3^{-1} f \bullet P)$ is a stabilizer of $|+\rangle^{\otimes 3}$. We show the complete enumeration in Table 6.3.

To conclude, we simply observe that there must exist (possibly multiple) Pauli $P \in S$ such that $\mathcal{D}(P) > \frac{1}{|S|} > c$. With high probability, we sample such P in the the first step of the protocol. Let $f \in H_3^{\oplus}$ be the associated function from Table 6.3. There is a single Pauli $Q := f \bullet P$ which reveals the permutation π. Suppose $\pi = \mathrm{id}$, then

$$\mathcal{D}_{\pi,f}(Q) = \mathcal{D}_{\mathrm{id},f}(Q) = \mathcal{D}(f \bullet Q) = \mathcal{D}(P).$$

On the other hand, if $\pi = C_3$, then

$$\mathcal{D}_{\pi,f}(Q) = \mathcal{D}_{C_3,f}(Q) = \mathcal{D}(C_3 f C_3 \bullet Q) = \mathcal{D}(C_3 f C_3 f \bullet P) = 0.$$

Thus, in $O(\log n)$ queries to $\mathcal{D}_{\pi,f}$, we will see Q with high probability if $\pi = \mathrm{id}$. Otherwise, we conclude that $\pi = C_3$. □

6.5.4 Initial state details

Our initial graph state $|\mathcal{G}_{n,m}\rangle$ will be arranged for the measurement-based computation of the product of CNOT gates $g_1, \ldots, g_n \in \mathrm{CNOT}_m$. In fact, recall that due to the randomization procedure, we will actually need to compute the product $g_1 h_1 g_2 h_2 \cdots g_n h_n$ where $h_i \in H_m$. For elements of H_m, we can apply the corresponding layer of R_z gates in constant depth by Theorem 110, so we will omit these from the discussion below.

Let us focus now on the application of a single CNOT gate from qubit i to j. In Section 6.2.5, we saw a general procedure to implement any transformation in $\mathcal{C}_2$ on a 2×16 grid. If we wished the apply the same construction in a circuit of m qubits, then we would need to know i and j in advance. That is, we want to construct some fixed graph such that there exists some measurement pattern that implements $\mathrm{CNOT}(i,j)$ for all possible values of i and j. Importantly, we would also like the graph to be embeddable in the grid.

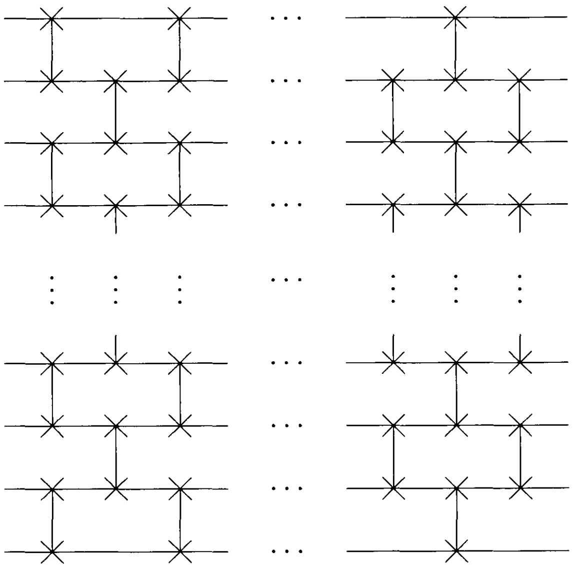


Figure 6-7: Even-Odd sorting network: given any permutation, there is some subset of the SWAP gates above that implements that permutation.

The most straightforward way to ensure that the graph state is grid-like is to only apply local operations. We accomplish this by swapping far-away qubits until they become adjacent. For instance, any CNOT(i,j) gate can be applied by swapping the ith bit until it adjacent to bit j, applying the gate, and then reversing the swaps. There are two downsides to this procedure:

- An element of H_m could consists of $\Omega(m^2)$ many CZ gates. So if we extended this swap architecture to apply an element of H_m, we would needed $\Omega(m^2)$ copies, adding a nontrivial overhead to the entire procedure.
- Which SWAP gates are applied depends on both i and j. This complicates the choice of measurement basis for some qubit in the measurement-based application of some CNOT(i,j) gate. Ideally, this choice of basis is a simple local operation.

We solve both issues with the same construction. The key idea will be to use a sorting network, which is a fixed network of SWAP gates on m qubits that can be toggled to implement every permutation of those qubits. That is, for every permutation, applying some subset of the SWAP gates in the network yields the permutation. Remarkably, there are $O(m)$-depth sorting networks such that every SWAP gate is applied locally.

We will use the even-odd sorting network [58] of depth-m shown in Figure 6-7. That is, on the kth layer, there are SWAP gates between qubits $(2i, 2i+1)$ for k even, and qubits $(2i-1, 2i)$ for k odd. We now claim that the even-odd sorting network implies a $m \times O(m)$ grid implementing any transformation in CNOT$_m$ or H_m. First, we use the sorting network to reverse the order of the list qubits, so qubit 1 becomes qubit m, qubit 2 becomes qubit $m-1$, and so on. It is simple to see that since all operations are local, at some point every pair of qubits must have been swapped. At that point, we can also apply any two-qubit gate using Theorem 111. Importantly, for both CNOT_m and H_m, there is only a single gate that we might want to apply at a given location, so these chosen basis depends only on a single bit of the input (where the specific choice of encoding is given below). Once the qubits are in reverse order, we can simply reverse the order again to complete the transformation.

In conclusion, we let $\mathcal{G}_{n,m}$ be the $m \times \Theta(nm)$ graph obtained by combining the above observations. This yields the following theorem:

Theorem 141. *Let $g_1, \dots, g_n \in \mathrm{CNOT}_m$ and $h_1, \dots, h_n \in H_m$ be given. Each element $h_i \in H_m$ is encoded by a matrix representing its canonical decomposition. Each element $g_i \in \mathrm{CNOT}_m$ is encoded by a binary $m \times m$ matrix whose only non-zero entry (i, j) represents the CNOT gate from qubit i to qubit j. There exist projections $P_1, \dots, P_{\Theta(nm^2)}$ such that*

- *P_i is either the projection $\frac{I+X}{2}$ or $\frac{I+Y}{2}$ onto the ith qubit,*
- *$P_1 \otimes \dots \otimes P_{\Theta(nm^2)} \otimes I^{\otimes m} |\mathcal{G}_{n,m}\rangle \propto |\psi\rangle \otimes g_1 h_1 g_2 h_2 \cdots g_n h_n |+\rangle^{\otimes m}$,*
- *$P_1, \dots, P_m$ are X-projections, and*
- *For all i, P_i depends on at most a single bit of the input.*

6.5.5 Main theorem

We now have all the ingredients to prove the main theorem.

Theorem 142. *Let $\mathcal{O}_{\mathrm{rev}}$ be the rewind oracle for* wide-CliffSim[2]. *Then*

$$\oplus\mathsf{L} \subseteq \mathsf{BPL}^{\mathcal{O}_{\mathrm{rev}}}.$$

Proof. Our goal is to use the rewind oracle to determine the product $g_1 \cdots g_n$, modulo Pauli operations, given unitaries $g_1, \dots, g_n \in \mathrm{CNOT}_m$. By Theorem 16, we will use that it is $\oplus\mathsf{L}$-hard to determine this product promised that it is the three-cycle or the identity transformation.

First, we construct Algorithm A from the rewind oracle. Algorithm A applies the oracle to $g'_1, \dots, g'_n \in G_m$ in the first round, then measures the five lines of the pentagram (all of them, by rewinding) in the second round. The first round measurements are specified by Theorem 141. Recall that the measurements in the second round may require some entangling operations on the first three qubits (not simply X and Y-basis measurments). However, we can simulate these gates by yet more measurement-based computation using Theorem 111. Since the depth of these computations is constant, an L verifier can apply the appropriate recovery operation to the oracle's answer in the second round. From these measurements, we identify a non-stabilizer $P \in \bigstar$ of the state by Theorem 138.

From Algorithm A, we construct Algorithm B from A (Theorem 131), and then Algorithm C from B (Theorem 139). We must now check that an L machine can perform these randomization steps. Algorithm B queries Algorithm A with elements

$h_1 g h_2$ for $h_1, h_2 \in H_m$ and $g \in \mathrm{CNOT}_m$. However, since Theorem 141 requires that each input be specified by a local input encoding, we must show that we can calculate the encoding of the product $h_1 g h_2$ given the individual encodings of h_1, g, and h_2. However, since H_m is abelian and conjugation by CNOT gates is locally computable (see Figure 6-5), preparing the final encoding is in BPL.

Finally, we use Algorithm C to compute $g_1 \cdots g_n$ using Theorem 140, which completes the proof. □

As previously discussed, if there were a L implementation of the oracle, this would imply $\oplus\mathsf{L} \subseteq \mathsf{BPL}$, which is considered unlikely. We conclude that $\mathcal{O}$ is probably not in InterL, and more generally, $\mathcal{O}$ is essentially $\mathsf{Inter} \oplus \mathsf{L}$-complete.

Appendix A

Proof of the multi-switching lemma (Lemma 70)

This appendix gives the proof of Lemma 70, an adaptation of Rossman's lemma [98] to the case of multiple outputs. As discussed in the main text, we need this to get the best parameters for Theorem 71, and thus for Theorem 56 as well.

We follow the notation that was set up by Rossman in [98] and define the following classes of Boolean functions. One difference is that we consider functions with multiple outputs $\{f : \{0,1\}^n \to \{0,1\}^m\}$, instead of $\{f : \{0,1\}^n \to \{0,1\}\}$.

- $\mathrm{DT}(k)$ is the class of depth-k decision trees with a single output bit.
- $\mathrm{CKT}(d; s_1, s_2, \ldots, s_d)$ denotes the class of depth-d AC^0 circuits with s_i nodes at height i for all $i \in \{1, \ldots, d\}$. Note that these circuits compute functions with s_d many output bits.
- $\mathrm{CKT}(d; s_1, s_2, \ldots, s_d) \circ \mathrm{DT}(k)$ is the class of circuits in $\mathrm{CKT}(d; s_1, s_2, \ldots, s_d)$ whose inputs are labeled by decision trees in $\mathrm{DT}(k)$.
- $\mathrm{DT}(t) \circ \mathrm{CKT}(d; s_1, s_2, \ldots, s_d) \circ \mathrm{DT}(k)$ is the class of depth-t decision trees, whose leaves are labeled by elements of $\mathrm{CKT}(d; s_1, \ldots, s_d) \circ \mathrm{DT}(k)$. (Note that these are functions with s_d output bits)
- $\mathrm{DT}(k)^m$ is the class of m-tuples of depth-k decision trees. That is a function $F \in \mathrm{DT}(k)^m$ is a tuple of m functions $F = (f_1, \ldots, f_m)$ where each $f_i \in \mathrm{DT}(k)$.
- $\mathrm{DT}(t) \circ \mathrm{DT}(k)^m$ is the class of depth-t decision trees, whose leaves are labeled by m-tuples of depth-k decision trees, one per output bit.

Recall that $\mathrm{DT}(t) \circ \mathrm{DT}(k)^m$ is the class of functions mapping $\{0,1\}^n$ to $\{0,1\}^m$ that can be evaluated by adaptively querying at most t coordinates globally, after which each of the m output bits can be evaluated by making at most k additional adaptive queries. Note that while the first t queries are global, the last k queries could be different for each output bit.

The next lemma shows that under random restriction with high probability objects of the form $\mathrm{DT}(\cdot) \circ \mathrm{CKT}(d; \ldots) \circ \mathrm{DT}(\cdot)$ reduce to objects of the form $\mathrm{DT}(\cdot) \circ \mathrm{CKT}(d-$

$1;\ldots) \circ \mathrm{DT}(\cdot)$, where the depth of the circuit reduces by one. Applying this lemma for d iterations would reduce the depth of the circuit to 1.

Lemma 143 ([98, Lemma 24]). *Let $d, t, \ell, k, s_1, \ldots, s_d \in \mathbb{N}$, $d \geq 2$, $p \in (0,1)$. If $\ell \geq \log(s_1) + 1$ and $f \in \mathrm{DT}(t-1) \circ \mathrm{CKT}(d; s_1, \ldots, s_d) \circ \mathrm{DT}(k)$, then*

$$\Pr_{\rho \sim \mathbf{R}_p}[f|_\rho \notin \mathrm{DT}(t-1) \circ \mathrm{CKT}(d-1; s_2, \ldots, s_d) \circ \mathrm{DT}(\ell)] \leq s_1(200pk)^{t/2}\,. \tag{A.1}$$

Lemma 144 (Multi-Switching Lemma [62], for this formulation see [112, Theorem 40]). *Let $m, k, q, t \in \mathbb{N}$, $p \in (0,1)$. If $f \in \mathrm{CKT}(1; m) \circ \mathrm{DT}(k)$, then*

$$\Pr_{\rho \sim \mathbf{R}_p}[f|_\rho \notin \mathrm{DT}(t-1) \circ \mathrm{DT}(q-1)^m] \leq m^{1+t/q} \cdot (25pk)^t\,. \tag{A.2}$$

Lemma 145 (Multiple output AC^0 circuits under random restrictions). *Let the following parameters be integral: $d, t, q, k, s_1, \ldots, s_{d-1} \in \mathbb{N}$, $p_1, \ldots, p_d \in (0,1)$. Let $f \in \mathrm{CKT}(d; s_1, \ldots, s_{d-1}, m)$ be a circuit of depth d, n inputs, m outputs, and the given size constraints. Let $s = s_1 + \ldots + s_{d-1} + m$. Let $p = p_1 \cdot p_2 \cdots p_d$. Then*

$$\Pr_{\rho \sim \mathbf{R}_p}[f|_\rho \notin \mathrm{DT}(2t-2) \circ \mathrm{DT}(q-1)^m]$$
$$\leq s_1 \cdot O(p_1)^{t/2} + \left(\sum_{i=2}^{d-1} s_i \cdot O(p_i \log s)^{t/2}\right) + m^{1+t/q} \cdot O(p_d \log s)^t. \tag{A.3}$$

Proof. Let $s_d = m$ for ease of notation. Let $\ell = \lceil \log(s) \rceil + 1$. We think of $\rho \sim \mathbf{R}_p$ as a composition of d random restrictions $\rho_1 \circ \ldots \circ \rho_d$ where each $\rho_i \sim \mathbf{R}_{p_i}$. For $i = 1, \ldots, d-1$ we let $\mathcal{E}_i$ be the event defined by

$$\mathcal{E}_i \iff f|_{\rho_1 \circ \cdots \circ \rho_i} \in \mathrm{DT}(t-1) \circ \mathrm{CKT}(d-i; s_{i+1}, \ldots, s_d) \circ \mathrm{DT}(\ell), \tag{A.4}$$

and denote by $\mathcal{E}_d$ the event that

$$\mathcal{E}_d \iff f|_{\rho_1 \circ \ldots \circ \rho_d} \in \mathrm{DT}(2t-2) \circ \mathrm{DT}(q-1)^m. \tag{A.5}$$

We shall show that $\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_d$ happens with high probability. We start with the first event $\mathcal{E}_1$. Since $f \in \mathrm{CKT}(d; s_1, \ldots; s_d)$, it also holds that

$$f \in \mathrm{DT}(t-1) \circ \mathrm{CKT}(d; s_1, \ldots, s_d) \circ \mathrm{DT}(1). \tag{A.6}$$

Thus, we may apply Lemma 143 with $k = 1$ and get

$$\Pr[\neg\mathcal{E}_1] \leq s_1 \cdot O(p_1)^{t/2}. \tag{A.7}$$

For $i = 2, \ldots, d-1$, using Lemma 143 again, we have

$$\Pr[\neg\mathcal{E}_i | \mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{i-1}] \leq s_i \cdot O(p_i \cdot \ell)^t. \tag{A.8}$$

We are left with the last event – showing that $\Pr[\neg\mathcal{E}_d|\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{d-1}]$ is small. We condition on $\rho_1, \ldots, \rho_{d-1}$ satisfying $\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{d-1}$, and denote by $g = f|_{\rho_1 \circ \ldots \circ \rho_{d-1}}$. Under this conditioning, we have that

$$g \in \mathrm{DT}(t-1) \circ \mathrm{CKT}(1;m) \circ \mathrm{DT}(\ell). \tag{A.9}$$

For each leaf λ of the partial decision tree of depth at most $t-1$ for g, denote by g_λ the function g restricted by the partial assignment made along the path to λ. Note that for each leaf λ, the function g_λ is in $\mathrm{CKT}(1;m) \circ \mathrm{DT}(\ell)$. By Lemma 144, for each leaf λ,

$$\Pr[g_\lambda|_{\rho_d} \notin \mathrm{DT}(t-1) \circ \mathrm{DT}(q-1)^m] \le m^{1+t/q} \cdot O(p_d \cdot \ell)^t. \tag{A.10}$$

Since there are at most 2^{t-1} leaves, by union bound, the probability that there exists a leaf λ for which the switching does not hold is at most

$$2^{t-1} \cdot m^{1+t/q} \cdot O(p_d \cdot \ell)^t\,. \tag{A.11}$$

In the complement event, the function $g|_{\rho_d} = f|_{\rho_1 \circ \ldots \circ \rho_d}$ is in $\mathrm{DT}(t-1) \circ \mathrm{DT}(t-1) \circ \mathrm{DT}(q-1)^m$ and thus $\mathcal{E}_d$ holds. Thus, we showed that

$$\Pr[\neg\mathcal{E}_d|\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{d-1}] \le 2^{t-1} \cdot m^{1+t/q} \cdot O(p_d \cdot \ell)^t \le m^{1+t/q} \cdot O(p_d \cdot \ell)^t. \tag{A.12}$$

Overall, we got that the probability that one of $\mathcal{E}_1, \ldots, \mathcal{E}_d$ does not hold is

$$\Pr[\neg\mathcal{E}_1 \vee \ldots \vee \neg\mathcal{E}_d)] = \sum_{i=1}^{d} \Pr[\neg\mathcal{E}_i|\mathcal{E}_1 \wedge \ldots \mathcal{E}_{i-1}] \tag{A.13}$$

$$\le s_1 \cdot O(p_1)^{t/2} + \left(\sum_{i=2}^{d-1} s_i \cdot O(p_i \cdot \ell)^{t/2}\right) + m^{1+t/q} \cdot O(p_d \cdot \ell)^t\,, \tag{A.14}$$

as promised. □

We are now ready to restate and prove Lemma 70. (Note that the formulation here is slightly stronger than what we used in Section 5.2.4, as we replace $2t$ and q with $2t-2$ and $q-1$, respectively.)

Lemma 146. *Let $f : \{0,1\}^n \to \{0,1\}^m$ be an AC^0 circuit of size s, depth d. Let $p = 1/(m^{1/q} \cdot O(\log s)^{d-1})$. Then*

$$\Pr_{\rho \sim \mathbf{R}_p}[f|_\rho \notin \mathrm{DT}(2t-2) \circ \mathrm{DT}(q-1)^m] \le s \cdot 2^{-t}. \tag{A.15}$$

Proof. We apply Lemma 145 with $p_1 = 1/O(1)$ and $p_2 = \ldots = p_{d-1} = 1/O(\log s)$ and $p_d = 1/O(m^{1/q} \cdot \log s)$. □

Bibliography

[1] S. Aaronson and A. Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1:47–79, 2005. quant-ph/0303041. [p. 42]

[2] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013. Conference version in Proceedings of ACM STOC'2011. ECCC TR10-170, arXiv:1011.3245. [pp. 16, 168]

[3] Scott Aaronson and Adam Bouland. Generation of universal linear optics by any beam splitter. *Phys. Rev. A*, 89(6):062316, 2014. arXiv:1310.6718. [p. 68]

[4] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004. [pp. 26, 27, 90, 157, 171, 172, 174]

[5] Scott Aaronson, Daniel Grier, and Luke Schaeffer. The classification of reversible bit operations. *ITCS*, 2017. arXiv:1504.05155. [pp. 31, 68, 70, 76, 77, 105, 106, 107, 110]

[6] Miklos Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983. [pp. 111, 126]

[7] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, STOC '84, pages 471–474, 1984. [p. 149]

[8] Panos Aliferis. Level reduction and the quantum threshold theorem. *arXiv preprint quant-ph/0703230*, 2007. [p. 69]

[9] Matthew Amy, Andrew N Glaudell, and Neil J Ross. Number-theoretic characterizations of some restricted clifford+ t circuits. *arXiv preprint arXiv:1908.06076*, 2019. [p. 67]

[10] Jonas T. Anderson. On the power of reusable magic states. *arXiv preprint arXiv:1205.0289*, 2012. [p. 69]

[11] Miriam Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014. [p. 164]

[12] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52(3457), 1995. arXiv:quant-ph/9503016. [p. 68]

[13] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, October 1988. [pp. 157, 174, 176]

[14] Adam Bene Watts, Aram W. Harrow, Gurtej Kanwar, and Anand Natarajan. Algorithms, Bounds, and Strategies for Entangled XOR Games. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [p. 147]

[15] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973. [pp. 23, 45]

[16] C. H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981. [pp. 183, 186]

[17] Debajyoti Bera, Frederic Green, and Steven Homer. Small depth quantum circuits. *ACM SIGACT News*, 38(2):35–50, June 2007. [p. 112]

[18] Elwyn Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, volume 2. A. K. Peters, 1st edition, 1982. [pp. 42, 43]

[19] Juan Bermejo-Vega, Dominik Hangleiter, Martin Schwarz, Robert Raussendorf, and Jens Eisert. Architectures for quantum simulation showing a quantum speedup. *Physical Review X*, 8:021010, Apr 2018. [p. 112]

[20] Alex Bocharov, Yuri Gurevich, and Krysta M Svore. Efficient decomposition of single-qubit gates into v basis circuits. *Physical Review A*, 88(1):012313, 2013. [p. 67]

[21] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595, 2018. [p. 16]

[22] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-Generation Topology of D-Wave Quantum Processors. Feb 2019. Accessed: 2019-08-27. [p. 18]

[23] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. Quantum supremacy and the complexity of random circuit sampling. *arXiv preprint arXiv:1803.04402*, 2018. [p. 17]

[24] Adam Bouland, Laura Mancinska, and Xue Zhang. Complexity classification of two-qubit commuting Hamiltonians. *arXiv preprint arXiv:1602.04145*, 2015. [p. 68]

[25] Gilles Brassard, Anne Broadbent, and Alain Tapp. Recasting Mermin's multi-player game into the framework of pseudo-telepathy. *Quantum Information & Computation*, 5(7):538–550, November 2005. [pp. 116, 121, 122, 124]

[26] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018. [pp. 19, 112, 113, 142, 144, 155]

[27] Sergey Bravyi, David Gosset, Robert König, and Marco Tomamichel. Quantum advantage with noisy shallow circuits in 3D. *arXiv e-prints*, page arXiv:1904.01502, Apr 2019. [p. 155]

[28] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005. [pp. 68, 77]

[29] M. Bremner, R. Jozsa, and D. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. Roy. Soc. London*, A467(2126):459–472, 2010. arXiv:1005.1407. [p. 16]

[30] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. [p. 126]

[31] Andrew M. Childs, Debbie Leung, Laura Mančinska, and Maris Ozols. Characterization of universal two-qubit Hamiltonians. *arXiv preprint arXiv:1004.1645*, 2010. [p. 68]

[32] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of Logic, Methodology, and Philosophy of Science II.* North Holland, 1965. [p. 16]

[33] Alan Cobham. The recognition problem for the set of perfect squares. In *7th Annual Symposium on Switching and Automata Theory (swat 1966)*, pages 78–87. IEEE, 1966. [p. 34]

[34] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011. [pp. 161, 162]

[35] John Horton Conway, Robert T. Curtis, Simon P. Norton, and Richard A. Parker. *Atlas of finite groups.* Oxford University Press, 1985. [p. 104]

[36] Matthew Cook. Universality in Elementary Cellular Automata. *Complex Systems*, 15(1):1–40, 2004. [pp. 42, 43]

[37] Stephen A. Cook. Deterministic cfl's are accepted simultaneously in polynomial time and log squared space. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 338–345, New York, NY, USA, 1979. ACM. [p. 111]

[38] Matthew Coudron, Jalex Stark, and Thomas Vidick. Trading locality for time: certifiable randomness from low-depth circuits. *arXiv e-prints*, page arXiv:1810.04233, Oct 2018. [pp. 112, 113, 155]

[39] Carsten Damm. Problems complete for ⊕L. In *International Meeting of Young Computer Scientists*, pages 130–137. Springer, 1990. [pp. 34, 36, 157, 186]

[40] Vasil S Denchev, Sergio Boixo, Sergei V Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3):031015, 2016. [p. 15]

[41] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. London*, A439:553–558, 1992. [p. 16]

[42] David Deutsch. Constructor Theory. *ArXiv e-prints*, October 2012. [p. 44]

[43] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In *International Colloquium on Automata, Languages, and Programming*, pages 285–296. Springer, 2010. [p. 162]

[44] R. H. Dye. Symmetric groups as maximal subgroups of orthogonal and symplectic groups over the field of two elements. *Journal of the London Mathematical Society*, s2-20(2):227–237, 1979. [p. 174]

[45] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. [p. 16]

[46] Maosen Fang, Stephen Fenner, Frederic Green, Steven Homer, and Yong Zhang. Quantum lower bounds for fanout. *Quantum Information & Computation*, 6(1):46–57, January 2006. [p. 112]

[47] Stephen Fenner, Frederic Green, Steven Homer, and Yong Zhang. Bounds on the power of constant-depth quantum circuits. In *Fundamentals of Computation Theory*, pages 44–55, 2005. [p. 112]

[48] R. P. Feynman. Simulating physics with computers. *Int. J. Theoretical Physics*, 21(6-7):467–488, 1982. [p. 16]

[49] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, Dec 1984. [pp. 111, 126]

[50] Martin Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, October 1970. [p. 43]

[51] Brett Giles and Peter Selinger. Exact synthesis of multiqubit clifford+t circuits. *Physical Review A*, 87(3):032332, 2013. [p. 67]

[52] Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, 1997. [p. 67]

[53] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127, 1998. [pp. 31, 172]

[54] Frederic Green, Steven Homer, Cristopher Moore, and Christopher Pollett. Counting, fanout and the complexity of quantum ACC. *Quantum Information & Computation*, 2(1):35–65, December 2002. [p. 112]

[55] Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. Going beyond Bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989. [p. 121]

[56] Daniel Grier and Luke Schaeffer. The Classification of Stabilizer Operations over Qubits. *arXiv e-prints*, page arXiv:1603.03999, Mar 2016. [p. 67]

[57] Walter Grimus and Patrick Otto Ludl. On the characterization of the SU(3)-subgroups of type C and D. *Journal of Physics A: Mathematical and Theoretical*, 47(7):075202, 2014. [p. 68]

[58] Nico Habermann. Parallel neighbor-sort (or the glory of the induction principle). *Technical Report*, 1972. [p. 192]

[59] J. Hardy, O. De Pazzis, and Y. Pomeau. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. *Physical Review A*, 13(5):1949–1961, 1976. [p. 48]

[60] Aram Harrow and Saeed Mehraban. Approximate unitary t-designs by short random quantum circuits using nearest-neighbor and long-range gates. *arXiv preprint arXiv:1809.06957*, 2018. [p. 42]

[61] Johan Håstad. *Computational limitations of small-depth circuits*. PhD thesis, Massachusetts Institute of Technology, 1986. [pp. 111, 115, 117, 126, 127, 148]

[62] Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM Journal on Computing*, 43(5):1699–1708, 2014. [pp. 115, 126, 127, 196]

[63] Ulrich Hertrampf, Steffen Reith, and Heribert Vollmer. A note on closure properties of logspace MOD classes. *Information Processing Letters*, 75(3):91–93, 2000. [p. 171]

[64] Peter Høyer and Robert Špalek. Quantum Fan-out is Powerful. *Theory of Computing*, 1(1):81–103, 2005. [pp. 112, 122, 168]

[65] Dominik Janzing. Is there a physically universal cellular automaton or Hamiltonian? *ArXiv e-prints*, September 2010. [pp. 42, 43, 44]

[66] N. Cody Jones, Rodney Van Meter, Austin G. Fowler, Peter L. McMahon, Jungsang Kim, Thaddeus D. Ladd, and Yoshihisa Yamamoto. Layered architecture for quantum computing. *Physical Review X*, 2(3):031007, 2012. [p. 69]

[67] Richard Jozsa and Maarten Van den Nest. Classical simulation complexity of extended Clifford circuits. *Quantum Information & Computation*, 14(7&8):633–648, 2014. [p. 171]

[68] Julian Kelly. A Preview of Bristlecone, Google's New Quantum Processor. `https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html`. Accessed: 2019-08-27. [pp. 15, 18, 111]

[69] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM. [p. 179]

[70] Alexei Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997. [p. 67]

[71] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits. *Physical review letters*, 110(19):190502, 2013. [p. 67]

[72] Tamara Kohler and Toby Cubitt. Translationally invariant universal classical hamiltonians. *Journal of Statistical Physics*, 04 2019. [pp. 43, 65]

[73] Kevin Krewell. Does IBM Have The Quantum Advantage? Aug 2018. Accessed: 2019-08-27. [pp. 18, 111]

[74] Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp. Reversible space equals deterministic space. *Journal of Computer and System Sciences*, 60(2):354–367, 2000. [p. 34]

[75] Lingling Lao, Daniel M Manzano, Hans van Someren, Imran Ashraf, and Carmen G Almudever. Mapping of quantum circuits onto nisq superconducting processors. *arXiv preprint arXiv:1908.04226*, 2019. [p. 42]

[76] François Le Gall. Average-case quantum advantage with shallow circuits. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. [pp. 112, 113, 155]

[77] Daniel Litinski. Magic state distillation: Not as costly as you think. *arXiv preprint arXiv:1905.06903*, 2019. [p. 68]

[78] Norman Margolus. Physics-like models of computation. *Physica D: Nonlinear Phenomena*, 10(1–2):81–95, 1984. [pp. 42, 43, 48]

[79] Catherine C McGeoch and Cong Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, page 23. ACM, 2013. [p. 15]

[80] N. David Mermin. Extreme quantum entanglement in a superposition of macroscopically distinct states. *Physical Review Letters*, 65:1838–1840, Oct 1990. [pp. 116, 121, 122]

[81] N. David Mermin. Simple unified form for the major no-hidden-variables theorems. *Phys. Rev. Lett.*, 65:3373–3376, Dec 1990. [pp. 158, 177, 178, 188]

[82] Ashley Montanaro. Learning stabilizer states by bell sampling. *Proceedings of the Royal Society A*, 463(2088), 2007. [p. 176]

[83] Cristopher Moore. Quantum Circuits: Fanout, Parity, and Counting. *arXiv:quant-ph/9903046*, March 1999. [p. 112]

[84] Cristopher Moore and Martin Nilsson. Parallel quantum computation and quantum codes. *SIAM Journal on Computing*, 31(3):799–815, March 2002. [pp. 112, 157]

[85] Cristopher Moore and Mats G. Nordahl. Lattice Gas Prediction is P-complete. *Contributions to Mineralogy and Petrology*, page 4001, April 1997. [p. 42]

[86] Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: An easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 890–901, 2018. [p. 112]

[87] Gabriele Nebe, Eric M. Rains, and Neil JA Sloane. The invariants of the clifford groups. *Designs, Codes and Cryptography*, 24(1):99–122, 2001. [p. 67]

[88] Gabriele Nebe, Eric M Rains, and Neil James Alexander Sloane. *Self-dual codes and invariant theory*, volume 17. Springer. [p. 67]

[89] Asher Peres. Incompatible results of quantum measurements. *Physics Letters A*, 151(3):107 – 108, 1990. [pp. 177, 178]

[90] E. L. Post. *The two-valued iterative systems of mathematical logic*. Number 5 in Annals of Mathematics Studies. Princeton University Press, 1941. [p. 76]

[91] Anup Rao. An exposition of Bourgain's 2-source extractor. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 07-34, 2007. [p. 152]

[92] Robert Raussendorf. Quantum cellular automaton for universal quantum computation. *Phys. Rev. A*, 72:022301, Aug 2005. [p. 42]

[93] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22):5188, 2001. [p. 160]

[94] Robert Raussendorf, Daniel E Browne, and Hans J Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(2):022312, 2003. [pp. 68, 77, 157, 160, 161, 167]

[95] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. [p. 118]

[96] Ran Raz and Avishay Tal. Oracle separation of BQP and PH. In *Proceedings of the 51st Annual Symposium on Theory of Computing*, STOC 2019, pages 13–23, 2019. [p. 120]

[97] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, Apr 1987. [pp. 119, 167]

[98] Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of AC^0. Manuscript, 2017. [pp. 115, 126, 127, 128, 195, 196]

[99] Ville Salo and Ilkka Törmä. A one-dimensional physically universal cellular automaton. In *Conference on Computability in Europe*, pages 375–386. Springer, 2017. [p. 62]

[100] Luke Schaeffer. A physically universal cellular automaton. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 237–246, New York, NY, USA, 2015. ACM. [p. 41]

[101] Luke Schaeffer. A physically universal quantum cellular automaton. In Jarkko Kari, editor, *Cellular Automata and Discrete Complex Systems - 21st IFIP WG 1.5 International Workshop, AUTOMATA 2015, Turku, Finland, June 8-10, 2015. Proceedings*, volume 9099 of *Lecture Notes in Computer Science*, pages 46–58. Springer, 2015. [p. 65]

[102] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1/2):1–22, July 2004. [p. 118]

[103] Yaoyun Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. *Quantum Information and Computation*, 3(1):84–92, 2002. quant-ph/0205115. [p. 68]

[104] Seung Woo Shin, Graeme Smith, John A Smolin, and Umesh Vazirani. How "quantum" is the d-wave machine? *arXiv preprint arXiv:1401.7087*, 2014. [p. 15]

[105] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027. [p. 16]

[106] Peter W. Shor. Fault-tolerant quantum computation. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 56–65. IEEE, 1996. [pp. 67, 68, 77]

[107] D. Simon. On the power of quantum computation. In *Proc. IEEE FOCS*, pages 116–123, 1994. [p. 16]

[108] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 77–82, 1987. [pp. 119, 148, 167]

[109] Andrew Steane. Multiple particle interference and quantum error correction. *Proc. Roy. Soc. London*, A452:2551–2577, 1996. quant-ph/9601029. [p. 67]

[110] Yasuhiro Takahashi and Seiichiro Tani. Collapse of the hierarchy of constant-depth exact quantum circuits. *Computational Complexity*, 25(4):849–881, Dec 2016. [p. 112]

[111] Yasuhiro Takahashi and Seiichiro Tani. Power of uninitialized qubits in shallow quantum circuits. In *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:13, 2018. [p. 112]

[112] Avishay Tal. Tight bounds on the fourier spectrum of AC0. In *Computational Complexity Conference*, volume 79 of *LIPIcs*, pages 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. [p. 196]

[113] Barbara M. Terhal and David P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games. *Quantum Information & Computation*, 4(2):134–145, March 2004. [p. 112]

[114] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling.* MIT Press, Cambridge, MA, USA, 1987. [p. 46]

[115] Wim van Dam. A universal quantum cellular automaton. In *Proceedings of PhysComp96*, pages 323–331. InterJournal, 1996. [p. 42]

[116] Umesh Vazirani. *Randomness, Adversaries and Computation.* PhD thesis, UC Berkeley, 1986. [pp. 118, 119, 138]

[117] Emanuele Viola. Extractors for circuit sources. *SIAM Journal on Computing*, 43(2):655–672, 2014. [p. 126]

[118] John von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, Champaign, IL, USA, 1966. [pp. 42, 43]

[119] John Watrous. On one-dimensional quantum cellular automata. In *36th Annual Symposium on Foundations of Computer Science*, pages 528–537. Society Press, 1995. [p. 42]

[120] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 515–526. ACM, 2019. [pp. 111, 155]

[121] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM*, 61(1):2:1–2:32, January 2014. [p. 112]

[122] Andrew C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th Annual Symposium on Foundations of Computer Science*, pages 1–10, 1985. [pp. 111, 126]