Quantum Complexity Theory

Sevag Gharibian

July 2024

Department of Computer Science
Paderborn University
Germany

Contents

1 Classical Complexity Theory Review **1**
- 1.1 Introduction . . . 1
- 1.2 Notation . . . 3
- 1.3 Classical complexity theory . . . 3
  - 1.3.1 Turing machines . . . 3
  - 1.3.2 P and NP . . . 5
  - 1.3.3 The Cook-Levin Theorem, NP-completeness, and Reductions . . . 7
  - 1.3.4 The Extended Church-Turing Thesis . . . 9

2 Quantum computation review **11**
- 2.1 Linear Algebra . . . 11
- 2.2 Basic quantum computation . . . 16
  - 2.2.1 Pure state quantum computation . . . 16
  - 2.2.2 Mixed state quantum computation . . . 21

3 Bounded error quantum polynomial time (BQP) **24**
- 3.1 BPP . . . 24
  - 3.1.1 Syntactic versus semantic classes, PromiseBPP, and what will really be PromiseBQP . . . 25
- 3.2 BQP . . . 27
  - 3.2.1 Norms and perturbations to quantum gates . . . 28
  - 3.2.2 Universal quantum gate sets . . . 30
  - 3.2.3 The BQP subroutine problem . . . 31
- 3.3 Relationship to other complexity classes . . . 32

4 Linear systems of equations and a BQP-complete problem **35**
- 4.1 The basic idea: Quantum eigenvalue surgery . . . 35
  - 4.1.1 Aside: Brief review of the QFT and QPE . . . 38
- 4.2 A quantum algorithm for linear systems of equations . . . 39
  - 4.2.1 Condition numbers of matrices . . . 39
  - 4.2.2 Assumptions for the algorithm . . . 40
  - 4.2.3 The algorithm . . . 41
- 4.3 A BQP-complete problem: Matrix inversion . . . 43

5 Quantum Merlin Arthur (QMA) and strong error reduction **47**
- 5.1 Quantum Merlin Arthur (QMA) . . . 47
- 5.2 Strong error reduction for QMA . . . 50
  - 5.2.1 Intuition: A spinning top . . . 50
  - 5.2.2 Proof of strong error reduction . . . 51
- 5.3 Relationship to other classes . . . 55
  - 5.3.1 The many cousins of QMA . . . 55
  - 5.3.2 Using strong error reduction to show QMA $\subseteq$ PP . . . 57

6 The quantum Cook-Levin theorem **59**
6.1 The Cook-Levin theorem . . . 59
6.2 Local Hamiltonians and Boolean Constraint Satisfaction . . . 62
6.3 The quantum Cook-Levin theorem . . . 65
6.3.1 Containment in QMA . . . 66
6.3.2 Hardness for QMA . . . 68

7 Quantum-Classical Merlin Arthur (QCMA) and Ground State Connectivity **76**
7.1 Quantum-Classical Merlin Arthur (QCMA) . . . 76
7.2 Ground State Connectivity . . . 77
7.2.1 QCMA-completeness of GSCON . . . 79

8 Quantum Merlin Arthur with Unentangled Proofs (QMA(2)) **86**
8.1 QMA with unentangled provers (QMA(k)) . . . 86
8.2 The product state test: QMA(2) and weak error reduction . . . 88
8.2.1 From SWAP test to the Product State test . . . 89
8.2.2 Applications of the Product State test . . . 90
8.3 Verifying NP with logarithmic-size unentangled proofs . . . 90
8.4 Beyond QMA(2) — Quantum Polynomial Hierarchies . . . 90

9 Quantum Interactive Proofs (QIP), semidefinite programs, and multiplicative weights **91**
9.1 The Multiplicative Weights algorithm . . . 91
9.2 QIP and semidefinite programs . . . 93
9.2.1 Quantum interactive proofs . . . 94
9.2.2 Semidefinite programming . . . 95
9.2.3 Quantum interactive proofs as SDPs . . . 98
9.3 QIP = PSPACE . . . 100
9.3.1 The algorithm . . . 101
9.3.2 Correctness . . . 103

10 Boson Sampling **106**
10.1 Of hedgehogs and photons . . . 106
10.2 Connection to the matrix permanent . . . 110
10.3 Boson Sampling . . . 112
10.3.1 The exact case . . . 112
10.3.2 The approximate case . . . 113

11 BQP versus the Polynomial Hierarchy **118**
11.1 The key claim . . . 119
11.1.1 The connection between bounded depth circuits and alternating quantifiers 120
11.1.2 Outline for lecture . . . 121
11.2 The distribution D . . . 122
11.3 Distinguishing D from U_{2N} is easy quantumly . . . 123
11.4 Distinguishing D from U_{2N} is hard classically . . . 125
11.4.1 Boolean circuits and multilinear polynomials . . . 125
11.4.2 Tools and proof approach . . . 125
11.4.3 Main theorem and proof sketch . . . 126

Bibliography **129**

1 Classical Complexity Theory Review

The future is not laid out on a track. It is something that we can decide, and to the extent that we do not violate any known laws of the universe, we can probably make it work the way that we want to.
— Alan Kay

1.1 Introduction

Welcome to Quantum Complexity Theory! In this course, we ask the central question: What *quantum* computational resources are required to solve certain tasks? These tasks might involve "classical" problems, such as factoring large integers or solving systems of linear equations, or "quantum" problems, such as computing properties of physical systems in Nature. Along the way, we shall encounter both "expected" complexity classes, such as quantum generalizations of P and NP, as well as "unexpected" classes with no interesting classical analogue, such as QMA(2), whose exact computational power remains frustratingly unknown. In a nutshell, our aim will be to flesh out an "answer" to Alan Kay's statement above — what does Nature allow us to do, or not do, from a computational perspective?

The first step in this journey is the all-important question:

Why should quantum complexity theory be interesting?

There are many reasons for this; let us focus on arguably the most famous one. Every day, millions of online consumers rely on cryptosystems such as Rivest-Shamir-Adleman (RSA) to keep (say) their credit card information secure online. There is, however, a theoretical problem with this — the RSA scheme has never been *proven* secure. Rather, it is *presumed* to be secure, assuming the mathematical problem of factoring large integers (denoted FACTOR) is "hard". Unfortunately, noone really knows how hard FACTOR really is. For example, it is neither known to be efficiently solvable classically (i.e. it is not known to be in P), nor is it provably intractable (e.g. FACTOR is not known to be NP-hard). In 1994, Peter Shor thread the needle (and in the process, shocked the theoretical computer science community) by demonstrating that a *quantum* computer can solve FACTOR efficiently. Thus, if large-scale quantum computers can be built, then RSA is broken and your credit card information is no longer secure. Moreover, we are faced with the confounding question of how the existence of Shor's algorithm should be interpreted:

- *Is Shor's algorithm evidence that FACTOR is actually in P, and that we as a community have failed to be clever enough to find good classical algorithms for it?*
- *Or perhaps the algorithm hints that quantum computers can solve vastly more difficult problems, including NP-complete ones?*
- *And maybe neither of these holds — maybe quantum computers can outperform classical ones on "intermediate" problems such as FACTOR, but not genuinely "hard" ones such as NP-hard problems?*

To date, we do not know which of these three possibilities holds. What we *do* know, is that we are at a very exciting crossroads in computer science history. Thanks to Shor's algorithm, one of the following statements must be *false*:

1. The Extended Church-Turing Thesis is true.
2. FACTOR cannot be solved efficiently on a classical computer.
3. Large-scale universal quantum computers can be built.

We will return to this crossroads at the end of the lecture, but for now let us note that the fact that quantum computers appear "more powerful" than classical ones is not entirely surprising. Indeed, the physics community has long known that simulating quantum systems with classical computers appears to require an inevitable exponential overhead in time. And as far back as the 1970's, visionaries such as Stephen Wiesner began to realize that quantum information seemingly allows one to do the impossible, such as create physically uncloneable "money" using quantum states. Thus, the question "What *quantum* computational resources are required to solve certain tasks?" indeed appears extremely interesting.

Scope of course. Our focus in this course is quantum complexity theory, strictly construed. The field itself is far too large to capture in a single course; thus, we shall aim to cover many fundamental algorithms and complexity classes, while striking a balance between approachable difficulty for a Masters course and more advanced results. Along the way, useful mathematical frameworks such as Semidefinite Programming (SDPs) and the multiplicative weights method will be covered, which find applications in areas beyond quantum computation.

There are unfortunately many complexity-related topics which, due solely to time constraints, we will likely be unable to cover. A select sample of these include quantum Turing machines, additional complexity classes (such as Quantum Statistical Zero Knowledge (QSZK), Stoquastic MA (StoqMA), Quantum Multi-Prover Interactive Proofs (QMIP), QMA with Polynomial-Size Advice (QMA/qpoly), complexity classes with exponentially small promise gap (such as $\mathrm{QMA}_{\mathrm{exp}}$), Quantum Polynomial-Time Hierarchies (QCPH and QPH), etc...), advanced topics from the subarea of Quantum Hamiltonian Complexity such as area laws, tensor networks, and perturbation theory gadgets for simulating local interaction terms, advanced circuit-to-Hamiltonian constructions such as 1D constructions and space-time constructions, undecidable problems in quantum information such as detecting spectral gaps of local Hamiltonians, other candidate models for demonstrating "quantum supremacy" such as IQP and DQC1, classical simulations of quantum circuits which minimize T-gate count, and so forth.

Resources. The entry point for any course on complexity theory should arguably be the "Complexity Zoo", which provides brief information on over 545 (as of 26.10.2020) complexity classes:

`https://complexityzoo.uwaterloo.ca/Complexity_Zoo`.

Existing surveys focusing on quantum complexity theory include:

- Quantum NP - A Survey (Aharonov and Naveh, 2002): `https://arxiv.org/abs/quant-ph/0210077`,
- Quantum Computational Complexity (Watrous, 2008): `https://arxiv.org/abs/0804.3401`,

- QMA-Complete Problems (Bookatz, 2014): `https://arxiv.org/abs/1212.6312`,
- Quantum Hamiltonian Complexity (Gharibian, Huang, Landau, and Shin, 2015): `https://arxiv.org/abs/1401.3916`.

Prerequisites. Before we can make sense of ideas in quantum complexity theory, we require a brief review of the basics of both classical complexity theory and quantum computing. This course assumes some familiarity with both areas (such as a previous introductory course, particularly for quantum computation), but we shall attempt to make the content as self-contained as possible.

Acknowledgements. We thank Cameron Calcluth, David Fitzek, Oliver Hahn, Stefan Heid, Raphael Heitjohann, Dorian Rudolph, Jannes Stubbemann, Marika Svensson for catching typos in these notes. We thank Harry Buhrman for discussions regarding the contents of Lecture 1.

1.2 Notation

Throughout this course, the symbols $\mathbb{N}, \mathbb{R}, \mathbb{Z}, \mathbb{C}$ refer to the sets of natural, real, integer, and complex numbers, respectively. We define $\mathbb{N}$ to include 0, and $\mathbb{Z}^+, \mathbb{R}^+$ to be the sets of non-negative integers and real numbers, respectively. The set of all n-bit strings is denoted $\{0,1\}^n$. The terminology $|x|$ is overloaded: If $x \in \mathbb{C}$, then $|x| = \sqrt{xx^*}$ (for x^* the complex conjugate of x), and if x is a string, $|x|$ denotes the length (i.e. number of symbols) of x. The notation $[n]$ denotes set $\{1, \ldots, n\}$. We use $:=$ to denote a definition.

1.3 Classical complexity theory

The "basis" of complexity theory is the pair of complexity classes Polynomial-Time (P) and Non-Deterministic Polynomial-Time (NP). To define these, we must recall the definition of a Turing machine. Note that while the remainder of this lecture works exclusively with the Turing machine model of classical computing, one can equivalently work in the *circuit* model, which we will switch to in Lecture 2 when discussing quantum computation. Nevertheless, the notion of a Turing machine will be crucial to defining "uniformly generated quantum circuit families" in Lecture 2, and is good to keep in mind for the remainder of the course. As this section is a review, we shall move rather quickly.

1.3.1 Turing machines

In order to formally study the nature of computing, we require a standard model of what it means to "compute". This role is played by the *Turing machine*, an idealized model for computing proposed by Alan Turing in 1936. In hindsight, the model is remarkably basic — it consists of a "brain" or "control" unit which makes decisions, an infinitely large memory stored on an infinitely long tape, and a "head" to read and write data from the tape. A formal definition is given below.

Definition 1.1 (Turing Machine (TM))**.** *A Turing Machine is given by a* 7*-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$*, defined as:*

- Q *- a finite set denoting the* states *the TM can be in.*

- Σ - *a finite set denoting the input* alphabet*, i.e. set of symbols which can be written on the tape to encode the input before the TM starts. (Σ is assumed not to contain the special blank symbol $\sqcup$.)*
- Γ - *a finite set denoting the tape alphabet, i.e. set of symbols which can appear on the tape. Note $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$.*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ - *the* transition function*, which takes in the current state of the TM and tape symbol under the head, and outputs the next state, tape symbol to write under the head, and whether to move the head left (L) or right (R) one cell on the tape.*
- $q_0 \in Q$ - *the designated start state of the TM.*
- $q_{\text{accept}} \in Q$ - *the designated accept state of the TM, i.e. the TM halts and accepts immediately upon entering state q_{accept}.*
- $q_{\text{reject}} \in Q$ - *the designated reject state of the TM, i.e. the TM halts and rejects immediately upon entering state q_{reject}.*

With this model in place, we can crucially define what is meant by "one step of a computation". Namely, this means our TM reads the current tape symbol under the head, updates the state of the TM, writes a new symbol under the head, and moves the head left or right precisely one cell.

Recall that a TM now operates as follows. Before it begins, we assume the input to the computational problem to be solved is written on the tape using symbols from Σ. Once the TM machine starts, it executes a sequence of computation steps. If the TM enters either state q_{accept} or q_{reject}, the TM halts and accepts or rejects, respectively. Without loss of generality, we may assume that if the TM also produces some output string z upon halting, then only z appears on the output tape once the machine halts. Notably, a TM need not ever enter either q_{accept} or q_{reject}, in which case it runs forever.

Exercise 1.2. Sketch a TM which, given input string $x \in \{0,1\}^n$, outputs 0 if x contains only zeroes, and outputs 1 otherwise. (In other words, the TM computes the OR function on all bits of x.) How many steps does your algorithm asymptotically take (i.e. state your answer in Big-Oh notation)? Does it always halt? A formal definition in terms of Q, δ, etc..., is not necessary; simply sketch the TM's action at a high level.

Languages and decision problems. Throughout this course, we assume strings consist of bits, i.e. $x \in \{0,1\}^n$. Recall that a *language* L is a subset of strings $L \subseteq \{0,1\}^*$, and that computational problems are often modeled as follows: Fix a language L. Given input $x \in \{0,1\}^*$, is $x \in L$? This is known as a *decision* problem, since the answer is either YES or NO.

We say a TM M *decides* a language L if, given any input $x \in \{0,1\}^n$, M halts and accepts if $x \in L$ (rejects if $x \notin L$). Note this definition says nothing about the worst-case number of steps required by M to decide L. Recall that there do exist *undecidable* languages, which encode computational problems which provably cannot be solved by any TM in a finite amount of time. Although undecidable problems do occur in quantum complexity, in this course we will unfortunately not have a chance to pursue such directions.

Exercise 1.3. Recall that the canonical undecidable problem is the Halting Problem (HALT): Given as input a description of a TM M and an input x, does M halt on input x? Can you prove that HALT is undecidable?

Exercise 1.4. Although HALT cannot be solved on a TM, on a real-life computer, HALT *can* be solved - why? What is the crucial difference between a real life computer and a TM which allows this distinction?

Significance. Two important remarks are in order. First, although the TM model nowadays seems "obvious", in that modern computers essentially work in a similar fashion (with the notable exception of solid state drives, which unlike electromechanical drives, do not have a physical "read-write head"), it is precisely due to Turing's inspiration that modern computers developed as they did. Moreover, although the informal concept of "algorithm" has existed for millennia (e.g. Euclid's algorithm for computing the greatest common divisor, dating to around 300 B.C.), the TM model finally gave a standard formal definition of what "algorithm" means. This, in turn, allowed one to show certain computational problems simply cannot be solved by *any* computer in finite time (modulo our next remark below).

The second important remark involves the longstanding *Church-Turing thesis*, which roughly says that TMs truly capture the "full power of computing". Namely, according to the thesis, any model of computation can be simulated by a TM (slightly more precisely, if there exists a mechanical process for computing some function f, then there also exists a Turing machine computing f). Thus, if our goal is to understand the nature of computing (as it is here), it suffices to restrict our attention to TMs. Note, however, that this is a *thesis*, and not a formally proven theorem — indeed, there does not appear to be any way to prove the Church-Turing thesis! And while it is generally believed that the Church-Turing thesis is on solid footing, a less well-known strengthening of the thesis, known as the *Extended Church-Turing thesis*, appears to potentially be in peril due to quantum computation (see section 1.3.4).

1.3.2 P and NP

P and NP are arguably the most famous exports of computational complexity theory. Roughly, they denote the sets of decision problems which can be solved efficiently on a TM, and verified efficiently on a TM, respectively. In this course, we define these complexity classes as follows.

Definition 1.5 (Polynomial-Time (P))**.** *A language $L \subseteq \{0,1\}^*$ is in P if there exists a (deterministic[1]) TM M and fixed polynomial $r_L : \mathbb{N} \to \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M halts in at most $O(r_L(n))$ steps, and:*

- *(Completeness/YES case) If $x \in L$, M accepts.*
- *(Soundness/NO case) If $x \notin L$, M rejects.*

Note that we define "efficient" computation, i.e. P, as taking at most *polynomially* many steps in the input size.

Exercise 1.6. Are all languages in P decidable?

Exercise 1.7. Can the choice of polynomial r_L also depend on the input, x? Why or why not?

A simple example of a decision problem in P is integer multiplication (MULTIPLY): Given as input $x, y \in \mathbb{Z}$, and threshold t, is the product $xy \leq t$? (All inputs are implicitly specified in binary.) The reverse of this problem is the integer factorization problem (FACTOR): Given

[1] In this course, all TM's are assumed to be deterministic.

$z \in \mathbb{Z}^+$ and threshold t, does z have a non-trivial factor $x \leq t$? In other words, do there exist integers $1 < x \leq t$ and y such that $xy = z$?

Exercise 1.8. You have implicitly known since elementary school that MULTIPLY is in P. Sketch a TM which decides MULTIPLY (i.e. recall your childhood algorithm for multiplication).

Exercise 1.9. If we change the definition of MULTIPLY to "given input $x, y \in \mathbb{Z}$, output the product xy", is this problem still in P?

There is a reason why multiplication is typically taught before factorization in school - unlike MULTIPLY, FACTOR is *not* known to be in P. It can, however, be efficiently *verified* by a TM if the TM is given some help in the form of a "short"/polynomial-size proof. Namely, given candidate factors $x, y \in \mathbb{Z}^+$ of z, a TM can efficiently check whether $x \leq t$ and whether $xy = z$. This is exactly the phenomenon encountered with puzzle games like Sudoku — filling out or *solving* the puzzle is difficult, but if someone gives you the solution, *verifying* the solution is correct is easy.

Definition 1.10 (Non-Deterministic Polynomial-Time (NP))**.** *A language $L \subseteq \{0,1\}^*$ is in NP if there exists a (deterministic) TM M and fixed polynomials $p_L, r_L : \mathbb{N} \to \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M takes in a "proof" $y \in \{0,1\}^{p_L(n)}$, halts in at most $O(r_L(n))$ steps, and:*

- *(Completeness/YES case) If $x \in L$, then there exists a proof $y \in \{0,1\}^{p_L(n)}$ causing M to accept.*
- *(Soundness/NO case) If $x \notin L$, then for all proofs $y \in \{0,1\}^{p_L(n)}$, M rejects.*

Thus, FACTOR $\in$ NP. Note the only difference between P and NP is the addition of the polynomial-size proof y. Also, recall the original definition of NP was in terms of *non-deterministic* TMs; here, we shall omit this view, as non-determinism appears generally less useful of a concept in quantum complexity theory.

The Boolean Satisfiability Problem. Finally, recall a canonical problem in NP is the Boolean k-Satisfiability Problem (k-SAT): Given a Boolean formula $\phi : \{0,1\}^n \to \{0,1\}$ in conjunctive normal form (CNF), is ϕ satisfiable? Here, a k-CNF formula has form (we demonstrate with an explicit example for $k = 3$)

$$\phi := (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_5} \vee x_3) \wedge \cdots \wedge (x_7 \vee x_4 \vee x_{11}),$$

where $\vee$ and $\wedge$ denote the logical OR and AND functions, respectively, and $\overline{x_i}$ is the negation of bit x_i. More generally, a k-CNF formula is an AND over "clauses" of size k, each of which is an OR over precisely k literals (a literal is either x_i or $\overline{x_i}$ for some variable x_i). We say ϕ is *satisfiable* if there exists $x \in \{0,1\}^n$ such that $\phi(x) = 1$.

Exercise 1.11. Why is k-SAT in NP? Does this still hold if k scales with n?

Exercise 1.12. Why is k-SAT not obviously in P?

Exercise 1.13. Is P $\subseteq$ NP? Is NP $\subseteq$ P?

1.3.3 The Cook-Levin Theorem, NP-completeness, and Reductions

What makes P versus NP such a prolific framework is that it captures many (if not most) practical computational problems of interest — from finding shortest paths between pairs of vertices in graphs, to solving systems of linear equations, to scheduling on multiple processors, all of these problems are in NP (with the first two also being in P). Indeed, the number of studied problems in NP number in the *thousands*.

But if the generality of P and NP is the "Eiffel Tower" of complexity theory, then the Cook-Levin Theorem is its "Arc de Triomphe" — for the Cook-Levin theorem showed that, remarkably, to solve *every* problem in NP efficiently (and thus most practical computational problems we might care about), it suffices to just solve a single problem — the Boolean Satisfiability problem. Coupled with Karp's "21 NP-complete problems", the community quickly realized that many longstanding problems which appeared difficult to solve, from 3-SAT to CLIQUE to KNAPSACK, are all one and the same problem as far as complexity theory is concerned.

We now recall the formal theory of NP-completeness, for which we first require the notion of a reduction.

Reductions. Intuitively, recall a reduction "reduces" one problem A to another problem B, so that if we can solve B, we can also solve A. A real-life example of this which you likely applied already at age two is this — the problem of opening the fridge (whose handle is too high up for a toddler!) can be reduced to getting a parent's attention; the ability to do the latter allows the toddler to accomplish the former. In this case, the reduction is being computed by the toddler; let us replace our toddler with a Turing machine.

Definition 1.14 (Reduction)**.** *Let $A, B \subseteq \{0,1\}^*$ be languages. A* reduction *from A to B is a computable[2] function $f : \{0,1\}^* \to \{0,1\}^*$, such that for any input $x \in \{0,1\}^*$, $x \in A$ if and only if $f(x) \in B$. If such a reduction exists, we write $A \leq B$. We further say the reduction is* polynomial-time *if the TM computing it runs in time polynomial in the input size, $|x|$; in this case, we write $A \leq_p B$.*

In essence, a reduction maps an instance x of problem A to an instance $f(x)$ of problem B such that x is a YES instance of A if and only if $f(x)$ is a YES instance of B.

Exercise 1.15. Define language ADD as the set of $(k+1)$-tuples $(x_1, \ldots, x_k, t) \subseteq \mathbb{Z}^{k+1}$, such that $x_1 + \cdots + x_k \leq t$. Give a reduction from MULTIPLY to ADD. (This reduction is, in fact, likely how you first learned the concept of multiplication in elementary school.) Is this reduction polynomial-time? (Hint: Think about the encoding size of the input to MULTIPLY.)

Many-one/Karp versus Turing reductions. The notion of reduction in Definition 1.14 is arguably the most natural one, as it maps a single instance of A to a single instance of B (note the mapping need not be a bijection). In the literature, this is typically called a *many-one*, *mapping*, or *Karp* reduction. A generalization of many-one reductions which also appears in quantum complexity theory is a *Turing* reduction. To define the latter, we refine our view of a Karp reduction. Suppose we have access to a "black box" or oracle O_B which magically decides B for us[3]. Then, a many-one reduction can be viewed as mapping instance x of A to instance $f(x)$ of

[2] A *computable* function f is one for which there exists a TM M which, given input x, halts and outputs $f(x)$.

[3] This is typically formalized via an oracle Turing machine. Such a TM has an extra tape, denoted the oracle tape, on which it gets to place a string corresponding to a query input. Once the TM's query input is ready on the query tape, the TM "calls" the oracle O_B, which replaces the query input with the query output on the query tape in a single step.

B, and then immediately feeding $f(x)$ to the oracle O_B to obtain the answer to the question: Is $x \in A$? In particular, we call the oracle O_B precisely once, and immediately return its output as our final answer. In a polynomial-time Turing reduction, we relax the latter constraint so that O_B can be called polynomially many times, and we may postprocess the answers to these queries using any polynomial-time computation we like before returning a final answer.

Exercise 1.16. Sketch a polynomial-time Turing reduction from the problem of finding a desired number in a sorted list of numbers to the problem of comparing which of a pair of integers is larger. What overhead does your reduction asymptotically require, if we treat each number as occupying a single cell of the TM's tape for simplicity? Does this overhead change if we allow the TM *random-access*, i.e. the ability to jump to any desired cell in a single time step?

There is formal evidence that many-one and Turing reductions are not the same in terms of "reduction power". A specific example in quantum information theory where this distinction seems to occur is the *Quantum Separability Problem*: Given a classical description of a bipartite quantum state, we wish to determine if the state is entangled. This problem is strongly NP-hard under polynomial time Turing reductions, but it remains open for over 15 years whether a similar result holds under polynomial-time many-one reductions. (Until 2019, a similar distinction also held for the genuinely "quantum" problem known as the CONSISTENCY problem[4], which since 2006 only had a known QMA-hardness proof under Turing reductions. Here, QMA is a quantum generalization of NP, and will be discussed in future lectures.)

NP-completeness. With reductions in hand, we can recall the definition of *NP-hard*, which in turn allows us to define NP-complete.

Definition 1.17 (NP-hard)**.** *A language $B \subseteq \{0,1\}^*$ is NP-hard if, for any language $A \in NP$, there exists a polynomial-time reduction from A to B.*

Definition 1.18 (NP-complete)**.** *A language $B \subseteq \{0,1\}^*$ is NP-complete if it is both in NP and NP-hard.*

Note that the definition of completeness for NP extends naturally to many other complexity classes. For example, we may replace NP with an arbitrary class C and define a lanuage as C-complete if it is both in C and C-hard. However, a subtlety in doing so is that depending on the class C, we may wish to use a different notion of reduction than "polynomial-time". For example, for P-complete problems, the notion of reduction used is sometimes a "logspace" reduction, meaning it should use at most a logarithmic amount of space. In this course, we shall clearly state (unless the instructor forgets) if a notion of reduction other than polynomial-time is used.

Intuitively, recall that NP-complete problems are the "hardest" problems in NP, and as far as polynomial-time reducibility is concerned, they are all equivalent in difficulty. (The latter statement is *not* generally true if one considers other definitions of "difficulty", such as *approximability* instead of exact solvability. As far as, say, approximability is concerned, some NP-complete problems really are "harder" than others.) In general, a C-complete problem for complexity class C should be thought of as "capturing" the difficulty of class C.

[4]CONSISTENCY is roughly defined as follows. The input is a set of k-qubit reduced density operators ρ_i acting on subsets of qubits $S_i \subseteq [n]$, respectively. The question is whether there exists a global density operator ρ acting on all n qubits, such that for each i, $\mathrm{Tr}_{[n]\setminus S_i}(\rho) = \rho_i$.

The Cook-Levin Theorem. The canonical NP-complete problem has traditionally been 3-SAT, as established by the Cook-Levin theorem (which showed NP-completeness for SAT) and Karp's followup work (which showed NP-completeness for 3-SAT). Here, SAT is the generalization of k-SAT in which the clauses can be of arbitrary size.

Theorem 1.19 (Cook-Levin theorem)**.** *SAT is NP-complete.*

An example of a reduction. Let us recall how reductions work in practice by showing 3-SAT is NP-hard under polynomial-time mapping reductions.

Lemma 1.20. 3-*SAT is NP-complete.*

Proof. Containment in NP is trivial. We sketch a reduction from SAT, which by Theorem 1.19 is NP-complete. The idea is as follows: Let $\phi : \{0,1\}^n \to \{0,1\}$ be an input SAT instance. We sketch how to efficiently construct a 3-SAT instance $\phi' : \{0,1\}^n \to \{0,1\}$, such that ϕ is satisfiable if and only if ϕ' is.

Let c be an arbitrary clause of ϕ of size s, so that we may write

$$c = (l_{c,1} \vee l_{c,2} \vee \cdots \vee l_{c,s})$$

for arbitrary literals $l_{c,i}$ (recall a *literal* is a variable which may or may not be negated). We show how to map this to a pair of clauses c_1 and c_2 of sizes $s-1$ and 3, respectively. Introduce a new variable y_c, and define

$$c_1 = (l_{c,1} \vee l_{c,2} \vee \cdots \vee l_{c,s-2} \vee y_c) \qquad c_2 = (l_{c,s-1} \vee l_{c,s} \vee \overline{y_c}).$$

Note that the new variable y_c will only occur in this particular pair of clauses. This operation is clearly polynomial-time, and repeating this until all clauses have size 3 also takes polynomial-time. We leave correctness as an exercise.

Exercise 1.21. Suppose $x_{c,1} \cdots x_{c,s}$ satisfies clause c above. Prove that there is a setting of y_c such that $x_{c,1} \cdots x_{c,s} y_c$ satisfies both c_1 and c_2. Similarly, show that if $x_{c,1} \cdots x_{c,s}$ does not satisfy clause c above, then $x_{c,1} \cdots x_{c,s}, y_c$ cannot satisfy both c_1 and c_2 regardless of the setting of y_c. □

Tip. For any reduction proof, your proof should be structured as follows. First, give the construction for the reduction itself. Second, argue what overhead or resources the reduction requires. Third, prove correctness of the reduction.

Exercise 1.22. Prove that 3-SAT remains NP-hard even if we restrict each variables x_i to appear at most in the 3-SAT formula ϕ. (Hint: Simulate equality constraints on pairs of variables.)

1.3.4 The Extended Church-Turing Thesis

Let us close this lecture by revisiting the "crossroads" stated in Section 1.1. Namely, we have reviewed the notions of P, NP, reductions, and NP-completeness. All of this relied heavily on the Turing machine model, which is "validated" by the Church-Turing thesis.

Due to Shor's factoring algorithm, we now know that one of the following statements must be *false*:

1. The Extended Church-Turing Thesis is true.

2. FACTOR cannot be solved efficiently on a classical computer.

3. Large-scale universal quantum computers can be built.

Formally proving any of these statements to be false would be a major step forward in theoretical computer science (although the first statement is arguably the least "earth-shattering" of the set, as it is less widely accepted). Here, the *Extended* Church-Turing Thesis says that any "reasonable" or "physically realizable" model of computation can be simulated by a Turing machine with at most polynomial overhead.

Any two of the three statements above together now imply the third statement is false[5]. For example, if FACTOR is intractable classically and large-scale quantum computers can be built, then the Extended-Church Turing thesis is false, since quantum computers *can* solve factoring efficiently. This is a best-case scenario for quantum computation. In the opposite direction, if the Extended Church-Turing Thesis is true and FACTOR is not classically tractable, then it must be that even though quantum computers can *theoretically* solve FACTOR, unfortunately a large-scale quantum computer cannot be built in practice.

Of course, here it should be noted that even in this "worst-case" latter scenario, all is not lost. Non-universal (i.e. special-purpose) quantum computers may nevertheless be possible on a large scale — indeed, there are already companies such as ID Quantique who build special-purpose quantum devices for cryptographic tasks such as quantum key distribution! (Another important point to bear in mind is that the development of quantum computation and information has had a dramatic impact on our understanding of theoretical physics, such as in the areas of entanglement theory, condensed matter physics, black holes, channel theory, etc... . This gained knowledge is unaffected by whether or not large-scale universal quantum computers can be built.)

[5]Some care is required when making this statement, since the first two statements of the "crossroads" are in computational models which are studied in the asymptotic limit of large input sizes, whereas strictly speaking, in practice the third statement refers to finite-size physical circuits where one does not necessarily care about extraordinarily large input sizes.

2 Quantum computation review

"I think that a particle must have a separate reality independent of measurements. That is, an electron has spin, location and so forth even when it is not being measured. I like to think the moon is there even if I am not looking at it."
— Albert Einstein

"... experiments have now shown that what bothered Einstein is not a debatable point but the observed behaviour of the real world."
— N. David Mermin

Introduction. In Lecture 1, we reviewed the basics of classical complexity theory, including Turing machines, P, NP, reductions, and NP-completeness. We now move to the quantum realm and review the basics of quantum computation. Again, we shall move rather quickly, as a beginning background in quantum computation is assumed for this course.

2.1 Linear Algebra

In this course, we shall discuss quantum computation exclusively from a finite-dimensional linear algebraic perspective. For this, we begin with a quick review of linear algebraic terminology and definitions.

We denote d-dimensional complex column vectors $|\psi\rangle \in \mathbb{C}^d$ using Dirac notation, i.e. as

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_d \end{pmatrix}, \tag{2.1}$$

for $\psi_i \in \mathbb{C}$. The term $|\psi\rangle$ is read "ket ψ". Recall also that a complex number $c \in \mathbb{C}$ can be written in two equivalent ways: Either as $c = a + bi$ for $a, b \in \mathbb{R}$ and $i = \sqrt{-1}$, or in its *polar form* as $c = re^{i\theta}$ for $r \in \mathbb{R}$ and $\theta \in [0, 2\pi)$. The *complex conjugate* of c is $c = a - bi$, or equivalently $c = re^{-i\theta}$.

Exercise 2.1. The *magnitude* or "length" of $c \in \mathbb{C}$ is given by $|c| = \sqrt{cc^*}$. What is the magnitude of $e^{i\theta}$ for any $\theta \in \mathbb{R}$? How about the magnitude of $re^{i\theta}$?

The *conjugate transpose* of $|\psi\rangle$ is given by

$$\langle\psi| = (\psi_1^*, \psi_2^*, \ldots, \psi_d^*), \tag{2.2}$$

where note $\langle\psi|$ is a *row* vector. The term $\langle\psi|$ is pronounced "bra ψ". This allows us to define how much two vectors "overlap" via the *inner product* function, defined as $\langle\psi|\phi\rangle = \sum_{i=1}^{d} \psi_i^* \phi_i$, which satisfies $(\langle\psi|\phi\rangle)^* = \langle\phi|\psi\rangle$. The "length" of a vector $|\psi\rangle$ can then be quantified by measuring the overlap of $|\psi\rangle$ with itself, which yields the *Euclidean norm*, $\| |\psi\rangle \|_2 = \sqrt{\langle\psi|\psi\rangle}$.

Exercise 2.2. Let $|\psi\rangle = \frac{1}{\sqrt{2}}(1, i)^T \in \mathbb{C}^2$, where T denotes the transpose. What is $\langle\psi|$? How about $\| |\psi\rangle \|_2$?

Orthonormal bases. A set of vectors $\{|\psi\rangle_i\} \subseteq \mathbb{C}^d$ is *orthogonal* if for all $i \neq j$, $\langle\psi|_i|\psi\rangle_j = 0$, and *orthonormal* if $\langle\psi|_i|\psi\rangle_j = \delta_{ij}$. Here, δ_{ij} is the Kroenecker delta, whose value is 1 if $i = j$ and 0 otherwise. For the vector space $\mathbb{C}^d$, which has dimension d, it is necessary and sufficient to use d orthonormal vectors in order to form an orthonormal basis.

One of the most common bases we use is the *computational* or *standard* basis, defined for $\mathbb{C}^d$ as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \cdots \quad |d-1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{2.3}$$

Since $\{|0\rangle, |1\rangle, \ldots, |d-1\rangle\}$ is an orthonormal basis, any unit vector $|\psi\rangle \in \mathbb{C}^d$ can be written as $|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$ for $\alpha_i \in \mathbb{C}$ satisfying the *normalization* condition $\| |\psi\rangle \|_2 = 1$.

Exercise 2.3. What does $\| |\psi\rangle \|_2 = 1$ mean in terms of a condition on the amplitudes α_i?

Linear maps and matrices. In this course, maps $\Phi : \mathbb{C}^d \to \mathbb{C}^d$ will typically be *linear*, meaning they satify for any $\sum_i \alpha_i |\psi_i\rangle \in \mathbb{C}^d$ that $\Phi\left(\sum_i \alpha_i |\psi_i\rangle\right) = \sum_i \alpha_i \Phi(|\psi_i\rangle)$. The set of linear maps from vector space $\mathcal{X}$ to $\mathcal{Y}$ is denoted $\mathcal{L}(\mathcal{X}, \mathcal{Y})$. For brevity, we use shorthand $\mathcal{L}(\mathcal{X})$ to mean $\mathcal{L}(\mathcal{X}, \mathcal{X})$.

Recall that linear maps have a *matrix* representation. A $d \times d$ *matrix* A is a two-dimensional array of complex numbers whose (i, j)th entry is denoted $A(i, j) \in \mathbb{C}$ for $i, j \in [d]$. To represent a linear map $\Phi : \mathbb{C}^d \to \mathbb{C}^d$ as a $d \times d$ matrix A_Φ, we use its action on a basis for $\mathbb{C}^d$. Specifically, define the ith column of A_Φ as $\Phi(|i\rangle)$ for $\{|i\rangle\}$ the standard basis for $\mathbb{C}^d$, or

$$A_\Phi = \left[\; \Phi(|0\rangle), \Phi(|1\rangle), \ldots, \Phi(|d-1\rangle) \; \right]. \tag{2.4}$$

In this course, we use both the matrix and linear map views interchangeably.

Exercise 2.4. Consider the linear map $\Phi : \mathbb{C}^2 \to \mathbb{C}^2$ with action $\Phi(|0\rangle) = |1\rangle$ and $\Phi(|1\rangle) = |0\rangle$. What is the 2×2 complex matrix representing Φ?

Exercise 2.5. Given any $d \times d$ matrix A, what is $A|i\rangle$ for $|i\rangle \in \mathbb{C}^d$ a standard basis state?

The product AB of two $d \times d$ matrices A and B is also a $d \times d$ matrix with entries $AB(i, j) = \sum_{k=1}^{d} A(i, k)B(k, j)$. Note that unlike for scalars, for matrices it is *not* always true that $AB = BA$. In the special case where $AB = BA$, we say A and B *commute*.

Exercise 2.6. Do the following Pauli X and Z matrices commute:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{and} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}? \tag{2.5}$$

The *image* of a matrix A is the set of all possible output vectors under the action of A, i.e. $\mathrm{Im}(A) := \{|\psi\rangle \in \mathbb{C}^d \mid |\psi\rangle = A|\phi\rangle \text{ for some } |\phi\rangle \in \mathbb{C}^d\}$. The *rank* of A is the dimension of its image, i.e. $\dim(\mathrm{Im}(A))$. The set of all vectors sent to zero by A is called its *null space*, i.e. $\mathrm{Null}(A) := \{|\psi\rangle \in \mathbb{C}^d \mid A|\psi\rangle = 0\}$. The *Rank-Nullity Theorem* says that these two spaces are related via $\dim(\mathrm{Null}(A)) + \dim(\mathrm{Im}(A)) = d$.

Exercise 2.7. Is the null space of matrix Z from Equation (2.5) non-empty? What is $\mathrm{rank}(Z)$?

Matrix operations. We will frequently apply the *complex conjugate*, *transpose* and *adjoint* operations to matrices in this course; they are defined, respectively, as

$$A^*(i,j) = (A(i,j))^* \qquad A^T(i,j) = A(j,i) \qquad A^\dagger = (A^*)^T. \tag{2.6}$$

Note that $(AB)^\dagger = B^\dagger A^\dagger$, and similarly for the transpose.

The *trace* is a linear map $\mathrm{Tr} : \mathcal{L}(\mathbb{C}^d) \to \mathbb{C}$ summing the entries on the diagonal of A, i.e. $\mathrm{Tr}(A) = \sum_{i=1}^d A(i,i)$. A wonderful property of the trace is that it is *cyclic*, i.e. $\mathrm{Tr}(ABC) = \mathrm{Tr}(CAB)$.

Exercise 2.8. In a previous exercise, you showed that X and Z do not commute. What is nevertheless true about $\mathrm{Tr}(XZ)$ versus $\mathrm{Tr}(ZX)$?

Outer products. Whereas the inner product mapped a pair of vectors $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$ to a scalar, the *outer product* produces a $d \times d$ matrix $|\psi\rangle\langle\phi| \in \mathcal{L}(\mathbb{C}^d)$. For example,

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad |1\rangle\langle 0| = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}. \tag{2.7}$$

More generally, the matrix $|i\rangle\langle j| \in \mathcal{L}(\mathbb{C}^d)$ has a 1 at position (i,j) and zeroes elsewhere. Thus, any matrix $A \in \mathcal{L}(\mathbb{C}^d)$ written in the computational basis can written $\sum_{ij} A(i,j)|i\rangle\langle j|$. We hence see that

$$\langle i|A|j\rangle = \langle i| \left(\sum_{i'j'} A(i',j')|i'\rangle\langle j'| \right) |j\rangle = \sum_{i'j'} A(i',j')\langle i|i'\rangle\langle j|j'\rangle = \sum_{i'j'} A(i',j')\delta_{ii'}\delta_{jj'} = A(i,j), \tag{2.8}$$

where the third equality follows since $\{|i\rangle\}$ forms an orthonormal basis for $\mathbb{C}^d$. In other words, $\langle i|A|j\rangle$ simply rips out entry $A(i,j)$.

Exercise 2.9. Observe that X from Equation 2.5 can be written $X = |0\rangle\langle 1| + |1\rangle\langle 0|$. What is $\langle 0|X|0\rangle$? How about $\langle 0|X|1\rangle$? How can you rewrite $\mathrm{Tr}(X)$ in terms of expressions of the form $\langle i|X|j\rangle$?

Eigenvalues and eigenvectors. Given any matrix $A \in \mathcal{L}(\mathbb{C}^d)$, an *eigenvector* is any non-zero vector $|\psi\rangle \in \mathbb{C}^d$ satisfying the equation

$$A|\psi\rangle = \lambda|\psi\rangle, \tag{2.9}$$

for some $\lambda \in \mathbb{C}$ which is the corresponding *eigenvalue*.

Exercise 2.10. Show that $|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ are eigenvectors of X from Equation (2.5). What are their respective eigenvalues?

A matrix $A \in \{\mathcal{L}(\mathbb{C}^d)\}$ is *normal* (i.e. satisfies $AA^\dagger = A^\dagger A$) if and only if it is unitarily diagonalizable, meaning it has *spectral decomposition*

$$A = \sum_{i=1}^{d} \lambda_i |\lambda_i\rangle\langle\lambda_i|, \tag{2.10}$$

where λ_i and $|\lambda_i\rangle$ are the eigenvalues and corresponding eigenvectors of A, respectively. Equivalently, there exists a unitary matrix (defined shortly) U such that $UAU^\dagger$ is diagonal. Note that if the eigenvalues λ_i are all distinct, then the eigenvectors $|\lambda_i\rangle$ are uniquely determined (and hence the spectral decomposition is unique). For normal operators, the eigenvectors form an orthonormal set. (Aside: It is worth noting that some non-normal matrices A may also be diagonalized, albeit with a similarity transformation more general than a unitary, i.e. by some invertible S such that SAS^{-1} is diagonal. In this case, the eigenvectors of A are no longer guaranteed to be orthonormal, but they are linearly independent. In this course, we will typically take "diagonalizable" to mean "unitarily diagonalizable".)

Exercise 2.11. Suppose A is unitarily diagonalizable and has two matching eigenvalues, e.g. $\lambda_1 = \lambda_2$. (We then say A is *degenerate*.) Prove that there are infinitely many eigenvectors $|\psi\rangle$ such that $A|\psi\rangle = \lambda_1|\psi\rangle$.

Using the spectral decomposition, we see that $\mathrm{Tr}(A)$ has a simple expression in terms of A's eigenvalues for diagonalizable A, namely $\mathrm{Tr}(A) = \sum_i \lambda_i$. Let us quickly prove this claim:

$$\mathrm{Tr}(A) = \mathrm{Tr}\left(\sum_i \lambda_i |\lambda_i\rangle\langle\lambda_i|\right) = \sum_i \lambda_i \mathrm{Tr}(|\lambda_i\rangle\langle\lambda_i|) = \sum_i \lambda_i \mathrm{Tr}(\langle\lambda_i|\lambda_i\rangle) = \sum_i \lambda_i. \tag{2.11}$$

Here, the second equality follows since the trace is linear, the third by the cyclic property of the trace, and the last since the eigenvectors $|\lambda_i\rangle$ are unit vectors.

Exercise 2.12. Prove that for diagonalizable A, $\mathrm{rank}(A)$ equals the number of non-zero eigenvalues of A.

Important classes of matrices. The following classes of matrices are ubiquitous in quantum information.

1. *Unitary matrices*: A matrix $U \in \mathcal{L}(\mathbb{C}^d)$ is *unitary* if $UU^\dagger = I$ (equivalently, $U^\dagger U = I$). Thus, all unitary matrices are invertible. The set of unitary matrices acting on space $\mathcal{X}$ is denoted $\mathcal{U}(X)$.

   Exercise 2.13. Prove that any eigenvalue of a unitary matrix U is of form $e^{i\theta}$ for some $\theta \in \mathbb{R}$. Thus, unitaries are high-dimensional generalizations of unit complex numbers.

2. *Hermitian matrices*: A matrix $M \in \mathcal{L}(\mathbb{C}^d)$ is *Hermitian* if $M = M^\dagger$. The set of Hermitian matrices acting on space $\mathcal{X}$ is denoted $\mathrm{Herm}\mathcal{X}$.

Exercise 2.14. Prove that any eigenvalue of a Hermitian matrix M is in $\mathbb{R}$. Thus, Hermitian matrices are high-dimensional generalizations of real numbers.

3. *Positive (semi-)definite matrices*: A Hermitian matrix with only positive (resp., non-negative) eigenvalues is called *positive definite* (resp., positive semidefinite). Thus, positive matrices generalize the positive (resp., non-negative) real numbers. We use $M \succ 0$ (resp., $M \succeq 0$) to specify that M is positive definite (resp. positive semidefinite). The set of positive semi-definite matrices acting on space $\mathcal{X}$ is denoted $\mathrm{Pos}(\mathcal{X})$.

   Exercise 2.15. Prove that the X and Z matrices are not positive semi-definite.

4. *Orthogonal projections:* A Hermitian matrix $\Pi \in \mathcal{L}(\mathbb{C}^d)$ is an *orthogonal projection* (or projector for short) if $\Pi^2 = \Pi$.

   Exercise 2.16. Prove a Hermitian matrix $\Pi \in \mathcal{L}(\mathbb{C}^d)$ is a projector if and only if all its eigenvalues are from set $\{0, 1\}$. Thus, projectors are high-dimensional generalizations of bits.

   Since a projector Π's eigenvalues are 0's and 1's, its spectral decomposition must take the form $\Pi = \sum_i |\psi_i\rangle\langle\psi_i|$, where $\{|\psi_i\rangle\}$ are an orthonormal set. Conversely, summing any set of orthonormal $\{|\psi_i\rangle\}$ in this fashion yields a projector. A projector Π has rank 1 if and only if $\Pi = |\psi\rangle\langle\psi|$ for some $|\psi\rangle \in \mathbb{C}^d$.

   Exercise 2.17. Let $\{|\psi_i\rangle\} \subseteq \mathbb{C}^d$ be an orthonormal set. Prove that $\Pi = \sum_i |\psi_i\rangle\langle\psi_i|$ is a projector.

   As for what a projector intuitively *does* — for any projector $\Pi = \sum_i |\psi_i\rangle\langle\psi_i|$ and vector $|\phi\rangle$,

$$\Pi|\phi\rangle = \left(\sum_i |\psi_i\rangle\langle\psi_i|\right)|\phi\rangle = \sum_i |\psi_i\rangle(\langle\psi_i|\phi\rangle) = \sum_i (\langle\psi_i|\phi\rangle)|\psi_i\rangle \in \mathrm{Span}(\{|\psi_i\rangle\}),$$

   where note $\langle\psi_i|\phi\rangle \in \mathbb{C}$. Thus, Π projects us down onto the span of the vectors $\{|\psi_i\rangle\}$.

   Exercise 2.18. Consider three-dimensional vector $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle \in \mathbb{C}^3$ and $\Pi = |0\rangle\langle 0| + |1\rangle\langle 1|$. Compute $\Pi|\phi\rangle$, and observe that the latter indeed lies in the two-dimensional space $\mathrm{Span}(\{|0\rangle, |1\rangle\})$.

Operator functions. A key idea used repeatedly in quantum information is that of an *operator function*, or in English, "how to apply real-valued functions to matrices". To apply function $f : \mathbb{R} \to \mathbb{R}$ to a Hermitian matrix $H \in \mathrm{Herm}\mathbb{C}^d$, we take the spectral decomposition $H = \sum_i \lambda_i |\lambda_i\rangle\langle\lambda_i|$, and define $f(H)$ as

$$H = \sum_i f(\lambda_i)|\lambda_i\rangle\langle\lambda_i|,$$

i.e. we apply f to the eigenvalues of H. Why does this "work"? Let us look at the Taylor series expansion of f, which for e.g. $f = e^x$ is (the series converges for all x)

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots . \tag{2.12}$$

The naive idea for defining e^H would be to substitute H in the right hand side of the Taylor series expansion of e^x:

$$e^H := I + H + \frac{H^2}{2!} + \frac{H^3}{3!} + \cdots . \tag{2.13}$$

Indeed, this leads to our desired definition; that to generalize the function $f(x) = e^x$ to Hermitian matrices, we apply f to the eigenvalues of H, as you will now show.

Exercise 2.19. Let H have spectral decomposition $H = \sum_i \lambda_i |\lambda_i\rangle\langle\lambda_i|$. Show that in Equation (2.13),

$$e^H = \sum_i e^{\lambda_i} |\lambda_i\rangle\langle\lambda_i|.$$

Exercise 2.20. Let $f(x) = x^2$. What is $f(X)$, for X the Pauli X operator? Why does this yield the same results as multiplying X by itself via matrix multiplication?

Exercise 2.21. Let $f(x) = \sqrt{x}$. For any pure state $|\psi\rangle \in \mathbb{C}^d$, define rank one density operator $\rho = |\psi\rangle\langle\psi|$. What is $\sqrt{\rho}$?

Exercise 2.22. What is $\sqrt{Z}$ for Z the Pauli Z operator? Is it uniquely defined?

2.2 Basic quantum computation

We now review the basics of quantum computation. Recall here there are two successively more general notions of quantum states we utilize. The first, *pure* states, are the quantum analogue of "perfect knowledge" about our state; in the classical world, a "pure state" means your computer's state is described by a fixed string $x \in \{0,1\}^n$. The second, and more general notion, is that of *mixed* states, which model the notion of "uncertainty" about our state. The classical analogue here would be a computer whose state is described by some *distribution* over n-bit strings x.

2.2.1 Pure state quantum computation

We begin by discussing pure state quantum computation.

Individual systems

Recall that an arbitrary d-dimensional pure quantum state is represented by a unit vector

$$|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle \in \mathbb{C}^d.$$

If we interpret quantum mechanics *literally* (i.e. adopt the "Copenhagen interpretation" of quantum mechanics), we take $|\psi\rangle$ to mean that our quantum system is in all d basis states $|i\rangle$ *simultaneously*, with some appropriate *amplitudes* $\alpha_i \in \mathbb{C}$. Typically, in this course we will work with $d = 2$, i.e. qubit systems.

Quantum gates

In the pure state setting, the set of allowable operations or *gates* on $|\psi\rangle \in \mathbb{C}^d$ is the set of $d \times d$ unitary matrices $U \in \mathcal{L}(\mathbb{C}^d)$ (i.e. $UU^\dagger = U^\dagger U = I$). In particular, this means pure-state quantum computation is fully reversible, since all gates have inverses.

You have already seen two of the three single-qubit Pauli matrices below, which are unitary. The fourth gate, H, is the Hadamard.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Exercise 2.23. What classical gate does Pauli X simulate? (Hint: Look at the action of X on $|0\rangle$ and $|1\rangle$.)

Exercise 2.24. What is the action of Pauli Z on the standard basis? Give the spectral decomposition of Z.

Recall the Z gate allows us to inject a *relative phase* into a quantum state. For example,

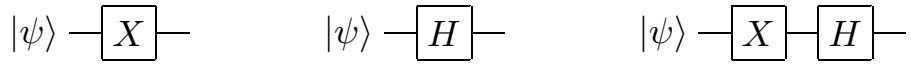
$$Z|+\rangle = Z\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}Z|0\rangle + \frac{1}{\sqrt{2}}Z|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle.$$

By relative phase, we mean that only the amplitude on $|1\rangle$ was multiplied by phase $e^{i\pi} = -1$. If *all* the amplitudes in the state were instead multiplied by $e^{i\pi}$, we could simply factor out the $e^{i\pi}$ from the entire state — in this case, $e^{i\pi}$ is a *global* phase, which cannot be detected via experiment, and hence is ignored.

The Hadamard, on the other hand, allows us to create or destroy certain superpositions. Namely, $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$, and $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$. In other words, H is self-inverse.

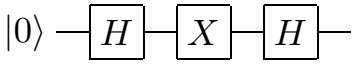
Exercise 2.25. Verify that X, Y, Z, H are all self-inverse, e.g. the inverse of X is just X. What does this mean about the eigenvalues of X? (Hint: Use the fact that the eigenvalues of any unitary must lie on the unit circle.)

In this course, we work with the *quantum circuit model*, which allows us to graphically depict gates:



These correspond to evolutions $X|\psi\rangle$, $H|\psi\rangle$, and $HX|\psi\rangle$, respectively. Each wire in such a diagram denotes a quantum system, and a box labelled by gate U depicts the action of unitary U. We think of time going from left to right; for the last circuit above, note that the X appears on the "left" in the circuit diagram but on the "right" in the expression $HX|\psi\rangle$; this is because X should be applied first to $|\psi\rangle$, then H.

Exercise 2.26. Which Pauli matrix does the following circuit simulate? (Hint: Use the spectral decomposition of X.)



Composite quantum systems

Thus far we have described single qudit systems. The mathematical formalism for describing the joint state for *multiple* qudits is the *tensor product*, $\otimes : \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2} \to \mathbb{C}^{d_1 \times d_2}$. For input vectors $|\psi\rangle \in \mathbb{C}^{d_1}, |\phi\rangle \in \mathbb{C}^{d_2}$, the (i,j)-th entry of their tensor product is $(|\psi\rangle \otimes |\phi\rangle)(i,j) := \psi_i \phi_j$, where recall ψ_i and ϕ_j are the ith and jth entries of $|\psi\rangle$ and $|\phi\rangle$, respectively. For example,

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}.$$

It is crucial to note that the tensor product *multiplies* the dimensions of its input spaces. This is why classical simulations of quantum mechanics appear to require an exponential overhead.

Exercise 2.27. What is $|+\rangle \otimes |0\rangle$ (expressed in the standard basis)?

Exercise 2.28. What dimension do n-qubit states live in, i.e. what is the dimension of space $(\mathbb{C}^2)^{\otimes n}$?

The tensor product has the following properties for any $|a\rangle, |b\rangle \in \mathbb{C}^{d_1}$ and $|c\rangle, |d\rangle \in \mathbb{C}^{d_2}$:

$$\begin{aligned}
(|a\rangle + |b\rangle) \otimes |c\rangle &= |a\rangle \otimes |c\rangle + |b\rangle \otimes |c\rangle && (2.14)\\
|a\rangle \otimes (|c\rangle + |d\rangle) &= |a\rangle \otimes |c\rangle + |a\rangle \otimes |d\rangle && (2.15)\\
c(|a\rangle \otimes |c\rangle) &= (c|a\rangle) \otimes |c\rangle = |a\rangle \otimes (c|c\rangle) && (2.16)\\
(|a\rangle \otimes |c\rangle)^\dagger &= |a\rangle^\dagger \otimes |c\rangle^\dagger = \langle a| \otimes \langle c| && (2.17)\\
(\langle a| \otimes \langle c|)(|b\rangle \otimes |d\rangle) &= \langle a|b\rangle\langle c|d\rangle. && (2.18)
\end{aligned}$$

For brevity, we shall often drop the notation $\otimes$ and simply write $|\psi\rangle \otimes |\phi\rangle = |\psi\rangle|\phi\rangle$.

Exercise 2.29. Using the properties above, prove that for orthonormal bases $B_1 = \{|\psi_0\rangle, |\psi_1\rangle\}$ and $B_2 = \{|\phi_0\rangle, |\phi_1\rangle\}$ for $\mathbb{C}^2$, the set $\{|\psi_0\rangle \otimes |\phi_0\rangle, |\psi_0\rangle \otimes |\phi_1\rangle, |\psi_1\rangle \otimes |\phi_0\rangle, |\psi_1\rangle \otimes |\phi_1\rangle\}$ is an orthonormal basis for $\mathbb{C}^4$.

Quantum entanglement. Recall that while any pair of states $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$ can be stitched together via the tensor product to obtain a d^2-dimensional state $|\psi\rangle \otimes |\phi\rangle \in \mathbb{C}^{d^2}$, the converse is not always true: Given any d^2-dimensional state $|\eta\rangle \in \mathbb{C}^{d^2}$, it is not always true that there exist $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$ satisfying $|\eta\rangle = |\psi\rangle \otimes |\phi\rangle$. Such $|\eta\rangle$ are called *entangled*.

For pure bipartite (i.e. two-party) states, entanglement is easy to characterize fully via the *Schmidt decomposition*, which says that any bipartite state $|\eta\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$ can be written

$$|\eta\rangle = \sum_{i=0}^{\min(d_1,d_2)-1} s_i |a_i\rangle|b_i\rangle,$$

for non-negative *Schmidt coefficients* s_i and orthonormal bases $\{|a_i\rangle\}_{i=0}^{d_1}$ and $\{|b_i\rangle\}_{i=0}^{d_2}$ for $\mathbb{C}^{d_1}$ and $\mathbb{C}^{d_2}$, respectively. The *Schmidt rank* of $|\eta\rangle$ is its number of non-zero Schmidt coefficients.

The canonical entangled two-qubit states are the Bell states

$$\begin{aligned}
|\Phi^+\rangle &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \\
|\Phi^-\rangle &= \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \\
|\Psi^+\rangle &= \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \\
|\Psi^-\rangle &= \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle,
\end{aligned}$$

where we simplified notation by letting (e.g.) $|0\rangle|0\rangle = |00\rangle$.

Exercise 2.30. Prove the Bell states are an orthonormal basis for $\mathbb{C}^4$.

It is worth mentioning that while the Schmidt rank of a bipartite pure state $|\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$ yields an efficient[1] test for entanglement in pure states, it is highly unlikely for there to be an efficient test for entanglement in mixed states. This is because determining whether a *mixed* state $\rho \in \mathcal{L}(\mathbb{C}^d \otimes \mathbb{C}^d)$ is separable is (strongly) NP-hard. (Mixed states are reviewed in Section 2.2.2.)

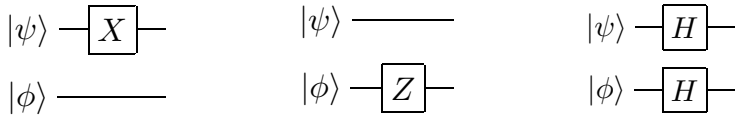
Two-qubit quantum gates. Two qubit gates are either a tensor product of one-qubit gates, such as $X \otimes Z$ or $H \otimes I$, or a genuinely two-qubit gate. For the former, recall the tensor product acts on matrices as

$$A = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}, \qquad B = \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix}, \qquad A \otimes B = \begin{pmatrix} a_1 \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} & a_2 \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} \\ a_3 \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} & a_4 \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} \end{pmatrix}.$$

The tensor product for matrices shares the properties of the tensor product for vectors, with the addition of two rules: $(A \otimes B)(C \otimes D) = AC \otimes BD$ and $\mathrm{Tr}(A \otimes B) = \mathrm{Tr}(A)\mathrm{Tr}(B)$.

Exercise 2.31. What is $\mathrm{Tr}((X \otimes X)(X \otimes X))$?

Circuit diagrams for tensor products of unitaries are depicted below: We consider the cases of $X \otimes I$, $I \otimes Z$, and $H \otimes H$, respectively.



Exercise 2.32. What is the circuit diagram for $Z \otimes Z$? What is $(X \otimes X)|0\rangle \otimes |1\rangle$? How about $(Z \otimes Z)|1\rangle \otimes |1\rangle$?

An important genuinely two-qubit gate is the *controlled-NOT* gate, denoted CNOT. The CNOT treats one qubit as the *control* qubit, and the other as the target *qubit*. It then applies the Pauli X gate to the target qubit only if the control qubit is set to $|1\rangle$. More precisely, the

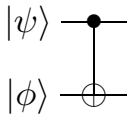
[1] "Efficient" here means the test can be computed in time polynomial in the *dimension*, d, of the system.

action of the CNOT on a two-qubit basis is given as follows, where qubit 1 is the control and qubit 2 is the target:

$$\text{CNOT}\,|00\rangle = |00\rangle \quad \text{CNOT}\,|01\rangle = |01\rangle \quad \text{CNOT}\,|10\rangle = |11\rangle \quad \text{CNOT}\,|11\rangle = |10\rangle.$$

Exercise 2.33. What is the matrix representation for CNOT?

The circuit diagram for the CNOT is given by



Exercise 2.34. What is $\text{CNOT}\,|\Phi^+\rangle$ for $|\Phi^+\rangle$ the Bell state? Based on this, give a circuit diagram mapping $|00\rangle$ to the Bell state $|\Phi^+\rangle$.

Measurement

Recall that measuring or *observing* a quantum system allows us to extract classical information from the system.

The most basic type of measurement is a *projective measurement*, given by a set of projectors $B = \{\Pi_i\}_{i=0}^m$ such that $\sum_{i=0}^m \Pi_i = I$, where the latter condition is the *completeness* relation. If each Π_i is rank one, i.e. $\Pi_i = |\psi_i\rangle\langle\psi_i|$, then we say B models a *measurement in basis* $\{|\psi_i\rangle\}$. Often, we shall measure in the computational basis for $\mathbb{C}^d$, which is specified by $B = \{|i\rangle\langle i|\}_{i=0}^{d-1}$ for standard basis vectors $|i\rangle \in \mathbb{C}^d$.

Exercise 2.35. Verify that $B = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ is a projective measurement on $\mathbb{C}^2$.

Given a projective measurement $B = \{\Pi_i\}_{i=0}^m \subseteq \mathbb{C}^d$ and quantum state $|\psi\rangle \in \mathbb{C}^d$, recall the probability of obtaining outcome $i \in \{0, \ldots, m\}$ when measuring $|\psi\rangle$ with B is given by

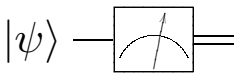
$$\Pr(\text{outcome } i) = \text{Tr}(\Pi_i|\psi\rangle\langle\psi|\Pi_i) = \text{Tr}(\Pi_i^2|\psi\rangle\langle\psi|) = \text{Tr}(\Pi_i|\psi\rangle\langle\psi|),$$

where the second equality follows by the cyclic property of the trace and the third since Π_i is a projector. Upon obtaining outcome i, our state $|\psi\rangle$ *collapses* to a state $|\psi'\rangle$ consistent with this outcome, i.e.

$$|\psi'\rangle = \frac{\Pi_i|\psi\rangle}{\|\,\Pi_i|\psi\rangle\,\|_2} = \frac{\Pi_i|\psi\rangle}{\sqrt{\langle\psi|\Pi_i\Pi_i|\psi\rangle}} = \frac{\Pi_i|\psi\rangle}{\sqrt{\langle\psi|\Pi_i|\psi\rangle}}.$$

Exercise 2.36. Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \in \mathbb{C}^2$. Show that if we measure in the computational basis, i.e. using $B = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, then the probabilities of obtaining outcomes 0 and 1 are $|\alpha|^2$ and $|\beta|^2$, respectively. What is the postmeasurement state $|\psi'\rangle$ if outcome 0 is obtained?

The circuit symbol denoting a measurement of state $|\psi\rangle \in \mathbb{C}^2$ in the *computational* basis is:



2.2.2 Mixed state quantum computation

Thus far, we have discussed pure state computation, where we know precisely the quantum state in which our system is throughout the computation. We now review *mixed state* computation, for which we recall the notion of density operators.

Recall that a *density operator* ρ acting on $\mathbb{C}^d$ is a $d \times d$ Hermitian matrix satisfying two properties: $\rho \succeq 0$ (ρ is positive-semidefinite) and $\mathrm{Tr}(\rho) = 1$. By the former, ρ has spectral decomposition

$$\rho = \sum_{i=1}^{m} p_i |\psi_i\rangle\langle\psi_i|,$$

with eigenvalues p_i and orthonormal basis $\{|\psi_i\rangle\}$. One way to interpret ρ is via the following experiment: With probability p_i, we prepare pure state $|\psi_i\rangle$. This, in particular, means that any pure state $|\psi\rangle$ has density matrix $|\psi\rangle\langle\psi|$. Conversely, any rank one density operator by definition must be of form $\rho = |\psi\rangle\langle\psi|$ (why?), and hence represents a pure state $|\psi\rangle$. The set of density operators acting on $\mathbb{C}^d$ is denoted $\mathcal{D}(\mathbb{C}^d)$.

Exercise 2.37. Prove that the eigenvalues $\{p_i\}$ of a density operator form a probability distribution.

Exercise 2.38. Write down the density operator $\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ and state vector $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. How do they differ?

Exercise 2.39. What is the density matrix for pure state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$?

The maximally mixed state. A special density matrix in $\mathcal{L}(\mathbb{C}^d)$ is the *maximally mixed* state $\rho = I/d$, which is the state of "maximum uncertainty". To see why, use the fact that for any orthonormal basis $\{|\psi_i\rangle\}_{i=1}^d$ for $\mathbb{C}^d$, $\sum_{i=1}^d |\psi_i\rangle\langle\psi_i| = I$. In other words, for *any* orthonormal basis $\{|\psi_i\rangle\}_{i=1}^d$, ρ represents the following experiment: Pick state $|\psi_i\rangle$ with probability $1/d$, and prepare $|\psi_i\rangle$. Since this holds for any basis, ρ gives us absolutely no information about which state $|\psi\rangle$ we actually have.

The partial trace operation. Density operators arise naturally in answering the question: For any entangled state $|\psi\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$, how do we describe the marginal state of $|\psi\rangle$ on (say) qubit 1? There is no way to answer this via pure states, since $|\psi\rangle$ is not a product state, and hence does not factorize. Instead, we require a density matrix to describe the state of qubit 1, and the correct procedure for obtaining it is the *partial trace* operation, which we now discuss.

For a bipartite density operator ρ system on parties A and B, the partial trace over B "discards" system B, and hence has signature $\mathrm{Tr}_B : \mathcal{L}(\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}) \to \mathcal{L}(\mathbb{C}^{d_1})$. To formally define Tr_B, recall that we may write the (usual) trace of $\rho \in \mathcal{L}(\mathbb{C}^d)$ as $\mathrm{Tr}(\rho) = \sum_i \rho(i,i) = \sum_{i=1}^d \langle i|\rho|i\rangle$. The *partial* trace over B applies this formula only to system B, i.e. for $\rho \in \mathcal{L}(\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2})$,

$$\mathrm{Tr}_B(\rho) = \sum_{i=1}^{d_2} (I_A \otimes \langle i|)\rho(I_A \otimes |i\rangle).$$

Exercise 2.40. What should $\mathrm{Tr}_B(I/4)$ intuitively be? Compute $\mathrm{Tr}_B(I/4)$ to check your guess.

Exercise 2.41. More generally, prove that $\mathrm{Tr}_B(\rho_A \otimes \rho_B) = \rho_A \cdot \mathrm{Tr}(\rho_B) = \rho_A$ for density matrices ρ_A, ρ_B.

Example 1: Separable states. We have said that a pure state $|\psi\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$ is not entangled, or *separable*, if and only if $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ for some $|\psi_1\rangle \in \mathbb{C}^{d_1}$ and $|\psi_2\rangle \in \mathbb{C}^{d_2}$. This idea extends to the setting of mixed states as follows: A bipartite density matrix $\rho \in \mathcal{L}(\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2})$ is unentangled or *separable* if

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \otimes |\phi_i\rangle\langle\phi_i|,$$

for some (possibly non-orthogonal) sets of vectors $\{|\psi_i\rangle\} \subseteq \mathbb{C}^{d_1}$ and $\{|\phi_i\rangle\} \subseteq \mathbb{C}^{d_2}$, and where the $\{p_i\}$ form a probability distribution. In other words, ρ is a probabilistic mixture of pure product states. An example of a separable state is

$$\rho = \frac{1}{2}|0\rangle\langle 0| \otimes |0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \otimes |1\rangle\langle 1|. \tag{2.19}$$

Since the partial trace is a linear map, and since we know that $\mathrm{Tr}_B(\rho_1 \otimes \rho_2) = \rho_1 \cdot \mathrm{Tr}(\rho_2) = \rho_1$ for density matrices ρ_1, ρ_2, computing the partial trace of ρ for separable states is simple:

$$\mathrm{Tr}_B\left(\sum_i p_i |\psi_i\rangle\langle\psi_i| \otimes |\phi_i\rangle\langle\phi_i|\right) = \sum_i p_i \mathrm{Tr}_B\left(|\psi_i\rangle\langle\psi_i| \otimes |\phi_i\rangle\langle\phi_i|\right) = \sum_i p_i |\psi_i\rangle\langle\psi_i| \cdot \mathrm{Tr}(|\phi_i\rangle\langle\phi_i|) = \sum_i p_i |\psi_i\rangle\langle\psi_i|.$$

Exercise 2.42. What is $\mathrm{Tr}_B(\rho)$ for ρ from Equation (2.19)?

Example 2: Pure entangled states. We compute the single-qubit state of the Bell state $|\Phi^+\rangle$ on qubit 1:

$$\begin{aligned}
\mathrm{Tr}_B(|\Phi^+\rangle\langle\Phi^+|) &= \frac{1}{2}\mathrm{Tr}_B(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|) \\
&= \frac{1}{2}|0\rangle\langle 0|\mathrm{Tr}(|0\rangle\langle 0|) + \frac{1}{2}|0\rangle\langle 1|\mathrm{Tr}(|0\rangle\langle 1|) + \frac{1}{2}|1\rangle\langle 0|\mathrm{Tr}(|1\rangle\langle 0|) + \frac{1}{2}|1\rangle\langle 1|\mathrm{Tr}(|1\rangle\langle 1|) \\
&= \frac{1}{2}I,
\end{aligned}$$

where we have used the linearity of the partial trace. Thus, the reduced state on qubit 1 for the Bell state is *maximally mixed*, i.e. it is a completely random state about which we have zero information.

Exercise 2.43. Show that $\mathrm{Tr}_A(|\Phi^+\rangle\langle\Phi^+|) = I/2$.

Operations and measurements on mixed states

We close this lecture by generalizing our discussion on gates and measurements from the pure state setting to mixed states.

Composite systems. If ρ_A and ρ_B are density operators, then $\rho_A \otimes \rho_B$ is a density operator.

Exercise 2.44. Prove the claim above. (Hint: The slightly trickier part is to show that the tensor product preserves positivity — use the spectral decomposition for this.)

Unitary operations. For density operator $\rho \in \mathcal{D}(\mathbb{C}^d)$ and unitary $U \in \mathcal{U}(\mathbb{C}^d)$, the action of U on ρ is given by $U\rho U^\dagger$.

Exercise 2.45. What is the action of any unitary $U \in \mathcal{U}(\mathbb{C}^d)$ on the maximally mixed state, $\rho = I/d$?

Measurements. For projective measurement $B = \{\Pi_i\}_{i=0}^m \subseteq \mathcal{L}(\mathbb{C}^d)$ applied to density operator $\rho \in \mathcal{D}(\mathbb{C}^d)$, the probability of outcome i and postmeasurement state $\rho' \in \mathcal{D}(\mathbb{C}^d)$ upon obtaining outcome i are

$$\Pr(\text{outcome } i) = \operatorname{Tr}(\Pi_i \rho) \qquad \text{and} \qquad \rho' = \frac{\Pi \rho \Pi}{\operatorname{Tr}(\Pi_i \rho)}.$$

Exercise 2.46. Show that the following important identity holds: For any bipartite state ρ_{AB} and matrix M_B acting on B, it holds that

$$\operatorname{Tr}(\rho_{AB} I_A \otimes M_B) = \operatorname{Tr}(\operatorname{Tr}_A(\rho_{AB}) M_B).$$

In other words, measuring just system B of a joint state ρ_{AB} is equivalent to first discarding system A of ρ_{AB}, followed a measurement on the reduced state on system B. Does this agree with your intuition of how a local measurement should behave?

3 Bounded error quantum polynomial time (BQP)

"An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it."
— Donald Knuth

Introduction. With our required complexity theory and quantum computation refreshers covered, we now focus on the cornerstones of quantum complexity theory — "quantum P" and "quantum NP". Defining the former of these will be arguably straightforward[1], but the same cannot be said for quantum "NP" — indeed, at present there is no "unique" definition of quantum NP, and for all we know, the various definitions currently in the literature may indeed be distinct entities.

This lecture begins by introducing the classical class BPP (Bounded-Error Probabilistic Polynomial Time), followed by BQP (i.e. "quantum Promise-BPP"). The lecture will introduce various fundamental concepts throughout, such as applications of norms to study error propagation in quantum circuits, as well as the notion of a quantum universal gate set. Note that while the quote atop this lecture is quite "hands-on" in nature, unfortunately we as a field are generally not in a position yet to simply "try out" quantum algorithms, at least on a large scale; indeed, we still have difficulty convincing ourselves the experiments we *do* run are doing what we expect them to do! (Such verification is a key issue in so-called "quantum supremacy" proposals, which may be discussed towards the end of the course.)

3.1 BPP

Recall from Lecture 2 that quantum computation is, in a rigorous sense, inherently probabilistic (i.e. we apply a unitary map to our input state, and then measure in the standard basis, yielding a probability distribution over outcomes). This raises the question of whether P is really the correct starting point for generalizing to BQP, since in stark contrast, P is about deterministic computation. A more suitable entry point seems to instead be *Bounded-Error Probabilistic Polynomial Time (BPP)*, which we now define.

Definition 3.1 (Bounded-Error Probabilistic Polynomial Time (BPP))**.** *A language $L \subseteq \{0,1\}^*$ is in BPP if there exists a (deterministic) TM M and fixed polynomials $s_L, r_L : \mathbb{N} \to \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M takes in string $y \in \{0,1\}^{s_L(n)}$, halts in at most $O(r_L(n))$ steps, and:*

- *(Completeness/YES case) If $x \in L$, then for at least $2/3$ of the choices of $y \in \{0,1\}^{s_L(n)}$, M accepts.*

[1] This is a borderline false statement; we will need highly non-trivial facts such as the Solovay-Kitaev theorem to make "quantum P" well-defined. But at least at a high level, the definition of "quantum P" is what one might expect.

- *(Soundness/NO case) If $x \notin L$, then for at most $1/3$ of the choices of $y \in \{0,1\}^{s_L(n)}$, M accepts.*

We say M accepts *(*rejects*) x in the YES case (NO case).*

There are three remarks worth making here. First, the definition of BPP looks suspiciously like NP — indeed, in NP we also had strings y, only we called them "proofs".

Exercise 3.2. How does the definition of BPP differ from that of NP?

The standard way to interpret y for BPP is as a uniformly random string over $\{0,1\}^n$. In other words, one thinks of a BPP machine M as a standard TM that also takes in a random string y as part of its input. With this view, in a YES case, M accepts with probability at least $2/3$, and in the NO case, M accepts with probability at most $1/3$. Second, under standard derandomization conjectures, it is generally believed that P = BPP. Proving this, however, is a longstanding open question.

Third, as phrased, BPP is a class of decision problems, i.e. every input x is either in L or not in L. This has a peculiar side-effect — on *any* input x, the machine M must accept or reject with probability at least $2/3$. It turns out that this subtle restriction is unnatural when moving to the quantum setting. Thus, BPP is *not* yet the correct starting point for generalizing to the quantum setting. Instead, we first need to move to *promise problems*.

3.1.1 Syntactic versus semantic classes, PromiseBPP, and what will really be PromiseBQP

To motivate PromiseBPP, we discuss syntactic versus semantic complexity classes.

Syntactic versus semantic classes. There is a subtle, but important difference, between how (say) P and NP are defined, as opposed to BPP. Namely, if we fix an appropriate encoding of TMs into strings, such that TM M has encoding $\langle M \rangle \in \{0,1\}^*$, then given a string $x \in \{0,1\}^*$, one can easily check if $x = \langle M \rangle$ for some P machine M. (For example, the encoding can explicitly state a threshold in terms of number of steps, after which the machine forcibly halts and outputs 0 or 1.) Complexity classes with this property are called *syntactic*. BPP, on the other hand, does not appear to have this property — how could any reasonable encoding of a TM M into strings encode the property that on all inputs, M accepts or rejects with probability at least $2/3$? (Indeed, deciding whether a TM has this property is undecidable, which follows from Rice's Theorem[2].) In other words, given $x \in \{0,1\}^*$, we cannot even decide whether x is a valid encoding of some BPP machine M!

Let us see how this causes problems. Consider the set of strings

$$L = \{\langle M, x \rangle \mid M \text{ is a P machine which accepts } x \in \{0,1\}^*\}.$$

Clearly, $L \in$ P, since a P machine can check if $\langle M \rangle$ encodes a valid P machine first, and if so, run M on x and output M's answer. Now, consider instead

$$A = \left\{\left\langle M, x, 1^t \right\rangle \mid M \text{ is a BPP machine which accepts } x \in \{0,1\}^* \text{ in at most } t \text{ steps}\right\}.$$

[2]Rice's Theorem says that, for any "non-trivial" property P, deciding whether a given TM M has property P is undecidable. Here, "non-trivial" means there exists at least one TM with property P, and not all TMs have property P.

Since BPP is a semantic class (i.e. we don't know how to check if $\langle M \rangle$ is a valid encoding of a BPP machine), A is *not* obviously in BPP. (Indeed, A is actually #P-complete, and thus unlikely to be in BPP!)

Exercise 3.3. Why doesn't the naive strategy we used for L show $A \in \text{BPP}$? In other words, why can a BPP machine not just try to simulate M on x?

This is a rather ridiculous state of affairs. Certainly the problem encoded by A is rather natural, and "feels like" it should be in "probabilistic polynomial time".

To circumvent this, we introduce a "promise" — namely, any algorithm attempting to decide A is "promised" that M is a valid BPP machine, and if this promise is "broken" (i.e. M is not a valid BPP machine), then A is allowed to act arbitrarily. We may formalize this as follows. Recall a *language* $L \in \{0,1\}^*$ partitions $\{0,1\}^*$ into two sets: L_{yes} (YES instances) and L_{no} (NO instances).

Definition 3.4 (Promise problem)**.** *A* promise problem $\mathcal{A}$ *is a partition of* $\{0,1\}^*$ *into* three *sets:* A_{yes} *(YES instances),* A_{no} *(NO instances),* A_{inv} *(invalid instances).*

Definition 3.5 (PromiseBPP)**.** *A promise problem* $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ *is in PromiseBPP if there exists a (deterministic) TM* M *and fixed polynomials* $s_A, r_A : \mathbb{N} \to \mathbb{R}^+$*, such that for any input* $x \in \{0,1\}^n$*,* M *takes in string* $y \in \{0,1\}^{s_A(n)}$*, halts in at most* $O(r_A(n))$ *steps, and:*

- *(Completeness/YES case) If* $x \in A_{\text{yes}}$*, then for at least* $2/3$ *of the choices of* $y \in \{0,1\}^{s_A(n)}$*,* M *accepts.*
- *(Soundness/NO case) If* $x \in A_{\text{no}}$*, then for at most* $1/3$ *of the choices of* $y \in \{0,1\}^{s_A(n)}$*,* M *accepts.*
- *(Invalid case) If* $x \in A_{\text{inv}}$*, then* M *may accept or reject arbitrarily.*

Exercise 3.6. What are the differences between the definitions of BPP and PromiseBPP?

Exercise 3.7. Why does the naive strategy for deciding L succeed in placing $\mathcal{A} \in \text{PromiseBPP}$?

Why all the fuss? There is a reason we are clearly delineating BPP from PromiseBPP here. The classical complexity theory community correctly distinguishes between BPP and PromiseBPP, as both are meaningful classes classically. The quantum complexity community, on the other hand, has a nasty habit of using terminology BQP to really mean PromiseBQP — this is not entirely surprising, as essentially any problem we care about quantumly is a promise problem. In an act of preserving sanity, the community has thus quietly decided not to carry around the "Promise" moniker each time BQP is mentioned. Let us hence clearly state the following, and promptly forget about it:

Throughout these notes, BQP shall be taken to mean PromiseBQP.

This distinction may seem petty, but it is actually crucial. For example[3], whereas BPP does

[3] Another illustration of this distinction is that BPP does not have time hierarchy theorems, but PromiseBPP does. Here, a time hierarchy theorem roughly says that a PromiseBPP machine running in, say, n^2 steps can solve more promise problems than a PromiseBPP machine running in only n steps. Intuitively, a time hierarchy theorem is exactly what one would expect to hold for a reasonable model of computation, so in this sense PromiseBPP seems more natural than BPP.

not have known complete problems (due to its semantic definition), we shall see in an upcoming lecture that BQP (i.e. PromiseBQP) *does* have complete problems.

3.2 BQP

BQP is the natural quantum generalization of PromiseBPP: It is the set of promise problems which can be solved by a poly-time quantum computation with bounded error. To make this formal, we first make a switch from Turing machines to circuits.

From Turing machines to circuits. Although quantum Turing machines are well-defined, it turns out that quantumly, the most natural computational model to work with is the *circuit model* (i.e. applying gates like Pauli matrices to wires, as done in Lecture 2). Unlike a TM, however, which can take strings $x \in \{0,1\}^*$ of arbitrary size as input, a circuit's size (by which we mean number of wires here) is *fixed*. Thus, given an input $x \in \{0,1\}^*$, we require an efficient algorithm for generating a description of a circuit C which takes inputs of the right size, $|x|$.

Definition 3.8 (P-uniform quantum circuit family)**.** *A family of quantum circuits $\{Q_n\}$ is called P-uniform if there exists a polynomial-time TM M, which given as input 1^n, outputs a classical description of Q_n.*

Exercise 3.9. Why does M take in 1^n as input, as opposed to n in binary as input?

We can now define BQP (which again, really means PromiseBQP).

Definition 3.10 (Bounded-error quantum polynomial-time (BQP))**.** *A promise problem $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in BQP if there exists a P-uniform quantum circuit family $\{Q_n\}$ and polynomial $q : \mathbb{N} \to \mathbb{N}$ satisfying the following properties. For any input $x \in \{0,1\}^n$, Q_n takes in $n+q(n)$ qubits as input, consisting of the input x in register A and $q(n)$ ancilla qubits initialized to $|0\rangle$ in register B. The first qubit of B, denoted B_1, is the designated output qubit, a measurement of which in the standard basis after applying Q_n yields the following (where measuring $|1\rangle$ in the output qubit B_1 denotes "accept"):*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, then Q_n accepts with probability at least $2/3$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then Q_n accepts with probability at most $1/3$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, Q_n may accept or reject arbitrarily.*

Exercise 3.11. Observe that a key difference between PromiseBPP and BQP is that the former can be viewed as a deterministic TM taking in a "random string" as input. In other words, the randomness could be "decoupled" from the TM. Do you think the randomness inherent in quantum cicuits could be "decoupled" from the quantum circuit Q_n in BQP?

Error reduction. Note that the completeness/soundness parameters $2/3$ and $1/3$ are not special (for both PromiseBPP and BQP). By simply repeating the circuit Q_n polynomially many times in parallel, measuring each output qubit, and then accepting if and only if a majority of the runs output 1, we may improve the error parameters to $1 - 2^n$ for completeness and 2^n for soundness.

Exercise 3.12. The error reduction claim above can be proven formally via the following *Chernoff bound*: If $X_1, \ldots, X_m$ are identically and independently distributed random variables over $\{0, 1\}$, such that each X_i has value 1 with probability $1/2 + \epsilon$, then the probability that $\sum_{i=1}^{m} X_i$ is strictly smaller than $m/2$ is at most $e^{-2\epsilon^2 m}$. With this bound in hand, how many repetitions m of a BQP circuit Q_n are required to get the completeness to at least $1 - 2^{-n}$?

3.2.1 Norms and perturbations to quantum gates

We will frequently need to measure distances between states and gates in this course; a crucial example of this will be in Section 3.2.2 below on universal gate sets. For this, we need to broaden our repertoire of norms.

Norms on matrices. Just as the Euclidean norm $\| |\psi\rangle \|_2$ gives a notion of "size" for vector $|\psi\rangle$, one may define norms on matrices. The two most common ones in quantum information are:

- **Spectral Norm:** The spectral norm of operator $M \in \mathcal{L}(\mathbb{C}^d)$ is

$$\| M \|_\infty := \max_{\text{unit vectors } |\psi\rangle \in \mathbb{C}^d} \| M|\psi\rangle \|_2 .$$

  Thus, $\| M \|_\infty$ measures the maximum amount M can "stretch" some vector in $\mathbb{C}^d$.

  Exercise 3.13. Define $M = 3|0\rangle\langle 0|$. What is $\| M \|_\infty$? Visualize the action of M in the 2D real Euclidean plane to intuitively confirm your calculation.

- **Trace Norm:** The trace norm of operator $M \in \mathcal{L}(\mathbb{C}^d)$ is

$$\| M \|_{\text{tr}} := \text{Tr}(|M|) = \text{Tr}\sqrt{M^\dagger M},$$

  where note $|M|$ and $\sqrt{M^\dagger M}$ are operator functions.

  Exercise 3.14. Let M be Hermitian with eigenvalues λ_i. Prove that $\| M \|_{\text{tr}} = \sum_i |\lambda_i|$.

  Exercise 3.15. For unit vectors $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$, prove that $\| |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi| \|_{\text{tr}} = 2\sqrt{1 - |\langle\psi|\phi\rangle|^2}$. Hint: Observe that $M = |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|$ is rank at most 2. Combined with the fact that $\text{Tr}(M) = 0$, this should reveal something about the structure of the eigenvalues of M. Finally, consider what the eigenvalues of $\text{Tr}(M^2)$ look like to complete the picture.

By definition of a *norm* (which should be thought of as a generalization of the absolute value function to higher dimensions), both the spectral and trace norms satisfy (1) the triangle inequality ($\| A + B \|_\infty \leq \| A \|_\infty + \| B \|_\infty$), (2) absolute homogeneity ($\| aM \|_\infty = |a| \, \| M \|_\infty$ for $a \in \mathbb{C}$), and (3) positive definiteness (if $\| M \|_\infty = 0$ then $M = 0$). However, they also satisfy three powerful properties:

- The Hölder inequality, which for the spectral and trace norms yields $\left|\text{Tr}(A^\dagger B)\right| \leq \| A \|_\infty \, \| B \|_{\text{tr}}$.
- Submultiplicativity, which says $\| AB \|_\infty \leq \| A \|_\infty \, \| B \|_\infty$.

- Invariance under unitaries, i.e. for any unitaries U and V, $\| UMV \|_\infty = \| M \|_\infty$.[4]

Motivating the trace norm. The trace norm may seem an odd quantity, but it has a remarkably important operational interpretation. Let us play a game: I have two density operators in mind, $\rho, \sigma \in \mathcal{L}(\mathbb{C}^d)$, of which you know a complete classical description. I privately flip a fair coin — if I get HEADS (respectively, TAILS), I prepare ρ (respectively, σ) as a quantum state and send it to you. Your task is to guess, using *any* measurement allowed by quantum mechanics (efficiently implementable or not), whether I sent you ρ or σ. Intuitively, if ρ and σ are "close" to one another, this should be "hard", and if they are "far", then, it should be "easy". It turns out that the right way to measure "closeness" is the trace norm, in that your *optimal* probability of winning this game is precisely

$$\frac{1}{2} + \frac{1}{4} \| \rho - \sigma \|_{\mathrm{tr}} .$$

This is a deep statement worth reflecting on — the trace norm characterizes how *distinguishable* two quantum states are.

Exercise 3.16. What is your probability of winning the game above if $\rho = \sigma$? How about if $\rho = |0\rangle\langle 0|$ and $\sigma = |1\rangle\langle 1|$?

Perturbations to quantum gate sequences. With norms in hand, we can study how errors affect quantum computations. Suppose we have a quantum circuit $U = U_m \cdots U_2 U_1$ for unitary operators U_i in mind, but experimentally we are not able to implement each U_i perfectly. Instead, for each i, we can implement some unitary U_i' satisfying $\| U_i - U_i' \| \leq \epsilon$ for $\| \cdot \|$ a unitarily invariant norm. The following useful lemma tells us how this per-error gate accumulates as we apply each U_i.

Lemma 3.17. *Let $\| \cdot \|$ be a norm satisfying unitary invariance. Let $U = U_m \cdots U_2 U_1$ and $U' = U_m' \cdots U_2' U_1'$ be a pair of quantum circuits, for unitaries U_i, U_i' satisfying $\| U_i - U_i' \| \leq \epsilon$ for all $i \in [m]$. Then, $\| U - U' \| \leq m\epsilon$.*

Proof. We proceed by induction on m. The base case $m = 1$ is trivial. Assume the claim holds for $m > 1$, and for brevity, let $V = U_{m-1} \cdots U_1$ and $V' = U_{m-1}' \cdots U_1'$. Then,

$$\begin{aligned} \| U - U' \| &= \| U_m V - U_m' V' + U_m V' - U_m V' \| \\ &= \| U_m (V - V') + (U_m' - U_m) V') \| \\ &\leq \| U_m (V - V') \| + \| (U_m' - U_m) V' \| \\ &= \| V - V' \| + \| U_m' - U_m \| \\ &\leq (m-1)\epsilon + \epsilon, \end{aligned}$$

where the first statement follows by adding and subtracting $U_m V'$, the third by the triangle inequality, the fourth by unitary invariance, and the fifth by the induction hypothesis. □

[4] There is an elegant reason why these norms satisfy invariance under unitaries — the norms are really "classical" norms in disguise, in that they do not depend on the choice of basis. Define for vector $|\psi\rangle \in \mathbb{C}^d$ the Schatten p-norm $\| |\psi\rangle \|_p = (\sum_{i=1}^d |\psi_i|^p)^{1/p}$, with $\| |\psi\rangle \|_\infty = \max_{i \in \{1,\ldots,d\}} |\psi_i|$. (What norm is recovered for $p = 2$?) Recall that every matrix $M \in \mathbb{C}^{d_1 \times d_2}$ can be written with respect to its *singular value decomposition* $M = \sum_{i=1}^{\min d_1, d_2} s_i |l_i\rangle\langle r_i|$, for singular values $s_i \in \mathbb{R}^+$, orthonormal left singular vectors $\{|l_i\rangle\} \subseteq \mathbb{C}^{d_1}$, and orthonormal right singular vectors $\{|r_i\rangle\} \subseteq \mathbb{C}^{d_2}$. (This should remind you of the Schmidt decomposition, which is no coincidence.) Then, the spectral and trace norms of A are just the ∞- and 1-norms applied to the vector of singular values of A, respectively. But since unitaries U, V applied as UMV leave the singular values invariant, it follows that these norms are unitarily invariant.

Thus, the error propagates nicely (i.e. linearly). As far as BQP is concerned, if we make an exponentially small additive error per gate, the overall error will hence be negligible (since BQP circuits contain polynomially many gates). Indeed, we can even get by with sufficiently small inverse polynomial additive error per gate.

Exercise 3.18. How can the requirement that the norm be unitarily invariant in Lemma 3.17 be relaxed? (Hint: Which of the other properties of the trace and spectral norms would also suffice?

How measurement statistics are affected by errors in quantum circuits. We have established how gate-wise error propagates through a circuit. Now comes the real question — as an experimenter in the lab looking at actual measurement statistics, what do the error bounds of Lemma 3.17 mean? For this, we state the following lemma, whose proof is the subsequent exercise.

Lemma 3.19. *Let $\rho \in \mathcal{D}(\mathbb{C}^d)$ be a quantum state, $\Pi \in \mathrm{Pos}\left(\mathbb{C}^d\right)$ a projector, and $U, V \in \mathcal{U}(\mathbb{C}^d)$ unitaries such that $\| U - V \|_{\mathrm{tr}} \leq \epsilon$. Then,*

$$\left|\mathrm{Tr}(\Pi U \rho U^\dagger) - \mathrm{Tr}(\Pi V \rho V^\dagger)\right| \leq 2\epsilon.$$

In words, any projective measurement outcome Π's probability changes by at most 2ϵ when evolving under V instead of U.

Exercise 3.20. Prove Lemma 3.19 using an approach similar to the proof of Lemma 3.17. (Hint: Begin by writing the left hand side as $\left|\mathrm{Tr}(\Pi(U\rho U^\dagger - V\rho V^\dagger))\right|$ and applying a useful inequality.)

3.2.2 Universal quantum gate sets

With Lemma 3.19 in hand, we can now discuss the fundamental notion of a *universal quantum gate set*. As computer scientists, in this course we shall measure the "cost" of a circuit by the number of gates it contains. From an experimental standpoint, however, since the set $\mathcal{U}(\mathbb{C}^d)$ is uncountably infinite, this seems somewhat unrealistic — implementing an arbitrary element of an uncountable set on demand is a tall order. Instead, we follow the classical route and recall that the NOT and AND gates together are *universal* for classical computation; in other words, any classical circuit can be rewritten using NOT and AND.

Amazingly, a similar statement holds quantumly — there exist finite universal gate sets $S \subseteq \mathcal{U}(\mathbb{C}^d)$, one of which is $S = \{H, \pi/8, CNOT\}$. Here,

$$\pi/8 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

S is universal in the intuive sense that any unitary $U \in \mathcal{U}(\mathbb{C}^d)$ can be approximated with gates from S. The question is: *How good is the approximation, and what overhead does it incur?*

There are two parts to the answer. First, an arbitrary unitary $U \in \mathcal{U}((\mathbb{C}^2)^{\otimes n})$ can be decomposed *exactly* as a product of CNOT gates and arbitrary 1-qubit gates. However, this decomposition is *not* necessarily efficient, requiring $O(n^2 4^n)$ gates in the worst case. But at least we are reduced to needing only a single 2-qubit gate, the CNOT, and now we need to approximate arbitrary 1-qubit gates. For this, the Solovay-Kitaev theorem says that an arbitrary

1-qubit gate can be approximated within ϵ additive error (with respect to, say, trace or spectral norm) using $O(\log^c(1/\epsilon))$ gates from $\{H, \pi/8\}$ for $c \approx 2$.

For the purposes of BQP, the overhead incurred by the Solovay-Kitaev theorem is very good — we may get sufficiently small inverse polynomial additive error with only a polylogarithmic overhead in gate count. As a result, BQP takes on a particularly well-motivated form: Without loss of generality, we may assume in Definition 3.10 that the Turing machine generating the P-uniform quantum circuit family only needs to pick gates from the universal set $S = \{H, \pi/8, CNOT\}$.

3.2.3 The BQP subroutine problem

The error reduction of BQP turns out to have a "boring" but crucial implication — if we have a BQP circuit solving a particular promise problem A, then we can use that circuit as a *subroutine* in solving other problems.

Classically, the analogous statement is trivial, as you know from the many times you have likely called (say) methods in Java as subroutines. So why is it non-trivial quantumly? Here is a sketch: Suppose we have a quantum circuit C, which applied to state $|x\rangle|0^m\rangle$ (for $x \in \{0,1\}^n$ the input) produces state $|\psi\rangle$ on $n+m$ qubits. If C accepts with probability exactly $2/3$ (i.e. $x \in A_{\text{yes}}$), then we may write

$$C|x\rangle|0^m\rangle = |\psi\rangle = \sqrt{\frac{1}{3}}|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \sqrt{\frac{2}{3}}|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}$$

for some $|\psi_0'\rangle, |\psi_1'\rangle \in (\mathbb{C}^2)^{\otimes(n+m-1)}$. In other words, the output qubit is potentially highly entangled with the remaining $n+m-1$ qubits (which happens, if say $|\psi_0'\rangle$ and $|\psi_1'\rangle$ are orthogonal). Since we are treating C as a subroutine, presumably we will only act in the future on this output qubit, and not the remaining $n+m-1$ qubits output by C. But recall the principle of deferred measurement states that we might as well thus assume that the remaining $2, \ldots, n+m-1$ have been discarded or *traced out*. Thus, if the output qubit is indeed entangled with the remaining qubits, we are in trouble — our machine will effectively be in a highly mixed state after we make the subroutine call! This is not at all what we were expecting. However, if we first reduce the error of C via parallel repetition, we know that we instead have (say)

$$C|x\rangle|0^m\rangle = |\psi\rangle = \sqrt{\frac{1}{2^n}}|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \sqrt{1-\frac{1}{2^n}}|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}.$$

While qubit 1 can still be entangled with qubits $2, \ldots n+m$ in this case, note that the vast majority of the amplitude is on the $|1\rangle_1$ branch — so tracing out qubits $2, \ldots, m+n$ will have only exponentially small disturbance on our state. Since our computation takes only polynomially many steps and is bounded error, such an exponentially small error at each step is negligible.

Exercise 3.21. One way to formalize the idea that the error is negligible is as follows. Define unit vectors:

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle_1|\psi_0'\rangle_{2,\ldots,n+m} + \beta|1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}, \\ |\phi\rangle &= |1\rangle_1|\psi_1'\rangle_{2,\ldots,n+m}. \end{aligned}$$

Prove that $\| |\psi\rangle - |\phi\rangle \|_2 = \sqrt{2}\sqrt{1-\text{Re}(\beta)}$, for $\text{Re}(\beta)$ the real part of β.

Exercise 3.22. Suppose $\| |\psi\rangle - |\phi\rangle \|_2 \leq \epsilon$. For any measurement projector Π, how can you upper bound the quantity $|\mathrm{Tr}(\Pi|\psi\rangle\langle\psi|) - \mathrm{Tr}(\Pi|\phi\rangle\langle\phi|)|$ in terms of ϵ? The precise formula is slightly messy, so it suffices to sketch which inequalities/equalities to chain together to derive a bound.

3.3 Relationship to other complexity classes

Given that quantum computers appear to outperform classical ones when it comes to the factoring problem, a natural question arises: Just how powerful are quantum polynomial-time computations? With BQP acting as a reasonable formal definition of what we view as "efficient quantum computation", we are in a position to make progress on this question.

Containment in PSPACE. The naive hope is that since quantum computers can put n-qubit systems into 2^n states simultaneously, perhaps one can simulate exponential-time classical computations in BQP. In other words, perhaps EXP $\subseteq$ BQP, for EXP the analogue of P in which the Turing machine may take exponentially many steps. This hope is dashed rather easily, however, via the following preliminary theorem, whose simple but elegant proof using a notion of "Feynman path integrals" repeatedly finds applications in the field.

Theorem 3.23. BQP $\subseteq$ PSPACE.

Recall that PSPACE is the set of decision problems decidable by a polynomial-*space* Turing machine, formally defined as follows.

Definition 3.24 (Polynomial-Space (PSPACE))**.** *A language $L \subseteq \{0,1\}^*$ is in* PSPACE *if there exists a (deterministic) TM M and fixed polynomial $s_L : \mathbb{N} \to \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M uses at most $O(s_L(n))$ cells on its work tape, and:*

- *(Completeness/YES case) If $x \in L$, M accepts.*
- *(Soundness/NO case) If $x \notin L$, M rejects.*

Exercise 3.25. How many steps can a PSPACE algorithm take in the worst case?

Exercise 3.26. Show that BQP $\subseteq$ EXP.

Proof of Theorem 3.23. Let x be an instance of any BQP promise problem $\mathcal{A} = (A_{\mathrm{yes}}, A_{\mathrm{no}}, A_{\mathrm{inv}})$ of length n. Then, there exists a poly-time TM M which given $|x|$ as input, outputs a quantum circuit $Q_n = U_m \cdots U_1$ of 1- and 2-qubit gates. Measuring the output qubit of M in the standard basis outputs 1 with probability at least $2/3$ if $x \in A_{\mathrm{yes}}$, or outputs 1 with probability at most $1/3$ if $x \in A_{\mathrm{no}}$. Thus, it suffices to sketch a PSPACE computation which estimates the probability of outputting 1 to within additive error, say, $1/12$.

Let $\Pi_1 = |1\rangle\langle 1| \in \mathcal{L}(\mathbb{C}^2)$ be a projective measurement onto the output qubit of Q_n, and $|\psi\rangle = Q_n|x\rangle|0^{q(n)}\rangle$. The probability of outputting 1 is

$$\Pr[\text{output } 1] = \mathrm{Tr}(\Pi_1|\psi\rangle\langle\psi|) = \langle\psi|\Pi_1|\psi\rangle.$$

Remark. *Note that $|\psi\rangle$ is an $n+q(n)$ qubit state, but Π_1 acts only on one qubit. To ensure dimensions match, throughout this course we implicitly mean $\Pi_1 \otimes I_{2,\ldots,n+q(n)}$ in such cases. This implies the probability of outputting 1 can equivalently be written*

$$\Pr[\textit{output } 1] = \mathrm{Tr}(\Pi_1 \cdot \mathrm{Tr}_{2,\ldots,n+q(n)}(|\psi\rangle\langle\psi|)).$$

Expanding the definition of $|\psi\rangle$,

$$\Pr[\text{output } 1] \quad = \quad \langle x|\langle 0^{q(n)}|U_m^\dagger \cdots U_1^\dagger \Pi_1 U_m \cdots U_1|x\rangle|0^{q(n)}\rangle.$$

Now, let us do something seemingly trivial — insert an identity operator I between each pair of operators, and use the fact that for N-qubit I, we may write $I = \sum_{x\in\{0,1\}^N} |x\rangle\langle x|$ (i.e. I diagonalizes in the standard basis with eigenvalues 1). Then:

$$\begin{aligned}\Pr[\text{output } 1] &= \langle x|\langle 0^{q(n)}| \cdot I \cdot U_m^\dagger \cdot I \cdots I \cdot U_1^\dagger \cdot I \cdot \Pi_1 \cdot I \cdot U_m \cdot I \cdots I \cdot U_1 \cdot I \cdot |x\rangle|0^{q(n)}\rangle \\ &= \sum_{x_1,\ldots,x_{2m+2}\in\{0,1\}^{n+q(n)}} (\langle x|\langle 0^{q(n)}|)|x_1\rangle\langle x_1|U_m^\dagger|x_2\rangle \cdots \langle x_{2m+1}|U_1|x_{2m+2}\rangle\langle x_{2m+2}|(|x\rangle|0^{q(n)}\rangle).\end{aligned}$$

For each fixed $x_1, \ldots, x_{m+2}$, note the summand is a product of $2m+3$ complex numbers of the form (for example) $\langle x_1|U_m^\dagger|x_2\rangle$. Since each U_i is at most a 2-qubit unitary, and since $\langle x_i|U|x_j\rangle$ is just the entry of U at row x_i and column x_j, we may compute this product in polynomial time.

Exercise 3.27. Convince yourself that one may compute $\langle x|I_{1,\ldots,n-1} \otimes U_n|y\rangle$ in polynomial time, for $x, y \in \{0,1\}^n$ and U a 1-qubit unitary acting on qubit n.

Exercise 3.28. There is a subtlety in the previous exercise we must be wary of: If each term in the summand is specified to (say) $O(1)$ bits, then how many bits can the product of all $2m+3$ terms have? Does this contradict your argument that the multiplication can be done in polynomial time? What if each term in the summand is specified to $O(n)$ bits?

Returning to the proof, we are essentially done — for each summand takes polynomially many steps on a TM to compute (and hence polynomial space), and the number of summands is $(2^{n+q(n)})^{2m+2}$, where by definition of BQP, m is polynomially bounded in n. Thus, although the number of summands is exponential in n, we can simply use a counter (requiring $\lceil \log((2^{n+q(n)})^{2m+2}) \rceil$ bits, i.e. polynomially many bits) to iterate through all summands one by one, each time adding the current summand to a running total. The final value of the running total is, by definition, the acceptance probability of circuit Q_n, as desired.

Exercise 3.29. How does the addition of all summands grow the number of bits required to represent the answer? In other words, given N n-bit rational numbers z_i, what is an upper bound on the number of bits required to represent $\sum_i z_i$?

Remark. *We noted at the outset of the proof that it would suffice to compute the acceptance probability of Q_n to within, say, $1/12$ additive error. So how accurate was our polynomial-space summation technique? This depends on how many bits of precision are used to specify the entries of each unitary U_i. If the gates U_i can be specified* exactly *using* $\mathrm{poly}(n)$ *bits, then the*

summation is perfect. *If, however, we have irrational entries in our gates, such as a* $1/\sqrt{2}$ *for the Hadamard, one approach is to approximate each entry up to* $p(n)$ *bits for* p *a sufficiently large polynomial. In other words, we can approximate each entry up to an additive error of* $2^{-p(n)}$. *One can then show that for sufficiently large but fixed* p*, the overall additive error over the entire summation can be made exponentially small in* n*, allowing us to distinguish YES from NO cases, as desired.* □

Better upper bounds. While the technique of Theorem 3.23 is elegant, PSPACE is not the best known upper bound on BQP. One can place BQP in the presumably smaller, but still powerful, class PP, which is the unbounded-error analogue of BPP.

Theorem 3.30. BQP $\subseteq$ PP.

Since we will discuss PP in the future, let us formally define it for completeness.

Definition 3.31 (Probabilistic Polynomial Time (PP))**.** *A language* $L \subseteq \{0,1\}^*$ *is in PP if there exists a (deterministic) TM* M *and fixed polynomials* $s_L, r_L : \mathbb{N} \to \mathbb{R}^+$*, such that for any input* $x \in \{0,1\}^n$*,* M *takes in string* $y \in \{0,1\}^{s_L(n)}$*, halts in at most* $O(r_L(n))$ *steps, and:*

- *(Completeness/YES case) If* $x \in L$*, then for strictly larger than* $1/2$ *of the choices of* $y \in \{0,1\}^{p_L(n)}$*,* M *accepts.*
- *(Soundness/NO case) If* $x \notin L$*, then for at most* $1/2$ *of the choices of* $y \in \{0,1\}^{p_L(n)}$*,* M *rejects.*

We say M accepts *(*rejects*)* x *in the YES case (NO case).*

There are various ways to prove Theorem 3.30. The approach we will take in an upcoming lecture is to actually prove that Quantum-Merlin Arthur (QMA), a quantum analogue of NP which trivially contains BQP, is itself contained in PP; this will be shown via *strong error-reduction* for QMA. Two other approaches use completely different techniques: The first is to show that PostBQP, which trivially contains BQP and is roughly BQP with the magical ability to "postselect" on certain outputs, equals PP. The second is to use a technique known as *hierarchical voting* to show that $\mathrm{P}^{\mathrm{QMA[log]}}$, which is the class of decision problems decidable by a P machine making logarithmically many queries to a QMA oracle (which hence trivially contains QMA, and thus also BQP), is also contained in PP.

Finally, let us close by remarking that, in addition to $\mathrm{P}^{\mathrm{QMA[log]}}$, another upper bound on QMA stronger than PP is known: BQP $\subseteq$ QMA $\subseteq$ $\mathrm{A_0PP}$ $\subseteq$ PP. And while $\mathrm{A_0PP}$ is a rather strange class, it does have the nice property that if $\mathrm{A_0PP} = \mathrm{PP}$, then PH $\subseteq$ PP, which is considered unlikely. Thus, as far as theoretical computer science is concerned, QMA (and hence BQP) are *strictly* contained in PP.

4 Linear systems of equations and a BQP-complete problem

> *"The method of Gaussian elimination appears in the Chinese mathematical text Chapter Eight: Rectangular Arrays of The Nine Chapters on the Mathematical Art ... parts of it were written as early as approximately 150 BCE ... Carl Friedrich Gauss in 1810 devised a notation for symmetric elimination that was adopted in the 19th century by professional hand computers to solve the normal equations of least-squares problems. The algorithm that is taught in high school was named for Gauss only in the 1950s as a result of confusion over the history of the subject."*
> — Wikipedia, `https://en.wikipedia.org/wiki/Gaussian_elimination#History`

Introduction. A classic problem with applications across just about any technical field is that of solving linear systems of equations. The task here is, given a set of N equations of the form $a_{i1}x_1 + \cdots + a_{iN}x_N = b_i$, for inputs $\{a_{ij}, b_i\} \in \mathbb{C}$ and variables $x_i \in \mathbb{C}$, find a simultaneous solution $\mathbf{x} \in \mathbb{C}^N$ to all equations. More compactly, one is given as input $A \in \mathcal{L}(\mathbb{C}^N)$ and target vector $\mathbf{b} \in \mathbb{C}^N$, and asked to find $\mathbf{x} \in \mathbb{C}^N$ satisfying $A\mathbf{x} = \mathbf{b}$.

In this lecture, we study a quantum algorithm for "solving" certain linear systems exponentially faster than known classically. We put the word "solving" in quotes, as the algorithm does not solve the system in the typical fashion of outputting $\mathbf{x} \in \mathbb{C}^N$; in fact, since the algorithm runs in time only *logarithmic* in N, we cannot even hope to output all of $\mathbf{x} \in \mathbb{C}^N$! Along the way, we shall see that the study of this algorithm reveals a natural problem complete for BQP — that of *matrix inversion*.

4.1 The basic idea: Quantum eigenvalue surgery

The basic principle behind the linear systems algorithm is elegantly simple, and allows for broader applications than just solving linear systems.

Ingredients. The ingredients we will need are as follows:

- The quantum phase estimation algorithm,
- a quantum Hamiltonian simulation algorithm, and
- postselection.

Problem specification. Suppose we have a Hermitian operator $A \in \mathrm{Herm}(\mathbb{C}^N)$ in mind, and a function $f : \mathbb{R} \to \mathbb{R}$. For example, f might be $f(x) = x^2$ or, in the case of linear systems solvers, $f(x) = x^{-1}$. We also have a vector $\mathbf{b} \in \mathbb{C}^N$. Our goal is to compute $f(A) \cdot \mathbf{b} \in \mathbb{C}^N$, where f is acting as an operator function. Classically, this computation would require at least $\Omega(N)$ time, since just writing out the solution vector takes $\Omega(N)$ time.

Now let us change the rules and make some assumptions. Suppose I have an efficient algorithm for preparing $\mathbf{b}$ as a *quantum state*, i.e. $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle \in \mathbb{C}^N$. Note that since $|b\rangle$ lives in an N-dimensional space, we require only $\lceil \log N \rceil + 1$ qubits to represent it. This also means the output vector I compute will presumably live in the same space, i.e. we shall compute

$$|x\rangle = f(A)|b\rangle \in \mathbb{C}^N,$$

which is also an $O(\log N)$-qubit state. Since I already assumed the ability to prepare $|b\rangle$ efficiently, it thus remains to simulate $f(A)$ via a quantum circuit. Before moving forward, however, some important observations:

1. The operator $f(A)$ might not be unitary. Thus, we can only hope to simulate its implementation *probabilistically*, and this is where the technique of postselection plays a role.

2. The output $|x\rangle$ may not be normalized as written above. However, any quantum state is implicitly normalized.

3. The operator $A \in \mathrm{Herm}\mathbb{C}^N$ is exponentially large in the number of qubits. We will therefore need to assume some appropriate form of "succinct access" to it, rather than writing it all down.

4. Finally, the output $|x\rangle$ is a *quantum* state, meaning we cannot just read all its entries. Rather, once we prepare $|x\rangle$, we can next perform any efficient measurement we like on $|x\rangle$. Thus, we may learn global properties of the state $|x\rangle$ quickly, but not all of its entries.

Algorithm sketch. Suppose A has spectral decomposition $A = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$. We first sketch the algorithm in the simple case where $|b\rangle = |\psi_j\rangle$, such that the goal is to compute

$$|x\rangle = f(A)|\psi_j\rangle = f(\lambda_i)|\psi_j\rangle. \tag{4.1}$$

As this expression reminds us, operator function f is just a function of the eigenvalues of A. Thus, to simulate $f(A)$, we shall manually "extract" these eigenvalues, "process" them to simulate application of f, and then "reinsert" them where we found them.

Step 1: Eigenvalue extraction. Since A is Hermitian and not necessarily unitary, we cannot implement it directly. However, recall that $U = e^{iA}$ *is* unitary, and has eigenvalues $e^{i\lambda_j}$. In principle, we may hence compute $U|\psi_j\rangle = e^{i\lambda_j}|\psi_j\rangle$. But to "process" λ_j, we need to extract it out of the phase and into a register, i.e. to instead compute $|\lambda_j\rangle|\psi\rangle$, so that we can next try to map this to $|f(\lambda_j)\rangle|\psi\rangle$.

Any time one wishes to "bring a quantity to or from a phase", a helpful tool tends to be the Quantum Fourier Transform (QFT). Indeed, using the QFT in a clever way, and given the ability to apply high powers U in a controlled fashion, the Quantum Phase Estimation (QPE) algorithm yields the mapping

$$|\psi_j\rangle \mapsto |\lambda_j\rangle|\psi_j\rangle.$$

(Above, it is implicitly assumed ancillae initialized to $|0\rangle$ are added over the course of the mapping to store λ_j.) Brief reviews of the QFT and QPE are given in Section 4.1.1.

Exercise 4.1. Is there an *a priori* bound on how many bits λ_j requires to represent? What would be an ideal way to cram, say, an irrational λ_j into a finite-size register using bit strings?

Step 2: Eigenvalue processing. With the eigenvalue λ_j sitting in a register, we now simply coherently compute f via a classical algorithm to map

$$|\lambda_j\rangle|\psi_j\rangle \mapsto |f(\lambda_j)\rangle|\lambda_j\rangle|\psi_j\rangle. \tag{4.2}$$

Step 3: Eigenvalue reinsertion. This step is actually a sequence of steps, since we need to uncompute any auxilliary data we produced along the way.

As a first step, recall that our aim in Equation (4.1) was to compute $f(\lambda_i)|\psi_j\rangle$ (as opposed to $|f(\lambda_i)\rangle|\psi_j\rangle$). Having accomplished the latter, we simulate the former as follows: Conditioned on the register containing $|f(\lambda_j)\rangle$ we may produce state

$$|\lambda_j\rangle|\psi_j\rangle \mapsto |\lambda_j\rangle|\psi_j\rangle \left(\sqrt{1-f(\lambda_j)^2}|0\rangle + f(\lambda_j)|1\rangle\right),$$

where the last register is a single qubit. This is just a single-qubit rotation on the third register, conditioned on the contents of the first register.

Exercise 4.2. What is the difference between $f(\lambda_i)|\psi_j\rangle$ and $|f(\lambda_i)\rangle|\psi_j\rangle$?

Exercise 4.3. What range of $f(\lambda_i)$ is permitted in the equation above?

If we now measure the last register in the standard basis and obtain 1, we collapse our state onto essentially what we wanted:

$$f(\lambda_j)|\lambda_j\rangle|\psi_j\rangle|1\rangle.$$

All that is left is to "uncompute the garbage". Namely, we apply a Pauli X to $|1\rangle$ to return it to state $|0\rangle$, and we invert the phase estimation algorithm on $|\psi_j\rangle$ to map $|\lambda_j\rangle|\psi_j\rangle$ back to $|0\cdots 0\rangle|\psi_j\rangle$. We may now safely discard the registers which have been reinitialized back to all zeroes, obtaining Equation (4.1).

Applying the algorithm to general states $|b\rangle \in \mathbb{C}^N$. In Equation (4.1), we assumed $|b\rangle$ was an eigenstate of H for simplicity. To now extend this to arbitrary $|b\rangle$, we give Nature a "high 5" and thank it for being linear, because our analysis actually extends trivially due to the facts that (1) quantum mechanics is linear and (2) we may write

$$|b\rangle = \sum_{j=1}^{N} \beta_j |\psi_j\rangle,$$

since recall $\{|\psi_i\rangle\} \subseteq \mathbb{C}^N$ is an orthonormal basis for $\mathbb{C}^N$. Thus, by linearity the algorithm will correctly map

$$\sum_{j=1}^{N} \beta_j |\psi_j\rangle \mapsto \sum_{j=1}^{N} \beta_j f(\lambda_j)|\psi_j\rangle,$$

where we stress the right-hand side is *not* normalized as written.

Exercise 4.4. Why did we need to uncompute the garbage at the send of Step 3 earlier? How might omitting this uncomputation cause a problem when we move to the setting of general $|b\rangle$?

4.1.1 Aside: Brief review of the QFT and QPE

Since the procedure we've sketched crucially uses the QFT and QPE, let us briefly review how these components work. (We will only use them as black boxes in this lecture, but they are used sufficiently frequently to warrant a refresher.)

The Quantum Fourier Transform (QFT). Recall the *Fourier transform* is a unitary operation, meaning it is just a change of basis. A powerful change of basis it is, however, as it can be viewed as mapping the "time domain" to the "frequency domain". For this reason, it is ubiquitous in areas such as signal processing (meaning you can thank the Fourier transform for making mp3 sound files possible), and even allows one to multiply pairs of n-bit integers in essentially $O(n \log n)$ time (as opposed to $O(n^2)$ time via the grade school multiplication algorithm). The *Quantum* Fourier Transform (QFT) similarly finds key uses in quantum algorithms, particularly when one wishes to move a quantity from a register up into a phase, or vice versa.

We may completely specify the N-dimensional $\mathrm{QFT_N}$ via its action on the standard basis:

$$|j\rangle \xrightarrow{\mathrm{QFT_N}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle.$$

As previously noted, observe that it lifts string j into the phase.

Exercise 4.5. Prove that $H = \mathrm{QFT}_2$. In other words, you have already been using the QFT.

A full recap of the QFT circuit would be distracting for the purposes of this lecture; we refer the reader to Chapter 5 of Nielsen and Chuang for a full exposition. We shall simply note that the basic implementation of the QFT requires quadratically many gates in the number of qubits; for us, this means a runtime of $O(\log^2 N)$, where recall N is our dimension.

Quantum Phase Estimation (QPE). With the QFT in hand, we may perform QPE. Specifically, we assume we have a black-box implementation of some unitary $U \in \mathcal{U}(\mathbb{C}^N)$, and an eigenvector $|\psi_j\rangle$ of U with eigenvalue $e^{2\pi i \phi_j}$ for some $\phi_j \in [0, 1)$ (without loss of generality). Our goal is to compute the n most significant bits of ϕ_j. Actually, we require a stronger assumption than just the ability to run U — we must be able to perform a controlled-U^k operation, which applies U k times to a target register, conditioned on a control register being set to $|1\rangle$. This is *not* a trivial assumption.

Exercise 4.6. Suppose $U \in \mathcal{U}((\mathbb{C}^2)^{\otimes n})$ has a polynomial-size quantum circuit implementation of 1- and 2-qubit gates. Is it true that for all such U, controlled-U^k can be implemented efficiently for $k \in \Theta(2^n)$? (Hint: Try to embed a brute force search algorithm over all assignments to a given 3-SAT formula $\phi : \{0,1\} \to \{0,1\}$ into U^k for $k = 2^n$.)

The algorithm for QPE is now sufficiently simple that we include it for completeness:

1. Start with initial state $|\eta_0\rangle = |0^t\rangle|\psi_j\rangle$, for $t = n + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$. Here, ϵ will dictate the success probability of the procedure.

2. Applying $H^{\otimes n}$ to the first register, we obtain state $|\eta_1\rangle = \left(\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle\right) |\psi_j\rangle$.

3. Apply the controlled-U operation, with register 1 as control and register 2 as the target, to obtain

$$|\eta_2\rangle = \left(\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} e^{2\pi i k \phi_j} |k\rangle\right) |\psi_j\rangle.$$

Exercise 4.7. To what power do we need to raise U in this step in the worst case?

4. This expression now has our desired phase ϕ_j encoded in the phase; in fact, if ϕ_j can be encoded exactly using n bits, then the first register is just the QFT_{2^t} of $|2^t \phi_j\rangle$ (and in this case we could have set $t = n$). Therefore, applying $\mathrm{QFT}_{2^t}^\dagger$ yields $|\eta_3\rangle = |2^t \phi_j\rangle |\psi_j\rangle$, from which we can compute ϕ_j.

Recall that in general, we have no guarantee that ϕ_j can be expressed in n bits. This is intuitively why we require $t > n$ ancilla bits; an appropriate analysis shows that choosing $t = n + p$ yields failure probability at most $[2(2^p - 1)]^{-1}$.

Exercise 4.8. Suppose the input state to the QPE algorithm is not an eigenvector $|\psi_j\rangle$ of U, but rather an arbitrary state $|\psi\rangle \in \mathbb{C}^N$. Sketch the action of each step of the QPE algorithm on $|\psi\rangle$.

4.2 A quantum algorithm for linear systems of equations

We now show how to instantiate the algorithm sketch of Section 4.1 in the setting of solving linear systems $A\mathbf{x} = \mathbf{b}$. Let us assume for simplicity that A is Hermitian and full rank (the algorithm can be modified to work even without these assumptions). Then, recall from elementary Linear Algebra that the system has the unique solution $\mathbf{x} = A^{-1}\mathbf{b}$. But now note A^{-1} is an operator function $f(x) = x^{-1}$ applied to Hermitian operator A. Thus, we can try to apply the "eigenvalue surgery" technique of Section 4.1 to "manually invert" the eigenvalues of A. Before proceeding, we must discuss a key quantity known as the *condition number* of A, which is relevant to both classical and quantum linear systems solvers.

4.2.1 Condition numbers of matrices

Whereas in theory, any full rank matrix A can be inverted, *in practice* this cannot necessarily be done reliably. For some choices of A, the computation of A^{-1} is very sensitive to numerical error, and this is captured by the notion of *condition number*.

Formally, suppose we are only able to represent $\mathbf{b}$ via a numerical approximation, i.e. we numerically compute $\mathbf{b}' = \mathbf{b} + \epsilon$ for some error vector ϵ. We are interested in understanding how $A^{-1}\mathbf{b}'$ compares with $A^{-1}\mathbf{b}$. By linearity, this depends on how $A^{-1}\mathbf{b}$ compares with $A^{-1}\epsilon$ — if the former is not much larger than the latter, the error will be quite noticeable. So let us look at the ratio between these two, normalized by the ratio of $\mathbf{b}$ and ϵ themselves, and where we employ the Euclidean norm:

$$\frac{\| A^{-1}\epsilon \|_2}{\| A^{-1}\mathbf{b} \|_2} \cdot \frac{\| \mathbf{b} \|_2}{\| \epsilon \|_2} = \frac{\| A^{-1}\epsilon \|_2}{\| \epsilon \|_2} \cdot \frac{\| \mathbf{b} \|_2}{\| A^{-1}\mathbf{b} \|_2}.$$

To maximize this ratio over *all* $\mathbf{b}$ and ϵ, we maximize the left quantity (giving us $\left\| A^{-1} \right\|_\infty$ by definition of the spectral norm) and minimize the right quantity (giving us $\| A \|_\infty$). This worst case ratio is precisely how we define the condition number,

$$\kappa(A) := \left\| A^{-1} \right\|_\infty \| A \|_\infty .$$

Thus, $\kappa(A)$ roughly captures the worst-case sensitivity of A, over all choices of $\mathbf{b}$, to encoding errors in $\mathbf{b}$.

Exercise 4.9. In a previous lecture, we defined the spectral norm slightly differently — instead of dividing by $\| \epsilon \|_2$ as in $\left\| A^{-1}\epsilon \right\|_2 / \| \epsilon \|_2$ above, we required ϵ to be a unit vector. Prove that both definitions of the spectral norm are equivalent.

Exercise 4.10. Prove that the minimum of $\left\| A^{-1}\mathbf{b} \right\|_2$ over all vectors $\mathbf{b}$ is indeed $\| A \|_\infty$. Hint: One easy way to see this is to note that $\| A \|_\infty$ is the largest singular value of A.

Exercise 4.11. What is $\kappa(A)$ for any unitary A?

Exercise 4.12. What is $\kappa(A)$ for rank deficient A? How does this support the idea that A is by definition non-invertible?

4.2.2 Assumptions for the algorithm

The following assumptions are required for the algorithm (in addition to the non-crucial assumptions that A is Hermitian and full rank):

1. That $\| A \|_\infty = 1$, and hence $\left\| A^{-1} \right\|_\infty = \kappa(A)$. Under this assumption, once the postselection step passes, the algorithm is guaranteed to be correct. This assumption can be relaxed, in exchange for allowing further errors (i.e. we would need to also consider the "ill-conditioned" subspace of A).

2. An efficient unitary implementation of $|b\rangle \in \mathbb{C}^N$. This is treated as a genuine black box.

3. An efficient Hamiltonian simulation algorithm to implement unitary e^{iAt}. Recall here $A \in \mathrm{Herm}(\mathbb{C}^N)$, whereas efficient means with respect to the number of qubits, $O(\log N)$. This is a highly non-trivial assumption.

   Exercise 4.13. Prove that the following is false: For any Hermitian A and evolution time $t \in \mathbb{R}$, there is a quantum circuit with size polylog(N) simulating e^{iAt}. (Hint: Recall the equivalence with arbitrary unitary operators $U \in \mathcal{U}(\mathbb{C}^N)$, and apply a basic counting argument to show the latter cannot have poly-size circuits for all U.)

   Luckily, there are some fairly broad classes of Hamiltonians which we *can* simulate efficiently — these include s-sparse Hamiltonians A, which have two important properties: (1) At most s non-zero entries per row, and (2) there exists a poly-time classical algorithm which, given a row index $r \in [N]$, outputs the non-zero entries in row r. In this case, there exist quantum algorithms for simulating e^{iAt} with error at most ϵ_H (with respect to trace distance) in time $\widetilde{O}(\log(N)s^2t)$, where the tilde means slower growing terms are omitted for simplicity. Note this runtime has been simplified using the assumption here that $\| A \|_\infty \leq 1$.

4.2.3 The algorithm

For simplicity in the analysis, we shall make the following additional assumptions: (1) The state $|b\rangle$ can be prepared without error. (2) $A \succeq 0$, which means that since we assumed $\| A \|_\infty = 1$ and $\left\| A^{-1} \right\|_\infty = \kappa(A)$ in Section 4.2.2, we have that $\lambda_{\max}(A) = 1$ and $\lambda_{\min}(A) = 1/\kappa(A)$. (3) All eigenvalues $1/\kappa(A) \leq \lambda_j \leq 1$ of A require at most n bits to represent, for some integer $n > 0$. (This n will shortly play a role in terms of run-time.) (4) Simulating e^{iAt} can be done without error for any time $t \geq 0$ (with some associated cost, of course).

Exercise 4.14. Prove that if $A \succeq 0$, then $\| A \|_\infty = \lambda_{\max}(A)$ and $\left\| A^{-1} \right\|_\infty = \lambda_{\min}(A)$.

The algorithm now follows the sketch of Section 4.1:

1. Use the assumed black box to prepare state
$$|b\rangle = \sum_{j=1}^{N} \beta_j |\psi_j\rangle \in \mathbb{C}^N,$$
where $|\psi_j\rangle$ are the eigenvectors of A with eigenvalues λ_j.

2. Apply QPE (for unitary e^{iA}) with an n-qubit ancilla to our state $|b\rangle$ to obtain
$$\sum_{j=1}^{N} \beta_j |2^n \lambda_j\rangle |\psi_j\rangle \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^N.$$

3. Conditioned on the first register, rotate a new single-qubit ancilla as follows:
$$\sum_{j=1}^{N} \beta_j |2^n \lambda_j\rangle |\psi_j\rangle \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)} \right) |1\rangle \right) \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^N \otimes \mathbb{C}^2. \tag{4.3}$$
Note that the third register is now entangled with the first two.

   Exercise 4.15. Which of our assumptions guarantee that $1/\kappa(A) \leq 1/(\lambda_j \kappa(A)) \leq 1$, and hence that the amplitudes above are well-defined?

4. Apply the inverse of the QPE algorithm to the first two registers to obtain
$$\sum_{j=1}^{N} \beta_j |0^n\rangle |\psi_j\rangle \left(\sqrt{1 - \frac{1}{\lambda_j^2 \kappa^2(A)}} |0\rangle + \left(\frac{1}{\lambda_j \kappa(A)} \right) |1\rangle \right) \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^N \otimes \mathbb{C}^2.$$
We may hence now discard the first register safely, having uncomputed it.

   Exercise 4.16. Why does applying the QPE^{-1} operation produce the state above, given that the third register in Equation (4.3) is entangled with the first two? (Hint: With respect to which basis did we define the action of QPE?)

5. Measure the third register in the standard basis, postselect on outcome 1, and then discard the third register to obtain a state
$$\sum_{j=1}^{N} \beta_j \left(\frac{1}{\lambda_j \kappa(A)} \right) |\psi_j\rangle \equiv \sum_{j=1}^{N} \beta_j \left(\frac{1}{\lambda_j} \right) |\psi_j\rangle \propto A^{-1} |b\rangle \in \mathbb{C}^N.$$

Exercise 4.17. The use of "equivalence" $\equiv$ above is not an accident — why are the first two expressions equivalent for the purposes of quantum information?

Exercise 4.18. Prove that in Step 5, the probability of obtaining outcome 1 is at least $1/\kappa^2(A)$.

We conclude, by the exercise above, that the expected number of repetitions to obtain $|1\rangle$ in Step 5 is $O(\kappa^2(A))$. Using the technique of quantum amplitude amplification, this can be reduced to $O(\kappa(A))$ repetitions; we omit the details of this step here, but assume it has been implemented in our runtime analysis below.

Runtime. Let T_b denote the number gates required to prepare $|b\rangle$. The total runtime we have hence accrued is $\widetilde{O}(\kappa(A)(T_b + ts^2\log(N)))$, where t is the time we run Hamiltonian simulation for in the QPE algorithm. Since we assumed all eigenvalues of A require at most n bits to represent exactly, this means QPE will simulate e^{iAt} for $t \in O(2^n)$ (in the notation of QPE from Section 4.1.1, there we would set $t = n$). Thus, the total runtime scales as

$$\widetilde{O}(\kappa(A)(T_b + 2^n s^2\log(N))),$$

assuming we wish to compute the *exact* answer proportional to $A^{-1}|b\rangle$, and assuming all subroutines we have called run without error. Recalling that the number of qubits is $O(\log(N))$ for A an $N \times N$ matrix, this means that if $n \in O(\log\log N)$ (i.e. QPE approximates phases up to additive inverse polynomial error in the number of qubits), we would get runtime $\widetilde{O}(\kappa(A)(T_b + s^2\log^2(N)))$. In the regime $\kappa(A), T_b, s \in \text{polylog}(N)$, this is exponentially faster than classical algorithms explicitly solving the entire $N \times N$ system.

Of course, we cannot assume $n \in O(\log\log N)$ *a priori* (since we know just about nothing about the spectrum of A), but a more involved algorithm and analysis can be used to show that if one wishes to output state $|\widetilde{x}\rangle$ satisfying $\| |x\rangle - |\widetilde{x}\rangle \|_2 \leq \epsilon$, it suffices to set $t \in O(\kappa(A)/\epsilon)$, obtaining a final runtime of

$$\widetilde{O}(\kappa(A)(T_b + \kappa(A)^2 s^2\log(N))/\epsilon).$$

This more advanced algorithm can also handle e.g. non-Hermitian A, dropping the assumption that $\kappa(A) = \left\| A^{-1} \right\|_\infty$, and relaxing the simplifications that all subroutines run perfectly (e.g. they may fail with some probability).

On optimality. In order to maintain an efficient runtime in the number of qubits, $O(\log N)$, the error ϵ permitted above must be at most inverse polynomial in the number of qubits. This is generally sufficient for the purposes of BQP to distinguish between YES and NO cases of promise problems. Nevertheless, it is natural to ask: *Can the runtime of this algorithm be improved to something* polylogarithmic *in $\kappa(A)$ and ϵ?*

It turns out that if the goal is to *classically estimate* the quantity $\langle x|\Pi|x\rangle$ for some efficiently implementable projector Π, then this is highly unlikely. (For example, it would imply BQP = PSPACE if a runtime in $\kappa(A)^{1-\delta}$ for $\delta > 0$ a constant would be possible. More on such ideas in Section 4.3.) However, if we relax the requirements so that the goal is to produce the quantum state $|x\rangle$ (i.e. this is now a different beast altogether, as the output is no longer classical), then an improvement *is* possible. Namely, recent advances in Hamiltonian simulation techniques have led to an improved linear systems solver which has runtime polynomial in $\log(1/\epsilon)$, as opposed to $1/\epsilon$ here. In other words, exponentially good approximations to $|x\rangle$ can be prepared efficiently quantumly (assuming A is well-conditioned, etc).

Exercise 4.19. Can you see where the polynomial dependence on $1/\epsilon$ in the runtime must come in, i.e. which step of the algorithm we described here cannot be run to full precision in general, even assuming all subroutines work perfectly?

Exercise 4.20. Why does the ability to prepare $|x\rangle$ efficiently to within 2^{-n} Euclidean distance not allow one to estimate the classical value $\langle x|\Pi|x\rangle$ to within 2^{-n} additive error efficiently (i.e. why does the improved algorithm not contradict the known lower bounds)? (Hint: Imagine we could even prepare $|x\rangle$ perfectly.)

4.3 A BQP-complete problem: Matrix inversion

At the end of the Section 4.2.3, we touched on the topic of optimality. It turns out that task of *matrix inversion*, which is the core subroutine of the linear systems algorithm, characterizes the complexity of BQP, in that it is a BQP-complete problem. Let us formalize it as follows.

Definition 4.21 (Matrix inversion problem (MI))**.** *The promise problem* $MI = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ *is as follows.*

- *Input: An* $O(1)$*-sparse invertible Hermitian matrix* $A \in \text{Herm}(\mathbb{C}^N)$ *satisfying* $\kappa^{-1}(A) \preceq A \preceq I$ *for* $\kappa(A) \in \text{polylog}(N)$*, specified via a polynomial-time Turing machine* M *which, given any row index* $r \in [N]$ *of* A*, outputs the* $O(1)$ *non-zero entries of* A*.*
- *Output: Let* $|\widetilde{x}\rangle \propto A^{-1}|0^N\rangle$ *be a unit vector, and* $\Pi = |1\rangle\langle 1| \in \mathcal{L}(\mathbb{C}^2)$ *a projector onto the first qubit of* $|\widetilde{x}\rangle$*.*
  - *(Completeness) If* $\text{Tr}(\Pi|\widetilde{x}\rangle\langle\widetilde{x}|) \geq 2/3$*, output YES.*
  - *(Soundness) If* $\text{Tr}(\Pi|\widetilde{x}\rangle\langle\widetilde{x}|) \leq 1/3$*, output NO.*
  - *(Invalid) Else, output YES or NO arbitrarily.*

Theorem 4.22. *MI is BQP-complete under polynomial-time many-one reductions.*

Proof. Containment of MI in BQP follows immediately from the linear systems algorithm. We show BQP-hardness. Let $V = V_m \cdots V_1$ be a BQP circuit acting on n qubits, and of size $m \in \text{poly}(n)$, for $n \in \log(N)$. Without loss of generality, we may assume m is a power of 2. We give a polynomial-time many-one reduction to a matrix A as in the definition of MI.

Let us begin with an idea that almost works. Define

$$U = \sum_{t=0}^{m-1} |t+1\rangle\langle t| \otimes V_{t+1} + \sum_{t=m}^{2m-1} |t+1 \bmod 2m\rangle\langle t| \otimes V_{2m-t}^\dagger \in \mathcal{U}((\mathbb{C}^2)^{\otimes \log m} \otimes (\mathbb{C}^2)^{\otimes n}),$$

where recall we implicitly assume V_t (which is a 2-qubit gate) is tensored with an identity on all $n-2$ qubits it does not act on.

Exercise 4.23. Show that U is indeed unitary.

Exercise 4.24. Show that $U^m|0^{\log m}\rangle|0^n\rangle = |m\rangle V|0^n\rangle$. Thus, measuring the first qubit of the second register simulates measuring the output qubit of the BQP computation V.

Exercise 4.25. For what values of k is $U^k|0\rangle|0^n\rangle = |0\rangle|0^n\rangle$ necessarily?

We would like to embed U in a matrix A, so that A^{-1} can be nicely expressed via U. The natural idea is to recall the Maclaurin series expansion

$$\frac{1}{1-x} = \sum_{l=0}^{\infty} x^l$$

which holds for $|x| < 1$. Applying this to U, we find

$$(I-U)^{-1} = \sum_{l=0}^{\infty} U^l.$$

Exercise 4.26. Technically, the last equation above is *not* correct (nevertheless, it provides the correct intuition; we will correct the error later). What requirement for the Maclaurin series expansion have we violated?

Defining $A = I - U$ and $|\widetilde{x}\rangle \propto A^{-1}|0^{\log m+n}\rangle$ a unit vector, we have

$$|\widetilde{x}\rangle \propto \sum_{l=0}^{\infty} U^l|0\rangle^{\log m}|0^n\rangle \propto |0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle.$$

Exercise 4.27. Again, technically, the last statement above is not exactly correct — not all terms on the right hand side will be equally weighted. Which two terms should be weighted slightly less than the others? (For pedagogical reasons, we nevertheless work with the statement above, as the discrepancy will not affect our high-level discussion other than to complicate the amplitudes involved.)

Since each term $|j\rangle V_j \cdots V_1|0^n\rangle$ is a unit vector, this means

$$|\widetilde{x}\rangle = \frac{1}{\sqrt{m+1}} \left(|0\rangle|0^n\rangle + |1\rangle V_1|0^n\rangle + \cdots + |m\rangle V_m \cdots V_1|0^n\rangle\right).$$

Thus, if we measure the first register of $|\widetilde{x}\rangle$ in the standard basis, with probability $1/(m+1)$ we collapse onto state $|m\rangle V|0^n\rangle$. Let E_m (respectively, E_0) be the event we postselect in the first register on $|m\rangle$ (respectively, $|1\rangle, \ldots, |m-1\rangle$). Then, conditioned on E_m, the probability of measuring 1 in the first qubit of register reveals the answer of the BQP computation with probabilty at least 2/3; otherwise, conditioned on E_0, we may assume[1] without loss generality that V rejects with probability 1. This implies (where recall $\Pi = |1\rangle\langle 1|$ is a single qubit projector onto the output qubit of the second register):

- If V denotes a YES instance, then $\langle\widetilde{x}|\Pi|\widetilde{x}\rangle \geq \frac{2}{3(m+1)}$.
- If V denotes a NO instance, then $\langle\widetilde{x}|\Pi|\widetilde{x}\rangle \leq \frac{1}{3(m+1)}$.

[1] One way to achieve this is to modify V so that in all but the last time step, V sets the output qubit to $|0\rangle$, and only in time step m does V perform a CNOT to copy over its answer to the output qubit.

Exercise 4.28. Prove the probability bounds claimed above.

This is almost what we want, except for three things: (1) A must be $O(1)$-sparse. (2) We should have probability bounds $2/3$ (completeness) and $1/3$ (soundness) for MI. (3) A should be an invertible Hermitian matrix $A \in \mathrm{Herm}(\mathbb{C}^N)$ satisfying $\kappa^{-1}(A) \preceq A \preceq I$ for $\kappa(A) = \mathrm{polylog}(N)$.

Exercise 4.29. Prove that U is $O(1)$-sparse. Conclude that A is also $O(1)$-sparse.

Exercise 4.30. How can one boost the probability bounds from $2/(3(m+1))$ and $1/(3(m+1))$ to $2/3$ and $1/3$? (Hint: Two tricks can be used together. First, use error reduction for BQP. Second, what happens if before constructing U, we modify the circuit V by appending M time steps in which nothing is done?)

Exercise 4.31. Is A necessarily invertible?

Obtaining the third desired property is slightly trickier. We begin by bringing the condition number of A down to polylogarithmic in N, which we now discuss.

Exercise 4.32. What is the worst case condition number $\kappa(I - U)$ for a unitary U?

Exercise 4.33. Show that defining $A = I - \frac{1}{2}U$ has $\kappa(A) \in O(1)$. Where in the previous analysis does this choice of A cause problems, however?

To circumvent the problems in the previous exercise, we choose $A = I - e^{-1/m}U$ for scalar $e^{-1/m}$.

Exercise 4.34. Prove that $\kappa(A) \in O(m)$ for the new definition of A. (Hint: You only need to use the fact that U is unitary, not the specific definition of U. Also, use the fact that for normal operators A, the singular values of A are $\{|\lambda(A)|\}$ (why?).)

Exercise 4.35. Show that A is now invertible.

Exercise 4.36. Rerun the analysis with our new choice of A and show that we are still able to distinguish between YES and NO cases for V with constant completeness-soundness gap.

With a bounded condition number in place, to make A Hermitian we apply the same trick for dealing with non-Hermitian matrices U as in the linear systems algorithm (which we shall again omit; roughly, one creates the anti-block diagonal matrix

$$\begin{pmatrix} 0 & I - e^{-1/m}U \\ I - e^{-1/m}U^\dagger & 0 \end{pmatrix},$$

which is clearly Hermitian, and whose inversion can be shown to also yield a similar final result to inverting $I - U$.) Finally, note that the requirement $\kappa^{-1} \preceq A \preceq I$ does not make sense until we map A to a Hermitian matrix, since the "$\preceq$" partial order is defined on the set of Hermitian matrices (just as the usual "$\leq$" total order is defined on the set of real numbers, but not on

complex numbers). Once A is Hermitian, however, a global rescaling of A will suffice to satisfy this final constraint, which we also omit. □

5 Quantum Merlin Arthur (QMA) and strong error reduction

"I have had my results for a long time, but I do not yet know how to arrive at them."
— Carl F. Gauss.

"If only I had the theorems! Then I should find the proofs easily enough."
— Georg B. Riemann.

Introduction. We have thus far defined BQP, studied the task of "solving" linear systems, and shown that matrix inversion is BQP-complete. We now wish to define a quantum analogue of NP. This is unfortunately a somewhat delicate issue; indeed, there are almost as many known quantum generalizations of NP as the story of Snow White has dwarves — there's QMA, QMA_1, QMA(2), QCMA, StoqMA, and NQP. With this said, there is a *de facto* definition of "quantum NP" used by the community: Quantum Merlin Arthur (QMA).

In this lecture, we begin by defining Merlin Arthur (MA) and Quantum Merlin Arthur (QMA). We then study the surprising *strong error reduction* property of QMA. Finally, we close by discussing the relationship of QMA to known complexity classes. As suggested by the opening quotes of this lecture, a key theme will be the power of proofs (particularly quantum proofs); as with NP, these proofs will in general be hard to produce, but easy to verify.

5.1 Quantum Merlin Arthur (QMA)

Just as PromiseBPP was the correct class to generalize to BQP, to define QMA we begin with the promise-problem probabilistic generalization of NP, PromiseMA.

Definition 5.1 (PromiseMA)**.** *A promise problem $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in* PromiseMA *if there exists a (deterministic) TM M and fixed polynomials $p, s, r : \mathbb{N} \to \mathbb{R}^+$, such that for any input $x \in \{0,1\}^n$, M takes in "proof" $y \in \{0,1\}^{p(n)}$ and string $z \in \{0,1\}^{s(n)}$, halts in at most $O(r(n))$ steps, and:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, there exists a proof $y \in \{0,1\}^{p(n)}$, such that for at least $2/3$ of the choices of $z \in \{0,1\}^{s(n)}$, M accepts.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then for all proofs $y \in \{0,1\}^{p(n)}$, at most $1/3$ of the choices of $z \in \{0,1\}^{s(n)}$ cause M to accept.*
- *(Invalid case) If $x \in A_{\text{inv}}$, then M may accept or reject arbitrarily.*

Exercise 5.2. How might we define Merlin-Arthur (MA), instead of PromiseMA (i.e. how does the definition above change if we drop the promise)?

Exercise 5.3. Show that the completeness and soundness parameters of 2/3 and 1/3, respectively, can be amplified without loss of generality to $1 - 2^{-n}$ and 2^{-n}, respectively. How many copies of the proof y suffice for this amplification?

The quantum analogue of PromiseMA which we focus on in this lecture is Quantum Merlin Arthur (QMA) (which, again, is really PromiseQMA, just as BQP is really PromiseBQP).

Definition 5.4 (Quantum Merlin Arthur (QMA))**.** *A promise problem $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in QMA if there exists a P-uniform quantum circuit family $\{Q_n\}$ and polynomials $p, q : \mathbb{N} \to \mathbb{N}$ satisfying the following properties. For any input $x \in \{0,1\}^n$, Q_n takes in $n + p(n) + q(n)$ qubits as input, consisting of the input x on register A, $p(n)$ qubits initialized to a "quantum proof" $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$ on register B, and $q(n)$ ancilla qubits initialized to $|0\rangle$ on register C. The first qubit of register C, denoted C_1, is the designated output qubit, a measurement of which in the standard basis after applying Q_n yields the following:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, there exists proof $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$, such that Q_n accepts with probability at least $2/3$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then for all proofs $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$, Q_n accepts with probability at most $1/3$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, Q_n may accept or reject arbitrarily.*

Exercise 5.5. If we replace the quantum proof $|\psi\rangle$ with a classical proof $y \in \{0,1\}^{p(n)}$ in the definition of QMA, do we recover PromiseMA?

Weak error reduction. Similar to PromiseMA, parallel repetition suffices to amplify the QMA completeness and soundness parameters to $1 - 2^{-n}$ and 2^{-n}, respectively. However, the proof of this fact is not entirely trivial.

Exercise 5.6. Suppose the QMA prover sends k copies of its proof, $|\psi\rangle$, instead of a single copy. On the jth copy of the proof, the verifier runs the verification circuit Q_n. Finally, the verifier measures the output qubits of all runs of Q_n, takes a majority vote of the resulting bits, and accepts if and only if the majority function yields 1. Prove that this procedure indeed amplifies the completeness and soundness parameters for QMA. (Hint: In the NO case, a cheating prover is *not* obligated to send k copies of some state $|\psi\rangle$ in tensor product, but rather can cheat by sending a large entangled state $|\phi\rangle \in (\mathbb{C}^2)^{\otimes k \cdot p(n)}$ across all k proof registers. Why does entanglement across proofs not help the prover in the NO case?)

Observe we have denoted the use of parallel repetition above as *weak* error reduction. This is because the amplification step blows up the size of the proof register. Naively, one may expect this blowup to be necessary, since *a priori* it seems we cannot "reuse" the quantum proof $|\psi\rangle$ — indeed, the QMA verifier's measurement of its output qubit disturbs its quantum state, and the no-cloning theorem says the verifier cannot sidestep this by simply copying its input proof $|\psi\rangle$ before verifying it. Nevertheless, it turns out that amplification without a blowup in proof size *is* possible — this is called *strong* error reduction (Section 5.2), a simple and elegant application of which is to show that QMA $\subseteq$ PP (Section 5.3).

Pure versus mixed proofs. We have assumed the proof $|\psi\rangle$ in QMA to be a *pure* state, as opposed to a mixed state. Let us now reformulate the optimal acceptance probability of the quantum verifier Q_n as an eigenvalue problem; along the way, we will not only see that the "pure state proof" assumption is without loss of generality, but the reformulation we derive will prove crucial in our analysis of Section 5.2.

Let Q_n be the circuit from Definition 5.4, acting on $n+p(n)+q(n)$ qubits. Recall that A, B, C denote the input, proof, and ancilla registers, respectively, and C_1 the designated output qubit of Q_n. Then, the probability that Q_n accepts proof $|\psi\rangle$ is

$$\begin{aligned}
\Pr[\text{accept}] &= \left\| |1\rangle\langle 1|_{C_1} Q_n |x\rangle_A \otimes |\psi\rangle_B \otimes |0\cdots 0\rangle_C \right\|_2^2 \\
&= \langle x|_A \otimes \langle\psi|_B \otimes \langle 0\cdots 0|_C Q_n^\dagger |1\rangle\langle 1|_{C_1} Q_n |x\rangle_A \otimes |\psi\rangle_B \otimes |0\cdots 0\rangle_C \\
&= \mathrm{Tr}\left[\left(\langle x|_A \otimes I_B \otimes \langle 0\cdots 0|_C Q_n^\dagger |1\rangle\langle 1|_{C_1} Q_n |x\rangle_A \otimes I_B \otimes |0\cdots 0\rangle_C \right) |\psi\rangle\langle\psi|_B \right] \\
&= \mathrm{Tr}(P_x |\psi\rangle\langle\psi|), \qquad (5.1)
\end{aligned}$$

where the third statement follows by cyclicity of the trace, the fourth by defining for convenience

$$P_x := \langle x|_A \otimes I_B \otimes \langle 0\cdots 0|_C Q_n^\dagger |1\rangle\langle 1|_{C_1} Q_n |x\rangle_A \otimes I_B \otimes |0\cdots 0\rangle_C. \qquad (5.2)$$

Henceforth, we shall abuse terminology by refering to P_x as the *POVM*[1] *for verifier* Q_n.

Exercise 5.7. What space does P_x act on?

Exercise 5.8. Prove $P_x \succeq 0$. (Hint: Prove first that if $A \succeq 0$, then $BAB^\dagger \succeq 0$ for any (possibly non-square) matrix B; the proof will be easier if you choose the "right" definition of positive semi-definiteness to work with.

Now we are ready to address the question: *What happens if we consider a mixed proof* ρ *in place of a pure state* $|\psi\rangle\langle\psi|$*?*

Exercise 5.9. Prove that for any density operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, there exists an i such that $\mathrm{Tr}(P_x\rho) \leq \mathrm{Tr}(P_x|\psi_i\rangle\langle\psi_i|)$. Conclude that, without loss of generality, quantum proofs in Definition 5.4 can be pure states.

Finally, we stated earlier that the optimal acceptance probability can be reformulated as an eigenvalue problem. Indeed, the optimal acceptance probability over all proofs $|\psi\rangle$ is now

$$\max_{\text{unit vectors } |\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}} \langle\psi|P_x|\psi\rangle = \lambda_{\max}(P_x), \qquad (5.3)$$

attained by any eigenvector $|\psi\rangle$ of P_x with eigenvalue $\lambda_{\max}(P_x)$.

[1] Formally, POVM stands for Positive-Operator Valued Measure, and it denotes an alternate approach for modelling measurements. Specifically, a POVM P acting on n qubits is a set of operators $P = \{P_1, \ldots, P_k\}$ for some $k > 0$, such that $P_i \succeq 0$ and $\sum_i P_i = I$. As with projective measurements, each i denotes a distinct outcome of the measurement encoded by P, and the probability of outcome i is $\mathrm{Tr}(P_i\rho)$ when measuring state ρ. Unlike projective measurements, we do *not* require that $P_iP_j = 0$ for $i \neq j$; in this sense, POVMs generalize projective measurements. (Note that also unlike projective measurements, the postmeasurement state upon obtaining outcome i is no longer specified by $P_i\rho P_i/\mathrm{Tr}(\rho P_i)$.) In the context of a QMA verifier Q_n, we may view the application of Q_n and subsequent measurement of the output qubit of Q_n in the standard basis as a POVM consisting of two elements: $P = \{I - P_x, P_x\}$ (since the measurement has only two outputs, $|0\rangle$ or $|1\rangle$, respectively). Since this two-outcome POVM P is fully specified by P_x, for simplicity we choose to abuse terminology and refer to P by P_x.

Exercise 5.10. Prove the equality above using the Courant-Fischer variational characterization of eigenvalues. The latter states: Let $A \in \mathrm{Herm}(\mathbb{C}^N)$ have eigenvalues $\lambda_1 \leq \cdots \leq \lambda_N$. Then,

$$\lambda_k = \min_{\text{subspaces } S \subseteq \mathbb{C}^N \text{ of dimension } k} \quad \max_{\text{unit vectors } |\psi\rangle \in S} \langle\psi|A|\psi\rangle. \tag{5.4}$$

5.2 Strong error reduction for QMA

In the setting of PromiseMA, a previous exercise essentially asked you to show that at given a single copy of proof y, the completeness and soundness parameters of the PromiseMA verifier could be amplified to exponentially close to 1 and 0, respectively. Quantumly, one might naively expect an analogous statement to be false — a quantum verifier measures and hence disturbs its state, so how can it "reuse" its proof, $|\psi\rangle$? A simple solution would be for the quantum verifier to create multiple copies of $|\psi\rangle$ before beginning its verification; unfortunately, the quantum no-cloning theorem rules this out. The following theorem hence comes as a surprise.

Theorem 5.11 (Strong error reduction for QMA)**.** *Let Q_n be a QMA verifier for promise problem $A = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$, where we assume the terminology of Definition 5.4. Then, for any polynomial $r : \mathbb{N} \to \mathbb{N}$, there exists a polynomial-time deterministic TM mapping Q_n to a (polynomial-size) quantum circuit R_n with the following properties for any input $x \in \{0,1\}^n$:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, there exists proof $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$, such that R_n accepts with probability at least $1 - 2^{-r(n)}$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then for all proofs $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$, R_n accepts with probability at most $2^{-r(n)}$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, R_n may accept or reject arbitrarily.*

It is crucial to note that both Q_n and R_n take in the same number of proof qubits, $p(n)$.

Remark. *As with weak error reduction, Theorem 5.11 holds even if the completeness parameter $c(n)$ and soundness parameter $s(n)$ for Q_n satisfy $c(n) - s(n) \geq 1/t(n)$ for some fixed polynomial $t : \mathbb{N} \to \mathbb{N}$.*

5.2.1 Intuition: A spinning top

To most easily see the intuition behind the proof of Theorem 5.11, assume Q_n has completeness 1, i.e. in the YES case, there exists a proof $|\psi\rangle$ accepted by Q_n with certainty. Assume first that $x \in A_{\text{yes}}$. There are two high-level steps to the amplification procedure:

1. *Run the verification.* Apply Q_n to obtain

$$|\phi\rangle = Q_n|x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C \in (\mathbb{C}^2)^{\otimes n+p(n)+q(n)}. \tag{5.5}$$

   Since $x \in A_{\text{yes}}$ and Q_n has perfect completeness, we know

$$|\phi\rangle = |1\rangle_{C_1}|\phi'\rangle \text{ for some unit vector } |\phi'\rangle \in (\mathbb{C}^2)^{\otimes n+p(n)+q(n)-1}.$$

   Thus, if we measure the output qubit, C_1, in the standard basis via projectors $\Pi^{\text{accept}} = |1\rangle\langle 1|$ and $\Pi^{\text{reject}} = |0\rangle\langle 0| \in \mathcal{L}(\mathbb{C}^2)$, we not only obtain outcome Π^{accept} with certainty, but the postmeasurement state is $\Pi^{\text{accept}}_{C_1}|\phi\rangle = |\phi\rangle$. In other words, the measurement *does not disturb* the output of Q_n.

2. *Run the verification in reverse.* Since measuring $|\phi\rangle$ did not disturb it, applying Q_n in reverse now trivially reverts us to our initial state:

$$Q_n^\dagger \left(\Pi_{C_1}^{\text{accept}}|\phi\rangle\right) = Q_n^\dagger|\phi\rangle = Q_n^\dagger \left(Q_n|x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C\right) = |x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C.$$

If we now measure projectors[2] $\Pi^{\text{reset}} = |x\rangle\langle x|_A \otimes |0\cdots 0\rangle\langle 0\cdots 0|_{A,C}, \Pi^{\text{new}} = I - \Pi^{\text{reset}} \in \mathcal{L}((\mathbb{C}^2)^{\otimes n+q(n)})$, we again leave the state invariant, i.e. with probability 1 we obtain outcome Π^{reset}, and the postmeasurement state satisfies

$$\Pi^{\text{reset}}|x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C = |x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C.$$

Note that we may repeat this procedure as many times as we like, and the outcomes will always be the same. This is akin to a perfectly spinning top — if we think of the each application of the amplification procedure as giving the top a supplemental twirl, the top will continue to spin blissfully along in a "steady state".

The interesting part is now the NO case. Here, in Step 1 of the amplification procedure above, since proof $|\psi\rangle$ is accepted with probability at most 1/3, we know $|\phi\rangle$ has form

$$|\phi\rangle = \alpha_0|0\rangle_{C_1}|(\phi')^\perp\rangle + \alpha_1|1\rangle_{C_1}|\phi'\rangle$$

for some orthonormal unit vectors $|\phi'\rangle, |(\phi')^\perp\rangle \in (\mathbb{C}^2)^{\otimes n+p(n)+q(n)-1}$, $|\alpha_0|^2 + |\alpha_1|^2 = 1$, and $|\alpha_1|^2 \leq 1/3$. The last of these properties guarantees that if we are lucky enough to measure $|1\rangle$ in C_1, the postmeasurement collapse will disturb $|\phi\rangle$ greatly (since most of the weight of $|\phi\rangle$ is on $|0\rangle_{C_1}$). This, in turn, suggests that when we now run Step 2 by inverting Q_n and measuring $\left\{\Pi^{\text{reset}}, \Pi^{\text{new}}\right\}$, we will obtain outcome Π^{new} with non-trivial probability, and again disturb our state greatly. And applying these two steps repeatedly will presumably amplify the disturbances further. This is analogous to saying that if we start with a top spinning with a slight wobble, each twirl we perform will further amplify the wobble until the top spins out of control.

5.2.2 Proof of strong error reduction

While Section 5.2.1 gave intuition as to why the amplification procedure might work, a formal analysis reveals the "motion of our top" can be tracked in a very elegant and precise fashion, even if we drop the assumption of perfect completeness.

Proof of Theorem 5.11. We begin by following Section 5.2.1. Let Q_n be a QMA verifier and $x \in \{0,1\}^n$ an input string. For brevity, we henceforth simply write Q for Q_n. To ease the analysis, we also rename our projectors

$$S_0 := \Pi^{\text{new}}, \qquad S_1 := \Pi^{\text{reset}}, \qquad E_0 := \Pi^{\text{reject}}, \qquad E_1 := \Pi^{\text{accept}}, \tag{5.6}$$

where S in S_i stands for "start" (since this measurement is on the start state) and E_i stands for "end" (since this measurement is on the end state).

[2]For clarity, the superscript for Π^{reset} means the A and C registers are *reset* to their original states $|x\rangle$ and $|0\cdots 0\rangle$, respectively, and for Π^{new} means the registers are set to a "new" start state.

The new verification procedure. The new circuit R_n (henceforth R for brevity) acts as follows.

1. Set $i = t = 0$.
2. While $i \leq N$:
   a) (Run the verification) Apply Q and measure output qubit C_1 with respect to $\{E_0, E_1\}$. If the outcome is E_j, set $y_i = j \in \{0, 1\}$, and increment i.
   b) (Run the verification in reverse) Apply $Q^\dagger$ and measure input and ancilla registers A and C with respect to $\{S_0, S_1\}$. If the outcome is S_j, set $y_i = j \in \{0, 1\}$, and increment i.
3. (Postprocessing) If the number of indices $i \in \{0, \ldots, N-1\}$ such that $y_i = y_{i+1}$ is at least $N/2$, accept. Otherwise, reject.

It suffices to set $N = 8r(n)/9$. Note the mapping from Q to V takes time polynomial in n.

Exercise 5.12. How many times does the while loop above run with respect to N?

Correctness. If we are in a YES case with perfect completeness, it is clear in Step 3 above that $y_i = y_{i+1}$ for all $i \in \{0, \ldots, N-1\}$. The aim is thus to show a similar statement for general YES (resp. NO) cases; that we are *more likely* to maintain (resp. flip) the value of y_i in setting y_{i+1} in the YES (resp. NO) case, thus leading to the correct answer with high probability in Step 3.

The starting point for the formal analysis is Equation (5.1), which said the probability that Q accepts proof $|\psi\rangle$ is $\mathrm{Tr}(P_x|\psi\rangle\langle\psi|)$, for positive-semidefinite operator

$$P_x := \langle x|_A \otimes I_B \otimes \langle 0 \cdots 0|_C Q^\dagger E_1 Q |x\rangle_A \otimes I_B \otimes |0 \cdots 0\rangle_C.$$

It turns out that if we restrict our attention to eigenvectors $|\psi\rangle$ of P_x, we obtain a clean closed form solution.

Closed form solution when $|\psi\rangle$ is an eigenvector of P_x. In the case when $|\psi\rangle$ is an eigenvector of P_x, we can *exactly* write down the acceptance probability of $|\psi\rangle$ by R, and this will turn out to suffice for the entire correctness analysis.

Lemma 5.13. *Suppose $|\psi\rangle$ is an eigenvector of P_x accepted by Q with probability p. Then, for any $i \in \{0, \ldots, N-1\}$, $\Pr[y_i = y_{i+1}] = p$. (Thus, $\Pr[y_i \neq y_{i+1}] = 1 - p$.)*

The magic of Lemma 5.13 is that even though *a priori* the action of R on $|\psi\rangle$ seems difficult to predict, when $|\psi\rangle$ is an eigenvector of P_x, each step 2(a) and 2(b) of R is just a *Bernoulli trial*[3]: Independently of all previous measurement outcomes, with probability p we don't flip our bit, and with probability $1 - p$ we do flip our bit. This means we can later apply powerful tail bounds like the Chernoff bound to analyze the acceptance probability of R on eigenvectors of $|\psi\rangle$.

Proof of Lemma 5.13. Assume $P_x|\psi\rangle = p|\psi\rangle$ for $0 < p < 1$.

[3]Recall from probability theory that *Bernoulli trials* refer to independently repeating a two-outcome sampling experiment.

Exercise 5.14. Prove the claim in the setting $p = 0$ and $p = 1$.

Recall from Equation (5.6) that S_1 and E_1 denote successful projections at the start and end of verification (i.e. onto the original input x and all-zeroes ancilla, and onto the accepting output qubit, respectively), and $S_0 = I - S_1$ and $E_0 = I - E_1$. Define for brevity

$$|\phi\rangle := |x\rangle_A|\psi\rangle_B|0\cdots 0\rangle_C \qquad \text{and} \qquad \Gamma := S_1 Q^\dagger E_1 Q S_1.$$

A key identity is now the following.

Exercise 5.15. Prove that

$$\Gamma|\phi\rangle = S_1 Q^\dagger E_1 Q S_1|\phi\rangle = p|\phi\rangle. \tag{5.7}$$

Why must we include projectors S_1 in the definition of Γ to make this a well-defined equality?

To show the claim, we trace through the first iteration of the while loop of R.

- The first run of Step 2(a) applies Q to $|\psi\rangle$. Since $E_0 + E_1 = I$, this step hence performs mapping

  $$|\phi\rangle \to Q|\phi\rangle = E_0 Q|\phi\rangle + E_1 Q|\phi\rangle.$$

  If we now measure $\{E_0, E_1\}$, we collapse to state

  $$|e_1\rangle := \frac{E_1 Q|\phi\rangle}{\| E_1 Q|\phi\rangle \|_2} \text{ with probability } \| E_1 Q|\phi\rangle \|_2^2 = \langle\phi|Q^\dagger E_1 Q|\phi\rangle = p.$$

  Exercise 5.16. Why is $\langle\phi|Q^\dagger E_1 Q|\phi\rangle = p$? (Hint: What is $S_1|\phi\rangle$?)

  Using the identity $E_0 = I - E_1$, we analogously collapse to

  $$|e_0\rangle := \frac{E_0 Q|\phi\rangle}{\| E_0 Q|\phi\rangle \|_2} \text{ with probability } \| E_0 Q|\phi\rangle \|_2^2 = \langle\phi|Q^\dagger E_0 Q|\phi\rangle = 1 - \langle\phi|Q^\dagger E_1 Q|\phi\rangle = 1 - p.$$

  Note that together, these statements imply

  $$Q|\phi\rangle = \sqrt{1-p}|e_0\rangle + \sqrt{p}|e_1\rangle. \tag{5.8}$$

- After Step 2(a), we have either $|e_0\rangle$ or $|e_1\rangle$. Step 2(b) now applies $Q^\dagger$, yielding one of two possible transitions:

  $$Q^\dagger|e_1\rangle = S_0 Q^\dagger|e_1\rangle + S_1 Q^\dagger|e_1\rangle \tag{5.9}$$
  $$Q^\dagger|e_0\rangle = S_0 Q^\dagger|e_0\rangle + S_1 Q^\dagger|e_0\rangle \tag{5.10}$$

  We may simplify each term on the right hand side as:

  $$S_1 Q^\dagger|e_1\rangle = S_1 Q^\dagger \frac{E_1 Q|\phi\rangle}{\sqrt{p}} = \frac{1}{\sqrt{p}}\Gamma|\phi\rangle = \sqrt{p}|\phi\rangle \tag{5.11}$$
  $$S_0 Q^\dagger|e_1\rangle = S_0 Q^\dagger \frac{E_1 Q|\phi\rangle}{\sqrt{p}} = \frac{1}{\sqrt{p}} S_0 Q^\dagger E_1 Q|\phi\rangle \tag{5.12}$$
  $$S_1 Q^\dagger|e_0\rangle = S_1 Q^\dagger \frac{E_0 Q|\phi\rangle}{\sqrt{1-p}} = \frac{1}{\sqrt{1-p}}\left(|\phi\rangle - \Gamma|\phi\rangle\right) = \sqrt{1-p}|\phi\rangle \tag{5.13}$$
  $$S_0 Q^\dagger|e_0\rangle = S_0 Q^\dagger \frac{E_0 Q|\phi\rangle}{\sqrt{1-p}} = \frac{1}{\sqrt{1-p}}\Big(S_0|\phi\rangle - S_0 Q^\dagger E_1 Q|\phi\rangle\Big) = -\frac{1}{\sqrt{1-p}} S_0 Q^\dagger E_1 Q|\phi\rangle \tag{5.14}$$

Exercise 5.17. Prove each of the four statements above. (Hint: Use the fact that $E_0 + E_1 = I$. Also, why is $S_0|\phi\rangle = 0$?)

Exercise 5.18. What is $\left\| S_0 Q^\dagger E_1 Q|\phi\rangle \right\|_2$?

Exercise 5.19. Prove that after we measure the right hand side of Equation (5.9) with $\{S_0, S_1\}$, we obtain S_1 with probability p and S_0 with probability $1-p$. Similarly, measuring the right hand side of Equation (5.10) with S_0, S_1 yields S_0 with probability p and S_1 with probability $1-p$.

Exercise 5.20. Conclude from the last exercise that after the first iteration of the while loop, $y_1 = y_2$ with probability p. Accordingly, $y_1 \neq y_2$ with probability $1-p$.

This is precisely the behavior we are seeking. In sum, the analysis of Step 2(b) yields the following.

Exercise 5.21. Define $|s_0\rangle := \frac{S_0 Q^\dagger E_1 Q|\phi\rangle}{\left\| S_0 Q^\dagger E_1 Q|\phi\rangle \right\|_2}$ and $|s_1\rangle := |\phi\rangle$. Prove the following:

$$\begin{aligned} Q^\dagger|e_0\rangle &= -\sqrt{p}|s_0\rangle + \sqrt{1-p}|s_1\rangle \\ Q^\dagger|e_1\rangle &= \sqrt{1-p}|s_0\rangle + \sqrt{p}|s_1\rangle. \end{aligned}$$

Exercise 5.22. Prove that $Q|s_0\rangle = -\sqrt{p}|e_0\rangle + \sqrt{1-p}|e_1\rangle$. (Hint: Use Equation (5.7).)

It will now be fruitful to step back and see the bigger picture which is emerging. Define $S_v := \text{Span}(|e_0\rangle, |e_1\rangle)$ and $S_w := \text{Span}(|s_0\rangle, |s_1\rangle)$. Then, our analysis above showed that Q maps S_w into S_v. Conversely, $Q^\dagger$ maps S_v back into S_w.

Exercise 5.23. Prove that $\{|e_0\rangle, |e_1\rangle\}$ is an orthonormal set.

Exercise 5.24. Prove that $\{|s_0\rangle, |s_1\rangle\}$ is an orthonormal set.

In other words, the evolution of R is *entirely confined* in a two-dimensional spaces S_v and S_w. With respect to these spaces, our analysis reveals the entire action of Q:

$$\begin{aligned} Q|s_0\rangle &= -\sqrt{p}|e_0\rangle + \sqrt{1-p}|e_1\rangle \\ Q|s_1\rangle &= \sqrt{1-p}|e_0\rangle + \sqrt{p}|e_1\rangle \\ Q^\dagger|e_0\rangle &= -\sqrt{p}|s_0\rangle + \sqrt{1-p}|s_1\rangle \\ Q^\dagger|e_1\rangle &= \sqrt{1-p}|s_0\rangle + \sqrt{p}|s_1\rangle. \end{aligned}$$

We are now ready to finish the proof of Lemma 5.13.

Exercise 5.25. Observe that $S_i|s_i\rangle = |s_i\rangle$, and $S_i|s_{i\oplus 1}\rangle = 0$. Similarly, $E_i|e_i\rangle = |e_i\rangle$, and $E_i|e_{i\oplus 1}\rangle = 0$. Why can we now conclude the analysis for a single loop iteration suffices to prove all of Lemma 5.13? □

With Lemma 5.13 in hand, we have a clean characterization of how R behaves on any proof $|\psi\rangle$ which is an eigenvector of P_x. We are now ready to complete the proof of Theorem 5.11.

Correctness proof for YES case. In the YES case, from Section 5.1 we know that the optimal proof for Q is an eigenvector $|\psi\rangle$ of P_x accepted with probability $p \geq 2/3$. Thus, by Lemma 5.13, for each $i \in \{0, \ldots, N-1\}$, $y_i = y_{i+1}$ with probability at least $2/3$. By the Chernoff bound (which we may apply since Lemma 5.13 reduces us to Bernoulli trials), the claim now follows.

Correctness proof for NO case. In the NO case, the optimal proof for Q is an eigenvector $|\psi\rangle$ of P_x accepted with probability $p \leq 1/3$. Unfortunately, here we cannot proceed as in the YES case by assuming a cheating prover sends an eigenvector of P_x as a proof. Luckily, it turns out that by applying a similar, but slightly more general, analysis to that above, one can explicitly show the desired bound for the NO case as well. We omit this additional analysis.

□

5.3 Relationship to other classes

5.3.1 The many cousins of QMA

There are a number of variants of QMA, the most prominent of which are arguably the following (in no particular order).

- **One-Sided Error Quantum Merlin Arthur ($\mathbf{QMA_1}$).** QMA with perfect completeness, i.e. in the YES case there exists a proof accepted with probability 1.
- **Quantum Classical Merlin Arthur (QCMA).** QMA, but with a classical proof $y \in \{0,1\}^{p(n)}$.
- **Stoquastic Merlin Arthur (StoqMA).** QMA, except (1) the ancilla qubits are allowed to be initialized (independently) to either $|0\rangle$ or $|+\rangle$, (2) the verification circuit consists solely of reversible classical gates, and (3) the final single-qubit measurement is in the X basis (i.e. $\{|+\rangle, |-\rangle\}$.
- **Quantum Merlin Arthur with Two Proofs (QMA(2))** QMA, except the proof is promised to be a tensor product of two proofs, i.e. $|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle$ for some $|\phi_1\rangle, |\phi_2\rangle$.

Again, despite the nomenclature, each of these is a class of promise problems, not a class of languages. Here is what is known about the relationships between these:

- $\mathrm{BQP} \subseteq \mathrm{QCMA} \subseteq \mathrm{QMA}_1 \subseteq \mathrm{QMA} \subseteq \mathrm{QMA}(2) \subseteq \mathrm{NEXP}$.

  Exercise 5.26. Which of these inclusions are trivial? Why?

Exercise 5.27. Why is it not clear that QMA = QMA(2)?

Two remarks: (1) The inclusion QCMA $\subseteq$ QMA_1 follows because one can show QCMA = QCMA_1, i.e. without loss of generality we may assume QCMA has perfect completeness. The analogous statement is *not* known for QMA. (2) The containment QMA(2) $\subseteq$ NEXP is, remarkably and sadly, the best trivial upper bound on QMA(2). This leaves quite a chasm between QMA and QMA(2), with the former contained in[4] PP. The only known *non-trivial*[5] upper bound on QMA(2) is

$$\mathrm{QMA}(2) \subseteq \mathrm{Q}\Sigma_3 \subseteq \mathrm{NEXP},$$

where $\mathrm{Q}\Sigma_3$ is a quantum analogue[6] of Σ_3^{p}, the third level of PH. It is not yet clear if this should be construed as strong evidence that QMA(2) $\neq$ NEXP; not much is known about $\mathrm{Q}\Sigma_3$, and it is entirely possible that QMA(2) = $\mathrm{Q}\Sigma_3$ = NEXP. On the other hand, if the study of the classical analogue of $\mathrm{Q}\Sigma_3$ is any guide, it would suggest QMA(2) $\neq$ $\mathrm{Q}\Sigma_3$ (and hence QMA(2) $\neq$ NEXP), as classically alternating quantifiers are strongly believed to add power to a proof system (otherwise, PH collapses).

Exercise 5.28. Why would QMA(2) = NEXP imply that alternating quantifiers do not strictly increase the power of a QMA(2) proof system?

- As for StoqMA, it is a rather strange fellow — in terms of lower bounds, it is the only "quantum" cousin of QMA which is *not* believed to contain BQP. Indeed, StoqMA $\subseteq$ PH (more precisely, it is in Arthur-Merlin (AM)), whereas it is believed BQP is not contained in PH. In terms of upper bounds, it is not clear whether StoqMA $\subseteq$ QCMA; this because the former has a quantum proof but "classical" verification, whereas the latter has a classical proof but quantum verification. In fact, it is not even known whether weak error reduction holds[7] for StoqMA, since the final X-basis measurement appears to prevent the standard "parallel repetition plus majority-vote" technique.

Upper bounds on QMA. The most "mainstream" upper bound on QMA is QMA $\subseteq$ PP. However, there are two strictly stronger known bounds (assuming standard complexity theoretic conjectures):

1. QMA $\subseteq$ $\mathrm{A_0PP}$ $\subseteq$ PP. Rather than defining $\mathrm{A_0PP}$, we will define the *quantum* class SBQP = $\mathrm{A_0PP}$. SBQP is the class of decision problems for which there exists a quantum polynomial time algorithm which, on input $x \in \{0,1\}^*$, accepts in the YES case with probability at least $2 \cdot 2^{-p(|x|)}$, and accepts in the NO case with probability at most $2^{-p(|x|)}$, for some polynomial p. Note that it is strongly believed that SBQP = $\mathrm{A_0PP} \neq$ PP, since equality would imply PH $\subseteq$ PP.

[4] Recall PP $\subseteq$ PSPACE $\subseteq$ EXP $\subseteq$ NEXP.

[5] That QMA(2) $\subseteq$ $\mathrm{Q}\Sigma_3$ is trivial; it is the containment $\mathrm{Q}\Sigma_3 \subseteq$ NEXP which is non-trivial.

[6] Roughly, in a YES instance for $\mathrm{Q}\Sigma_3$, there is a proof ρ_1, such that for all proofs ρ_2, there exists a ρ_3 leading the quantum verifier to accept (ρ_1, ρ_2, ρ_3) with probability at least 2/3. Analogously for a NO instance, for all proofs ρ_1, there is a proof ρ_2, such that for all ρ_3 the quantum verifier accepts with probability at most 1/3. All proofs are polynomial-size and allowed to be mixed. In strong contrast to QMA, it is *not* clear whether the proofs can be assumed pure without loss of generality.

[7] Very recently, it was shown that if one could reduce the error bounds for StoqMA to $1 - o(1/\operatorname{poly}(n))$ versus $O(1)$, then StoqMA = MA. (Note the "little-oh" here — the result does not apply to inverse polynomial error reduction.) Whether this should be seen as evidence that error reduction for StoqMA is impossible, or that StoqMA = MA, is yet to be seen.

2. $\mathrm{QMA} \subseteq \mathrm{P}^{\mathrm{QMA[log]}} \subseteq \mathrm{PP}$. Here, $\mathrm{P}^{\mathrm{QMA[log]}}$ is the set of decision problems solved by a P machine which can make at most $O(\log n)$ (adaptive) queries to a QMA oracle. Again, this strictly separates QMA from PP, in the sense that it is unlikely that $\mathrm{QMA} = \mathrm{P}^{\mathrm{QMA[log]}}$. This is because the latter contains both QMA and co-QMA (the complement of QMA), and so $\mathrm{QMA} = \mathrm{P}^{\mathrm{QMA[log]}}$ would have the unlikely implication that $\mathrm{QMA} \supseteq \text{co-QMA}$.

Exercise 5.29. Why does $\text{co-QMA} \subseteq \mathrm{P}^{\mathrm{QMA[log]}}$ hold?

While these two upper bounds on QMA are likely stronger than PP, we now close the lecture by showing the weaker bound $\mathrm{QMA} \subseteq \mathrm{PP}$; this latter containment follows via a simple application of strong error reduction.

5.3.2 Using strong error reduction to show QMA ⊆ PP

Theorem 5.30. $\mathrm{QMA} \subseteq \mathrm{PP}$.

Proof idea. The proof idea is most easily grasped by using it to show $\mathrm{NP} \subseteq \mathrm{PP}$. For this, suppose we have a 3-SAT input formula $\phi : \{0,1\}^n \to \{0,1\}$. To put NP in PP, our goal is to show that there exists a probabilistic polynomial-time algorithm A which, given ϕ, accepts with probability strictly larger than $1/2$ if ϕ is satisfiable, and accepts with probability at most $1/2$ otherwise. The approach for doing so is simple — if and only if ϕ is satisfiable, it has a satisfying assignment x; so, A randomly picks an assignment $y \in \{0,1\}^n$, and outputs $\phi(y)$.

Exercise 5.31. Prove that if ϕ is satisfiable, A accepts with probability at least $1/2^n$. On the other hand, if ϕ is unsatisfiable, A accepts with probability 0. Why is this enough to imply $\text{3-SAT} \in \mathrm{PP}$?

Proof of Theorem 5.30. We shall show $\mathrm{QMA} \subseteq \mathrm{PQP}$, for PQP defined essentially identically to PP except with a P-uniform quantum circuit family in place of a Turing machine. It is known that $\mathrm{PQP} = \mathrm{PP}$, whose proof we omit here.

Let $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ be a QMA promise problem, and $x \in \{0,1\}^n$ an input. The overall proof idea is the same as in the classical case — the PQP machine A simply "guesses" a quantum proof $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$, feeds it into verifier Q_n, and outputs Q_n's answer. Formally, to model a "random proof" $|\psi\rangle$, A instead feeds Q_n the maximally mixed state $I/2^{p(n)} \in \mathcal{L}((\mathbb{C}^2)^{\otimes p(n)})$.

Exercise 5.32. Why does $\frac{1}{2^{p(n)}} I$ correctly model a random pure state $|\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$?

Recall we may now write the acceptance probability of Q_n on proof $I/2^{p(n)}$ as (for POVM P_x defined as in Equation (5.1))

$$\Pr[\text{accept}] = \mathrm{Tr}\left(P_x \cdot \frac{I}{2^{p(n)}}\right) = \frac{1}{2^{p(n)}} \mathrm{Tr}(P_x).$$

Exercise 5.33. Recall that $P_x \succeq 0$, and that $\mathrm{Tr}(P_x)$ is the sum of all eigenvalues of P_x. Why does the Pr[accept] above not suffice to separate YES from NO cases of $\mathcal{A}$?

As the exercise above shows, this naive idea alone does not work. Rather, we must first use strong error reduction to amplify the completeness and soundness parameters of Q_n. Specifically, recall that Q_n takes in $p(n)$ proof qubits, for some polynomial p. For any polynomial r, Theorem 5.11 says we may map Q_n to a new circuit R_n which still takes in $p(n)$ proof qubits, but has completeness and soundness parameters $1 - 2^{-r(n)}$ and $2^{-r(n)}$, respectively. This now suffices to complete the proof.

Exercise 5.34. Prove that for sufficiently large fixed r, feeding $I/2^n$ into R_n and outputting its answer suffices to decide in PQP whether $x \in A_{\text{yes}}$ or $x \in A_{\text{no}}$. More formally, let P_x^R denote the POVM for verifier R_n (c.f. Equation (5.1)). Prove that:

- If $x \in A_{\text{yes}}$, then $\frac{1}{2^{p(n)}}\text{Tr}(P_x^R) \geq \frac{1}{2^{p(n)}} - \frac{1}{2^{p(n)+r(n)}}$.
- If $x \in A_{\text{no}}$, then $\frac{1}{2^{p(n)}}\text{Tr}(P_x^R) \leq \frac{1}{2^{r(n)}}$.

What choice of r hence suffices to distinguish YES from NO cases in PQP? (Hint: You do not need to use the precise structure of P_x^R; the relationship between the optimal probability of acceptance of R_n and the eigenvalues of P_x^R suffices.)

Exercise 5.35. Would the approach above work if we used weak error reduction instead of strong error reduction? Why or why not? □

6 The quantum Cook-Levin theorem

> *"Steve Cook was primarily in math but also in the new CS Department. It is to our everlasting shame that we were unable to persuade the math department to give him tenure. Perhaps they would have done so if he had published his proof of the NP-completeness of satisfiability a little earlier."*
> — Richard Karp, speaking about UC Berkeley Computer Science in the late 1960's

Introduction. Among the many advances of theoretical computer science during the 20th century, three are unquestionably among the crown jewels of the field: First, that there exist *unsolvable* computational problems (Turing's proof that the Halting Problem is undecidable, with Gödel's incompleteness theorem acting as an important precursor). Second, even among solvable problems, not all of them can be solved *efficiently* (the Cook-Levin theorem, which spawned the theory of NP-completeness and arguably founded the field of complexity theory). Third, some problems are not only hard to solve exactly, but are even hard to solve *approximately* (the PCP theorem of the 1990's).

In this lecture, we focus on the Cook-Levin theorem, and in particular its quantum analogue. The latter roughly says that the quantum analogue of Boolean Constraint Satisfaction, known as the *Local Hamiltonian problem*, is QMA-complete. Thus, just as the Cook-Levin theorem says that (assuming P $\neq$ NP) 3-SAT cannot be solved in polynomial time, the *quantum* Cook-Levin theorem implies the Local Hamiltonian problem has no efficient classical or quantum solution.

This statement marks a striking departure from the realm of computation theory into the realm of quantum physics. For the quantum Cook-Levin theorem implies that, at least in principle, there exist quantum many-body systems which *cannot be "cooled to their lowest energy configuration in polynomial time"*. This is particularly enlightening given the original motivation for quantum computation, at least from the perspective of Richard Feynman — that quantum physics seems "difficult" to study with classical computers due to the exponential blowup in dimension. Indeed, the quantum Cook-Levin formally confirms Feynman's intuition, by showing that (assuming QMA $\neq$ BQP) certain properties of low-temperature quantum systems simply cannot be computed efficiently in polynomial time.

This lecture begins by briefly reviewing the classical Cook-Levin theorem, whose techniques will inspire its quantum generalization. We then introduce and motivate "local Hamiltonians", followed by a proof of the quantum Cook-Levin Theorem. For the latter, the soundness proof technique introduces the Geometric Lemma, a lemma with applications beyond quantum complexity theory.

6.1 The Cook-Levin theorem

Recall that the Cook-Levin theorem states the following, for SAT the generalization of k-SAT in which clauses need not be of size at most k.

Theorem 6.1 (Cook-Levin Theorem)**.** *SAT is NP-complete.*

In other words, any instance x of a decision problem L in NP can be efficiently encoded into a CNF Boolean formula $\phi : \{0,1\}^m \to \{0,1\}$, such that $x \in L$ if and only if ϕ is satisfiable. The high-level outline of the proof is also exploited for the quantum Cook-Levin theorem; let us hence sketch a proof of Theorem 6.1 now.

Proof. Let $L \subseteq \{0,1\}^*$ be a language with NP verifier (i.e. deterministic TM) $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, and $z \in \{0,1\}^n$ an input. Recall here that Q is the set of states for M, δ its transition function, and Γ its tape alphabet (which contains the input alphabet, Σ).

We sketch a polynomial-time many-one reduction to SAT, i.e. to a CNF formula $\phi : \{0,1\}^m \to \{0,1\}$ which is satisfiable if and only if $z \in L$. To do so, we design ϕ so that it "simulates" M. Any such simulation must presumably capture the following three key properties:

- **Tape initialization.** Before M begins, its tape must be initialized correctly with the input z and blank symbols elsewhere.
- **Correct propagation.** Step $i+1$ of the computation must legally follow from step i according to the rules of the transition function δ for M.
- **Correct output.** Once the final step of the computation ends, the state of M should be q_{accept} if and only if $z \in L$.

With this high-level view in mind, we shall design ϕ to consist of four "components", $\phi_{\text{in}}, \phi_{\text{prop}}, \phi_{\text{out}}, \phi_{\text{alpha}}$, the first three of which correspond to the three bullet points above, respectively.

Construction sketch. We begin by viewing the computation of M as a sequence of *configurations* of M arranged as rows of a table or *tableau*. Specifically, the ith row of the tableau encodes the ith configuration entered by M. Here, recall that a *configuration* is a snapshot in time of M, and is given by string xqy for $x, y \in \Gamma^*$ and $q \in Q$, where xy denote the current tape contents and q the current state of M. The placement of q to the left of y indicates that the head of M is on the first bit of y.

Exercise 6.2. What does the starting configuration for M look like on input z? How about the accepting configuration?

As a first step, we require some way to map the symbols which can appear in a configuration (i.e. $Q \cup \Gamma$) to the single-bit literals which ϕ can use. This is achieved by the clever idea of defining, for any symbol $s \in Q \cup \Gamma$, an indicator variable $x_{ijs} \in \{0,1\}$, which is set to 1 if and only if cell (i,j) in the tableau contains symbol s. Thus, for example, we can "simulate" a particular cell (i,j) containing at least one valid symbol via formula $\bigvee_{s \in Q \cup \Gamma} x_{ijs}$.

Exercise 6.3. Fix a cell position (i,j). Give a CNF formula encoding the constraint "cell (i,j) contains *precisely one* symbol from $Q \cup \Gamma$".

Exercise 6.4. Use the previous exercise to build ϕ_{alpha}, which should enforce that *all* cells (i,j) contain precisely one symbol from $Q \cup \Gamma$.

With our mapping from $Q \cup \Gamma$ to binary literals in place, we can now sketch the remaining components of ϕ.

- **Tape initialization: ϕ_{in}.** This CNF formula enforces that the first row of the tableau contains the correct starting configuration for M on input z (where recall $q_0 \in Q$ is the start state of M). Namely, if the start configuration is $q_0 z_1 \cdots z_n y_1 \cdots y_s \sqcup \cdots$, for some proof y, and where $\sqcup$ denotes blank symbols on the tape, the formula we construct is:

$$\phi_{\text{in}} = x_{11q_0} \wedge x_{12z_1} \wedge x_{13z_2} \wedge \cdots \wedge x_{1,n+1,z_n} \wedge x_{1,n+s+1,\sqcup} \wedge \cdots .$$

  Above, note that we cannot encode proof y into this formula, since we do not know y ahead of time.

  Exercise 6.5. Fill in the rest of ϕ_{in}. Note that, obviously, we cannot encode all blank symbols on the infinite length tape explicitly into ϕ_{in}. Rather, we may truncate the tape at a finite length — how many cells of the tape are sufficient to keep around?

- **Correct output: ϕ_{out}.** This CNF formula checks whether there exists a time step in which M enters its accepting state:

$$\phi_{\text{out}} = \bigvee_{(i,j)} x_{ijq_{\text{accept}}}.$$

- **Correct propagation: ϕ_{prop}.** Finally, we must enforce that row $i+1$ of the tableau correctly follows from row i. We omit the full details of this construction, but the key observation is that *computation is local* — from time step i to $i+1$, the string encoding the configuration of M can *only change* at the positions directly adjacent to the head's location. For example, if given configuration $c_i = 000q111$, M writes 0, moves the head right, and enters state q', our new configuration is $c_{i+1} = 0000q'11$ — note that only *two* symbols changed between configurations from i to $i+1$.

  Using this observation, it turns out that to ensure that configuration c_{i+1} follows from c_i, it suffices to check all 2×3 "windows" between rows i and $i+1$ of the tableau. Continuing our example above, rows i and $i+1$ of our tableau

0	0	0	q	1	1	1
0	0	0	0	q'	1	1

  are fully characterized by the set of five 2×3 windows:

$$\begin{array}{|ccc|}\hline 0&0&0\\ \hline 0&0&0\\ \hline\end{array} \quad \begin{array}{|ccc|}\hline 0&0&q\\ \hline 0&0&0\\ \hline\end{array} \quad \begin{array}{|ccc|}\hline 0&q&1\\ \hline 0&0&q'\\ \hline\end{array} \quad \begin{array}{|ccc|}\hline q&1&1\\ \hline 0&q'&1\\ \hline\end{array} \quad \begin{array}{|ccc|}\hline 1&1&1\\ \hline q'&1&1\\ \hline\end{array} \tag{6.1}$$

  Here, for example, the first window depicts the first three symbols of rows i and $i+1$, the second window symbols 2 to 4 of rows i and $i+1$, and so forth. Encoding each of these windows into a CNF formula is done analogously to (e.g.) ϕ_{out}.

  Exercise 6.6. Give a CNF formula encoding the constraint that all windows of Equation (6.1) contain the specified symbols.

  Exercise 6.7. Checking 2×3 windows of rows i and $i+1$ seems a bit funny — why does the seemingly more natural idea of simply checking *all* of row i and $i+1$ simultaneously not work? (Hint: How many possible ways are there to validly fill out rows i and $i+1$ of the tableau? How many terms would this lead to in your CNF formula encoding that row $i+1$ correctly follows from row i?)

Finally, the output of the construction is CNF formula $\phi = \phi_{\text{alpha}} \wedge \phi_{\text{in}} \wedge \phi_{\text{out}} \wedge \phi_{\text{prop}}$.

Exercise 6.8. Assuming ϕ_{prop} correctly enforces valid propagation from row i to row $i+1$ of the tableau, why is ϕ satisfiable if and only if $z \in L$? □

It is worth pausing to reflect on the crucial fact that made the Cook-Levin theorem possible — that computation (in the TM model) is *local*, meaning only bits around the head can change in any give time step. Remarkably, it turns out that this is no coincidence; in Section 6.2, we shall see that Nature itself also behaves in a local fashion.

6.2 Local Hamiltonians and Boolean Constraint Satisfaction

We now move from Boolean constraint satisfaction to "quantum constraint satisfaction". Our motivating theme, foreshadowed by the end of Section 6.1, shall be that like "computation", Nature is "local".

Hamiltonians. To appreciate the connection between quantum constraint satisfaction and quantum mechanics, we must return to one of the defining equations of the theory: *Schrödinger's equation*. This equation prescribes how quantum systems evolve in time. Namely, given a starting state $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$, the rate of change of $|\psi\rangle$ with respect to time t is given by the differential equation (ignoring Planck's constant)

$$i\frac{d|\psi\rangle}{dt} = H|\psi\rangle,$$

where $H \in \text{Herm}((\mathbb{C}^2)^{\otimes n})$ is known as a *Hamiltonian*. In other words, the change in $|\psi\rangle$ through time t is fully specified by H. By solving this equation, one obtains that after time t, our new state $|\psi_t\rangle$ is given by

$$|\psi_t\rangle = e^{-iHt}|\psi\rangle.$$

Exercise 6.9. What type of operator is e^{-iHt}? How does this explain the time evolution postulate of quantum mechanics, which states that the set of allowed operations on quantum states is precisely the set of unitary operations?

Local Hamiltonians. Schrödinger's equation tells us that, in principle, any Hamiltonian $H \in \text{Herm}(\mathbb{C}^2)^{\otimes n}$ describes the evolution of *some* quantum system. The natural question is now: *Which Hamiltonians actually arise in Nature?* It is here that Nature takes a page from theoretical computation's book (or perhaps it is theoretical computation which has taken a page from Nature's book), in that essentially all known naturally occurring quantum systems evolve according to *local* Hamiltonians, which we now define.

Definition 6.10 (k-local Hamiltonian)**.** *A Hermitian operator $H \in \text{Herm}((\mathbb{C}^2)^{\otimes n})$ acting on n qubits is a k-local Hamiltonian if it can be written*

$$H = \sum_{\{S|S\subseteq[n] \text{ s.t. } |S|=k\}} H_S \otimes I_{[n]\setminus S},$$

where each operator $H_S \in \text{Herm}((\mathbb{C}^2)^{\otimes k})$ acts on the subset S of qubits. (Note that we allow $H_S = 0$.) The eigenvalues of H denote energy levels *of the system described by H, with $\lambda_{\min}(H)$ denoting the* ground state energy*. The eigenvectors corresponding to $\lambda_{\min}(H)$ are* ground states*.)*

Let us dissect this definition, as it will play a crucial role in the remainder of this course.

- Definition 6.10 says that the action of H on all n qubits is fully specified by a set of *local* operators H_S, each acting non-trivially on some subset S of k out of n qubits. For example, the following are 2-local Hamiltonians (subscripts denote qubit indexes acted on by the respective operators, and Z is the Pauli operator):

$$H_1 = Z_1 \otimes Z_2, \qquad H_2 = Z_1 \otimes Z_2 \otimes I_{3,4} + I_1 \otimes Z_2 \otimes Z_3 \otimes I_4 + I_{1,2} \otimes Z_3 \otimes Z_4.$$

  The first of these acts on a 2-qubit system, and the second on a 4-qubit system. For brevity, we typically omit the identity terms and simply write $H_2 = Z_1 \otimes Z_2 + Z_2 \otimes Z_3 + Z_3 \otimes Z_4$.

  Exercise 6.11. What are the matrix representations of H_1 and H_2 above?

- Hamiltonians describing physical quantum systems are typically[1] k-local for *constant* k, and there is something very special about this, which you will explore in the next exercise.

  Exercise 6.12. How many bits are required to specify an *arbitrary* Hamiltonian H? How about a k-local Hamiltonian for $k \in O(1)$? What does this tell us about our ability to efficiently represent the dynamics of typical physical systems in Nature?

  Thus, although in principle, one requires exponential space to write down the Hamiltonian governing an arbitrary many-body quantum system, in *practice* this is one place we catch a break — for physical systems, we can at least succinctly describe the rules governing their time evolution. However, let us be clear that this is just a minor concession by quantum physics — for even though we can *describe* the dynamics, *computing properties* of the evolution is complexity theoretically hard.

Physical motivation. The eigenvalues of a local Hamiltonian (more generally, of any Hamiltonian) are known as *energy levels*, literally because they represent the energy levels the system may settle into. The quantum state of the system at energy level λ is none other than the eigenvector $|\lambda\rangle$ satisfying $H|\lambda\rangle = \lambda|\lambda\rangle$.

Remark. *Despite the fact that quantum time evolution is continuous (i.e. described by unitary maps), the Schrödinger equation highlights that a quantum system may settle into only a* discrete *or "quantized" set of energy values $\{\lambda_i\}$. Indeed, this type of "quantization" phenomenon is precisely what gives* quantum *mechanics its name.*

The *ground state energy* $\lambda_{\min}(H)$ plays a particularly important role — it describes the energy level the system will relax into when cooled to very low temperature (think billionths of a degree above absolute zero[2]). This regime is particularly important, as it gives rise to

[1] This is an idealization — for any actual many-body quantum system, one can only guess at what the defining Hamiltonian should be, and attempt to corroborate this via experiment. Thus, what we write down as "the defining local Hamiltonian" for a system is really a well-motivated model or *approximation* to the true dynamics of the system. It just so happens that setting $k \in O(1)$ typically suffices to accurately reproduce the desired local physical properties of quantum systems.

[2] Again, this is an excellent opportunity to procrastinate by heading to the Wikipedia page for "absolute zero". For example, a 1999 experiment cooled nuclear spins in rhodium metal to 0.0000000001 Kelvin. (Recall absolute zero is defined as 0 Kelvin.)

exotic phenomena such as superconductivity and superfluidity. It is thus of utmost importance to fields such as materials design to be able to understand and predict the properties of such low temperature systems; in particular, this means one wishes to understand the properties of $\lambda_{\min}(H)$ and its corresponding eigenvector, the ground state $|\lambda_{\min}(H)\rangle$. Unfortunately, it turns out that $\lambda_{\min}(H)$ is hard to estimate, not least of which because one can embed the answers to NP-complete problems such as 3-SAT into it, as we now discuss.

Embedding k-SAT into local Hamiltonians. We said above that even though a k-local Hamiltonian $H \in \mathrm{Herm}((\mathbb{C}^2)^{\otimes n})$ has a succinct representation in n, computing properties of H is hard. The intuitive reason for this is that H is really the quantum analogue of a k-SAT formula $\phi : \{0,1\}^n \to \{0,1\}$; even though ϕ describes a truth table of size 2^n, it also has a succinct representation of size $\mathrm{poly}(n)$, and computing its properties is NP-complete. We may formalize this connection as follows.

Consider a 2-SAT clause $c = (x_1 \vee \overline{x}_2)$, which has unique unsatisfying assignment $|01\rangle$. We may embed this into a 2-local Hamiltonian term

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

so that any *satisfying assignment* $x \in \{0,1\}^2$ to c is a ground state of H. More generally, we think of the rows of H as labelled by binary strings 00, 01, 10, 11, and denote the label of row r as $l(r)$. Then, for a 2-SAT constraint c, define H to be all-zeroes except on the diagonal, where we place a 1 if and only if $l(r)$ is a non-satisfying assignment to c.

Exercise 6.13. What is the ground state energy of H?

Exercise 6.14. Prove that for any $x \in \{0,1\}^2$, $\langle x|H|x\rangle = 0$ if $c(x) = 1$, and $\langle x|H|x\rangle = 1$ if $c(x) = 0$. Conclude that any satisfying assignment to c is a ground state of H. Given the structure of H, why can we assume without loss of generality that the best assignment is a standard basis state?

This construction generalizes directly to *any* Boolean function $c : \{0,1\}^k \to 0,1$, so that (say) a 3-SAT clause c is embedded into a $2^k \times 2^k$ quantum constraint H.

Exercise 6.15. Give the quantum constraint H encoding the 3-SAT clause $c = (\overline{x_1} \vee x_2 \vee \overline{x_3})$.

Suppose now we have three clauses $c_1 = (x_1 \vee x_2)$, $c_2 = (\overline{x_2} \vee x_3)$, $c_3 = (x_3 \vee x_4)$. The full CNF formula $\phi = c_1 \wedge c_2 \wedge c_3$ is given by adding all quantum constraints H_{c_i} for each clause c_i:

$$H = H_{c_1} \otimes I_{3,4} + I_1 \otimes H_{c_2} \otimes I_4 + I_{1,2} \otimes H_{c_3}.$$

Exercise 6.16. Prove that $|x\rangle$ for $x \in \{0,1\}^4$ satisfies $\langle x|H|x\rangle = 0$ if and only if $\phi(x) = 1$. More generally, prove that $\langle x|H|x\rangle$ counts the number of unsatisfied clauses for x with respect to ϕ.

Exercise 6.17. We are almost ready to conclude that "the ground state energy of H equals 0 if and only if ϕ is satisfiable", thus yielding that estimating $\lambda_{\min}(H)$ is NP-hard if ϕ is a 3-SAT formula. It only remains to prove that if ϕ is unsatisfiable, without loss of generality the *best* assignment $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ (i.e. minimizing $\langle\psi|H|\psi\rangle$) is an n-bit string $|x\rangle$, as opposed to a genuinely quantum state on n qubits; show this.

Exercise 6.18. The ground state energy of H above encodes something more precise than whether ϕ is satisfiable — what does it actually *count*?

Exercise 6.19. Give a 2-local Hamiltonian whose ground state energy encodes the problem MAX CUT. Recall the latter is defined as: Given a simple, undirected graph $G = (V, E)$, partition E into sets E_1 and E_2 so that the maximum number of edges possible crosses between E_1 and E_2. (Hint: For each edge $(i, j) \in E$, start by thinking about 2-local quantum constraint $H_{ij} = I - Z_i \otimes Z_j$; what is the matrix representation of H_{ij}?)

6.3 The quantum Cook-Levin theorem

We have seen in Section 6.2 that, given a 3-local Hamiltonian H as input, estimating its ground state energy $\lambda_{\min}(H)$ allows us to decide 3-SAT instances ϕ, and is thus NP-hard. The problem of estimating $\lambda_{\min}(H)$ more generally (i.e. for non-diagonal H) is sufficiently important to warrant its own name.

Definition 6.20 (k-local Hamiltonian problem (k-LH))**.** *Fix a polynomial $p : \mathbb{N} \to \mathbb{R}^+$. The promise problem k-LH is defined as follows.*

- *Input:*
  - *A k-local Hamiltonian $H = \sum_S H_S \in \mathrm{Herm}((\mathbb{C}^2)^{\otimes n})$.*
  - *Efficiently computable threshold functions $\alpha(n), \beta(n) \in \mathbb{R}$ satisfying promise gap $\alpha(n) - \beta(n) \geq 1/p(n)$ $\forall n \geq 1$.*
- *Output:*
  - *If $\lambda_{\min}(H) \leq \alpha(n)$, accept.*
  - *If $\lambda_{\min}(H) \geq \beta(n)$, reject.*
  - *Else, accept or reject arbitrarily.*

A few remarks are in order: (1) Unlike the examples of k-SAT from Section 6.2, in Definition 6.20 H need not be diagonal in the standard basis, and the sets of qubits S acted on H_S need not be constrained in any particular geometric fashion (say, on a 1D chain). (2) Technically, one should write k-LH(p), since the problem is parameterized by the promise gap polynomial p. We will implicitly set p to the polynomial arising from the QMA-hardness reduction for k-LH below, and henceforth simply write k-LH. (3) The fact that the promise gap $\alpha(n) - \beta(n)$ is at least inverse polynomial is crucial; making the gap, say, inverse *exponential* yields a much more difficult PSPACE-complete problem, rather than a QMA-complete problem for k-LH as defined here. (4) In principle, the number of terms H_S, denoted m, need not be polynomial in n, the number of qubits. Thus, p, α, and β should more generally depend on both m and n. However, for simplicity and due to physical motivation, the community typically assumes $m \in \mathrm{poly}(n)$ and thus drops the m parameter.

Exercise 6.21. Why is it not a problem if, say, $m \in \Theta(2^n)$? (Hint: To a computer scientist, relative to which input parameter do we typically define run-times? (It's not just the number of qubits, n.))

Statement of the quantum Cook-Levin theorem. The quantum Cook-Levin theorem of Kitaev states that, just as k-SAT is NP-complete for $k \geq 3$, k-LH is QMA-complete for $k \geq 5$. In other words, Feynman's intuition was right — estimating the ground state energy of a local Hamiltonian, and hence more generally properties of quantum many-body systems, is provably *hard* (assuming QMA $\neq$ BQP).

Theorem 6.22 (Quantum Cook-Levin Theorem)**.** *There exists an efficiently computable polynomial $p : \mathbb{N} \to \mathbb{R}^+$ such that* 5-LH *with promise gap p is QMA-complete.*

The QMA-hardness result above can be improved to $k \geq 2$, even if one has (higher-dimensional) quantum systems on a 1D chain. Due to time constraints and for pedagogical reasons, however, we shall restrict our exposition to Theorem 6.22. As with any completeness proof, we proceed by showing containment in QMA in Section 6.3.1, followed by a proof of QMA-hardness in Section 6.3.2.

6.3.1 Containment in QMA

Lemma 6.23. *For any polynomial $p : \mathbb{N} \to \mathbb{R}^+$ and $k \in O(\log n)$,* k-LH $\in$ QMA.

Proof. Let (H, α, β) be an instance of k-LH. We wish to decide in QMA whether $\lambda_{\min}(H) \leq \alpha$ or $\lambda_{\min}(H) \geq \beta$, assuming one of the two is the case. In the YES case, the proof is obvious — the prover sends the ground state $|\psi\rangle$ satisfying $H|\psi\rangle = \lambda_{\min}(H)|\psi\rangle$. The question is: *How do we verify that $\langle\psi|H|\psi\rangle \leq \alpha$?*

The key insight is that H is Hermitian, and hence may be viewed as an observable. Recall that for any observable H with spectral decomposition $H = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$, H encodes a projective measurement in basis $\{|\psi_i\rangle\}$ with corresponding outcomes λ_i. Moreover, the expected value of measuring $|\psi\rangle$ with H is

$$\mathrm{Tr}(H|\psi\rangle\langle\psi|) = \langle\psi|H|\psi\rangle.$$

In other words, if we could simulate a measurement in the eigenbasis of H, then we could estimate $\langle\psi|H|\psi\rangle$.

Exercise 6.24. What does it mean to "simulate a measurement in a given basis B"? (Hint: Which unitary operation must we be able to perform if we restrict our circuits to only perform measurements in the standard basis, as we did for BQP?)

Unfortunately, the eigenbasis of H may be quite complicated; thus, we cannot directly measure with respect to it. However, the *local* structure of H allows us to approximate such a measurement in a simple fashion: Since by linearity

$$\langle\psi|H|\psi\rangle = \sum_S \langle\psi|H_S|\psi\rangle,$$

intuitively we may equivalently measure the local terms H_S.

Exercise 6.25. Suppose $H_S \in \mathrm{Herm}(\mathbb{C}^2)^{\otimes k}$ for $k \in O(\log n)$. How can one efficiently simulate a measurement of $|\psi\rangle$ with respect to the eigenbasis of H_S? (Hint: Use the previous exercise on simulating arbitrary measurements via standard basis measurements.)

We hence apply the following procedure, denoted V: Suppose there are m terms H_S. Pick a term H_S uniformly at random from H, and simulate a measurement with respect to observable $\langle\psi|H_S|\psi\rangle$. Since expectation is linear, the total expectation for this procedure with respect to the random choice of S is

$$\sum_S \Pr[\text{picking } S] \cdot \langle\psi|H_S|\psi\rangle = \frac{1}{m}\sum_S \langle\psi|H_S|\psi\rangle = \frac{1}{m}\langle\psi|\left(\sum_S H_S\right)|\psi\rangle = \frac{1}{m}\langle\psi|H|\psi\rangle.$$

Exercise 6.26. In principle, m may be exponential with respect to the number of qubits n. Why is this not a problem for the protocol above? (Hint: With respect to which parameter must we run efficiently?)

Recalling the Courant-Fischer variational characterization of eigenvalues from Lecture 4, we conclude that in the YES case, there exists a quantum proof $|\psi\rangle$ such that the expected output value of V is at most α/m, and in the NO case, any proof $|\psi\rangle$ has expected value at least β/m.

There is only one thing left to do — the definition of QMA says nothing about *expectation values* of the verification. Rather, we must strengthen our expectation bounds for V to a *high-probability statement*: In the YES case, our verifier must accept with probability at least $2/3$, and in the NO case with probability at most $1/3$.

Exercise 6.27. Give an example of a probability distribution over an appropriate sample space so that the expected value is 0, and yet the probability of obtaining any outcome with value v satisfying $|v| \leq \epsilon$ (for some fixed $\epsilon > 0$) is zero. In other words, a statement about expectation is in general *not* sufficient to yield a high-probability statement.

To give a high probability statement, we employ the *Höffding bound*, which is worth knowing in its own right. Let $\Omega \subseteq \mathbb{R}$ denote our sample space, meaning the union over all eigenvalues of all terms H_S. Let $X_i \in \Omega$ denote the random variable corresponding to the measurement outcome of the ith run of V, given state $|\psi\rangle$. Intuitively, if we repeat V N times and take the average measurement result, $A := (\sum_{i=1}^N X_i)/N$, we might expect A to approximate the true expected value, $\langle\psi|H|\psi\rangle$. To formalize this, the Höffding bound says that if $a_i \leq X_i \leq b_i$ for all i, and if the X_i are independent, then

$$\Pr[|A - E[A]| \geq t] \leq 2^{-\frac{2N^2t^2}{\sum_{i=1}^N (b_i - a_i)^2}}.$$

Thus, the final verification procedure, denoted V', is precisely this: Take in N copies of the proof $|\psi\rangle$. For each copy $|\psi_i\rangle$, independently repeat the procedure V to obtain outcome $X_i \in \Omega$. If $A := (\sum_{i=1}^N X_i)/N \leq \alpha + (1/4p(n))$, accept, and if $A \geq \beta - (1/4p(n))$, reject.

Exercise 6.28. Use the Höffding bound to prove that in the YES case, for sufficiently large polynomial N, V' accepts with probability exponentially close to 1. For simplicity, you may assume $0 \preceq H_S \preceq I$ for all H_S, i.e. all H_S have eigenvalues between 0 and 1, and so $a_i = 0$ and $b_i = 1$ in the statement of the Höffding bound.

Exercise 6.29. In the NO case, the prover can again cheat by sending a large entangled state over the N copies of the proof space; why does V' still reject with high probability in this case? □

6.3.2 Hardness for QMA

In Section 6.3.1, we showed k-LH $\in$ QMA for $k \in O(\log n)$. Theorem 6.22 thus immediately follows from the following lemma.

Lemma 6.30. k-LH *is QMA-hard under polynomial-time many-one reductions for* $k \geq 5$.

Proof. The proof is based on an old trick of Feynman, employed in a clever way by Kitaev; a similar trick was used in the proof of BQP-completeness for the Matrix Inversion problem from Lecture 4 (which was discovered after Theorem 6.22). To begin, let $A = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ denote a QMA promise problem. Let $x \in \{0,1\}^n$ be an input, with corresponding QMA verifier $V = V_m \cdots V_1 \in (\mathbb{C}^2)^{\otimes n} \otimes (\mathbb{C}^2)^{\otimes p(n)} \otimes (\mathbb{C}^2)^{\otimes q(n)}$. Recall V is a uniformly generated quantum verification circuit consisting of 1- and 2-qubit unitary gates, acting on registers A (n qubits containing the input x), B ($p(n)$ qubits containing the proof $|\psi\rangle$), and C ($q(n)$ ancilla qubits initialized to all zeroes). Assume without loss of generality that the completeness and soundness parameters for V are $1-\epsilon$ and ϵ, so that $1 - 2\epsilon \in \Omega(1/\operatorname{poly}(n))$. Our goal is to construct an instance (H, α, β) of 5-local Hamiltonian H such that, if $x \in A_{\text{yes}}$, then $\lambda_{\min}(A) \leq \alpha$, and if $x \in A_{\text{no}}$, then $\lambda_{\min}(A) \geq \beta$.

Construction. The high level setup is analogous to that of the classical Cook-Levin theorem (Theorem 6.1), in that we will track a sequence of "quantum configurations"' $|\psi_t\rangle$ over time, and use "local Hamiltonian checks" to ensure the propagation from configuration $|\psi_t\rangle$ to $|\psi_{t+1}\rangle$ proceeds correctly. The main difference is that instead of encoding each configuration as a row of a tableau, we shall encode it as a term in a superposition, i.e. as $\sum_t |\psi_t\rangle$. Unfortunately, in doing so, we lose our notion of *time*, in that for a tableau, time was encoded by *position* — the row index of a configuration in the tableau gave away the time step during which the configuration was entered by the verifier. To recover this, we use an idea of Feynman and attach a new ancilla register to track time, D, denoted the "clock" register. Thus, we aim to encode the computation as a state of the form $\sum_t |\psi_t\rangle_{A,B,C}|t\rangle_D$.

It remains to specify what a "quantum configuration" $|\psi_t\rangle$ for time t should be — but this is easy, since at time t we have applied the first t gates. In other words, defining $|\psi_t\rangle := V_t \cdots V_1 |x\rangle_A |\psi\rangle_B |0\cdots 0\rangle_C$, we arrive at the so-called *history state*

$$|\psi_{\text{hist}}\rangle = \frac{1}{\sqrt{m+1}} \sum_{t=0}^{m} V_t \cdots V_1 |x\rangle_A |\psi\rangle_B |0\cdots 0\rangle_C |t\rangle_D.$$

Just as Theorem 6.1 used local Boolean checks to force a tableau to have certain properties, we now use local Hamiltonian terms to force a quantum state to look like $|\psi_{\text{hist}}\rangle$. Specifically, we will design $H = H_{\text{in}} + H_{\text{out}} + H_{\text{prop}}$ with the hope of making $|\psi_{\text{hist}}\rangle$ its ground state.

- **Ancilla initialization: H_{in}.** This constraint enforces that at time step $t = 0$, the A register reads $|x\rangle$ for x the input, and the ancilla register C is all zeroes:

$$H_{\text{in}} = (I - |x\rangle\langle x|)_A \otimes I_B \otimes I_C \otimes |0\rangle\langle 0|_D + I_A \otimes I_B \otimes (I - |0\cdots 0\rangle\langle 0\cdots 0|)_C \otimes |0\rangle\langle 0|_D.$$

Exercise 6.31. Prove $H_{\text{in}} \succeq 0$.

Exercise 6.32. Let $|\phi(y)\rangle := |x\rangle_A|\psi\rangle_B|y\rangle_C|0\rangle_D$. Prove that $\langle\phi(y)|H_{\text{in}}|\phi(y)\rangle$ equals 0 if $y = 0^{q(n)}$, and equals 1 if y has Hamming weight at least one.

Exercise 6.33. As stated, H_{in} is not local — the projector in C, for example, acts non-trivially and simultaneously on $q(n)$ qubits. Show that replacing $\Delta := (I - |0\cdots0\rangle\langle 0\cdots0|)_C$ with $\Delta' := \sum_{i=1}^{q(n)} |1\rangle\langle 1|_{C_i}$ in H_{in} obeys the same properties regarding $|\phi(y)\rangle$ as in the previous exercise, and that Δ' is 1-local. How can we similarly reduce the locality of $(I - |x\rangle\langle x|)_A$ to 1 in H_{in}?

- **Correct output: H_{out}.** This constraint checks whether, at time step m, the verifier accepted (recall the verifier's output qubit is C_1):

$$H_{\text{out}} = I_A \otimes I_B \otimes |0\rangle\langle 0|_{C_1} \otimes |m\rangle\langle m|_D.$$

Exercise 6.34. Why do we project onto $|0\rangle\langle 0|_{C_1}$ above, as opposed to $|1\rangle\langle 1|_{C_1}$?

Exercise 6.35. Prove $H_{\text{out}} \succeq 0$.

- **Correct propagation: H_{prop}.** Interestingly, specifying the propagation Hamiltonian H_{prop} is simpler than specifying ϕ_{prop} classically. Namely,

$$H_{\text{prop}} = \sum_{t=0}^{m-1} -V_{t+1} \otimes |t+1\rangle\langle t|_D - V_{t+1}^\dagger \otimes |t\rangle\langle t+1|_D + I \otimes |t\rangle\langle t|_D + I \otimes |t+1\rangle\langle t+1|_D,$$

where recall V_t acts on A, B, C. The intuition is best captured by the first term above, which encodes the idea that in going from time step t to $t+1$, we must apply V_{t+1}.

Exercise 6.36. For any state of the form $|\phi\rangle = \frac{1}{\sqrt{m+1}} \sum_{t=0}^{m} V_t \cdots V_1 |\eta\rangle_{A,B,C}|t\rangle_D$, prove that $H_{\text{prop}}|\phi\rangle = 0$. Conclude that vectors encoding "correct propagation according to V" fall into the null space of H_{prop}.

Finally, the output of the construction is Hamiltonian $H = H_{\text{in}} + H_{\text{prop}} + H_{\text{out}}$. We will choose α and β as needed based on the correctness analysis below.

Exercise 6.37. Is H as specified 5-local? (Hint: Think about the clock register.) We will revisit this question at the end of the proof.

Correctness. We now show correctness.

Completeness. Assume first $x \in A_{\text{yes}}$. Then, there exists a proof $|\psi\rangle$ accepted by V with probability at least $1 - \epsilon$. We must prove that $\lambda_{\min}(H) \leq \alpha$, or equivalently, there exists $|\phi\rangle$ such that $\langle\phi|H|\phi\rangle \leq \alpha$. The obvious choice is to choose $|\phi\rangle$ as the history state $|\psi_{\text{hist}}\rangle$. Then,

$$\langle\psi_{\text{hist}}|H|\psi_{\text{hist}}\rangle = \langle\psi_{\text{hist}}|H_{\text{in}}|\psi_{\text{hist}}\rangle + \langle\psi_{\text{hist}}|H_{\text{out}}|\psi_{\text{hist}}\rangle + \langle\psi_{\text{hist}}|H_{\text{prop}}|\psi_{\text{hist}}\rangle = \langle\psi_{\text{hist}}|H_{\text{out}}|\psi_{\text{hist}}\rangle,$$

which follows by the previous exercises in this proof.

Exercise 6.38. Prove that $\langle\psi_{\text{hist}}|H_{\text{out}}|\psi_{\text{hist}}\rangle = \frac{1}{m+1}\Pr[V \text{ rejects } |\psi\rangle] \leq \frac{\epsilon}{m+1}$.

Thus, setting $\alpha := \frac{\epsilon}{m+1}$, we have shown the YES case.

Soundness. Assume now $x \in A_{\text{no}}$. Then, for all proofs $|\psi\rangle$, V accepts $|\psi\rangle$ with probability at most ϵ. We must prove that $\lambda_{\min}(H) \geq \beta$, or equivalently, for all states $|\phi\rangle$, $\langle\phi|H|\phi\rangle \geq \beta$. Unfortunately, due to the universal quantifier on $|\psi\rangle$ we can no longer give a simple constructive proof as in the YES case. Things are further complicated by the fact that the terms $H_{\text{in}}, H_{\text{out}}, H_{\text{prop}}$ do not pairwise commute, so it is not in general true that $\lambda_{\min}(H) = \lambda_{\min}(H_{\text{in}}) + \lambda_{\min}(H_{\text{out}}) + \lambda_{\min}(H_{\text{prop}})$.

Exercise 6.39. If $[A, B] = 0$ for normal operators A, B, prove that $\lambda_{\min}(A + B) = \lambda_{\min}(A) + \lambda_{\min}(B)$. (Hint: Recall the simultaneous diagonalization theorem, which states that normal operators A and B commute if and only if they diagonalize in a common eigenbasis.)

As lower bounding $\lambda_{\min}(H)$ is rather involved, let us state the result below as a lemma, and prove it in Section 6.3.2.

Lemma 6.40. *If $x \in A_{\text{no}}$, it holds that $\lambda_{\min}(H_{\text{in}} + H_{\text{out}} + H_{\text{prop}}) \geq \frac{\pi^2(1-\sqrt{\epsilon})}{2(m+1)^3}$.*

Thus, setting $\beta = \frac{\pi^2(1-\sqrt{\epsilon})}{2(m+1)^3}$ completes the proof of the NO case, with the exception of one observation.

Exercise 6.41. Is it true in general that $\alpha - \beta \geq 1/\operatorname{poly}(n)$? What value of ϵ suffices for this to hold, and why can we let ϵ take this value without loss of generality?

The last stand: Locality. We have shown correctness for our many-one reduction to Hamiltonian $H = H_{\text{in}} + H_{\text{out}} + H_{\text{prop}}$. Congratulations! There is one tiny problem, however; as foreshadowed in a previous exercise, H is technically not $O(1)$-local. This is because implicitly we have assumed that the clock register D is encoded in *binary*, and is hence $\Theta(\log m)$ qubits in size. To correctly identify a time step written in binary, we must read *all* the bits in D; thus, the projectors onto D in $H_{\text{in}}, H_{\text{out}}, H_{\text{prop}}$ are all $O(\log m)$-local.

The solution is rather simple; let us encode D in *unary*. Specifically, encode time $t \in \{0, \ldots, m\}$ as 1^t0^{m-t}. There is a tradeoff here — now the D register is m qubits, up from $\Theta(\log m)$ qubits. However, to check the current time step, we can do a *local* check — namely, we just have to identify the position of the leading 1 in 1^t0^{m-t}. And this identification can be made by checking at most 3 qubits of D at a time. With respect to this new encoding, one can rewrite the terms of $H_{\text{in}}, H_{\text{out}}, H_{\text{prop}}$ acting on D to obtain a new Hamiltonian H (we omit this for brevity). Since it now takes three qubit checks to identify the time in register D, and since H_{prop} now pairs such 3-local clock checks with 2-local gates V_t, the new Hamiltonian H we get is 5-local as claimed.

The final hitch is that we now have to add yet more constraints to H to enforce that the states appearing in D are indeed valid unary time encodings of form $1^t0^m - t$ (e.g. we want to disallow setting D to $|01^{m-1}\rangle$). This is accomplished by adding a fourth Hamiltonian term to H:

$$H_{\text{stab}} := I_{A,B,C} \otimes \sum_{i=1}^{m-1} |0\rangle\langle 0|_i \otimes |1\rangle\langle 1|_{i+1}.$$

Exercise 6.42. Why does H_{stab} correctly enforce unary time encodings in register D?

Our final Hamiltonian is $H = H_{\text{in}} + H_{\text{out}} + H_{\text{prop}} + H_{\text{stab}}$, with the time register encoded in unary. Of course, now one must revisit the soundness analysis to account for the H_{stab} term (we do not need to repeat the completeness analysis, since in the YES case an honest prover will correctly encode D anyway). It turns out this can be done rather easily; nevertheless, again we shall omit it for brevity. □

Proof of soundness via Geometric Lemma

We now prove the eigenvalue lower bound of Lemma 6.40 required to complete the soundness analysis of Theorem 6.22.

Proof of Lemma 6.40. It will be enlightening to first rewrite H_{prop} via a unitary change of basis, i.e. to instead consider $UH_{\text{prop}}U^\dagger$ for some cleverly chosen U.

Exercise 6.43. Why is $\lambda_{\min}(H_{\text{in}} + H_{\text{prop}} + H_{\text{out}}) = \lambda_{\min}(UH_{\text{in}}U^\dagger + UH_{\text{prop}}U^\dagger + UH_{\text{out}}U^\dagger)$? Conclude that there is no loss of generality in applying a unitary change of basis for the purposes of our proof. More generally, observe that for eigenvalue problems, a trick one should always keep in mind is the potential for viewing the problem in a different basis (which may simplify the matrices involved).

A convenient change of basis. The intuition for choosing U is this: H_{prop}, reproduced below for convenience,

$$H_{\text{prop}} = \sum_{t=0}^{m-1} -V_{t+1} \otimes |t+1\rangle\langle t|_D - V_{t+1}^\dagger \otimes |t\rangle\langle t+1|_D + I \otimes |t\rangle\langle t|_D + I \otimes |t+1\rangle\langle t+1|_D,$$

"feels" a lot like a random walk on a line. It has four equally weighted terms, two of which correspond to staying in the same spot (i.e. time step) and doing nothing (i.e. I), one of which corresponds to moving to right one step on the line (i.e. forward one time step) and applying V_{i+1}, and one of which corresponds to moving left one step on the line (i.e. backward one time step) and applying $V_{i+1}^\dagger$. In fact, if it wasn't for those pesky V_{i+1} terms, it would essentially be a random walk on a line, where with probability $1/2$ we stay put, and otherwise we flip a fair coin and move right one step if we get heads and left one step if we get tails. And indeed, we can get rid of those V_{i+1} terms by conjugating H_{prop} by unitary

$$U = \sum_{t=0}^{m} V_1^\dagger \cdots V_t^\dagger \otimes |t\rangle\langle t|_D.$$

Exercise 6.44. Prove U is unitary.

Exercise 6.45. Prove that

$$UH_{\text{prop}}U^\dagger = \sum_{t=0}^{m-1} -I \otimes |t+1\rangle\langle t|_D - I \otimes |t\rangle\langle t+1|_D + I \otimes |t\rangle\langle t|_D + I \otimes |t+1\rangle\langle t+1|_D.$$

The exercise above shows that under the change of basis U, H_{prop} acts non-trivially *only on the clock register*, D. Thus, it has a nice matrix representation via $UH_{\text{prop}}U^\dagger = I_{A,B,C} \otimes \Lambda_D$ for

$$\Lambda := \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & \cdots \\ -1 & 2 & -1 & 0 & 0 & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ 0 & 0 & -1 & 2 & -1 & \cdots \\ 0 & 0 & 0 & -1 & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}.$$

Exercise 6.46. Prove that Λ indeed has the matrix representation above.

Exercise 6.47. Write down the transition matrix for a random walk on a line of length $m+1$, where with probability $1/2$ we stay put, and otherwise we flip a fair coin and move right one step if we get heads and left one step if we get tails. How does Λ compare with this transition matrix?

The advantage of this representation for $UH_{\text{prop}}U^\dagger$ is that it is easier to understand its spectral decomposition (indeed, the matrix is "almost diagonal" now). Using tools from the analysis of 1D random walks, one may now show that the eigenvalues of Λ are $\lambda_k(\Lambda) = 2(1 - \cos(\pi k/(m+1)))$, a fact we will use later.

Exercise 6.48. What is the full set of eigenvalues for H_{prop}?

Exercise 6.49. Prove that the *unique* null vector (and in this case, ground state) of Λ is

$$|\gamma\rangle := \frac{1}{\sqrt{m+1}} \sum_{t=0}^{m} |t\rangle.$$

(Hint: Checking that $\Lambda|\gamma\rangle = 0$ is straightforward. For uniqueness, use the fact that if a normal operator has a non-degenerate spectrum (i.e. all eigenvalues are distinct), then it has a unique eigenbasis.)

By applying the change of basis U, we hence have a full understanding of H_{prop}. It remains to understand how U affects the other important quantities in this proof: H_{in}, H_{out}, and $|\psi_{\text{hist}}\rangle$.

Exercise 6.50. Prove that $UH_{\text{in}}U^\dagger = H_{\text{in}}$.

Exercise 6.51. Prove that $UH_{\text{out}}U^\dagger = (V^\dagger_{A,B,C} \otimes I_D)H_{\text{out}}(V_{A,B,C} \otimes I_D)$.

Exercise 6.52. Prove that $U|\psi_{\text{hist}}\rangle = |x\rangle_A|\psi\rangle_B|0\cdots0\rangle_C|\gamma\rangle_D$.

In the remainder of this proof, we hence work in this new basis and refer to quantities $H'_{\text{in}} := H_{\text{in}}$, $H'_{\text{out}} := (V^\dagger_{A,B,C} \otimes I_D)H_{\text{out}}(V_{A,B,C} \otimes I_D)$, $H'_{\text{prop}} := I_{A,B,C} \otimes \Lambda_D$, and $|\psi'_{\text{hist}}\rangle := |x\rangle_A|\psi\rangle_B|0\cdots0\rangle_C|\gamma\rangle_D$.

The Geometric Lemma. Recall that our goal is now to prove $\lambda_{\min}(H'_{\text{in}} + H'_{\text{out}} + H'_{\text{prop}}) \geq \beta$, and the difficulty is that the matrices involved do not all pairwise commute. However, one pair *does* commute.

Exercise 6.53. Prove that $[H_{\text{in}}, H_{\text{out}}] = 0$.

Let us hence write $G := H'_{\text{in}} + H'_{\text{out}}$, so that we want to lower bound $\lambda_{\min}(G + H'_{\text{prop}})$. For this, we will need a technical tool known as the *Geometric Lemma*, whose intuition we now explain. We know from our previous discussion that $G \succeq 0$ and $H'_{\text{prop}} \succeq 0$. Thus, for any $|\phi\rangle$ the minimum expectation we can hope for against G and H_{prop} is $\langle\phi|G|\phi\rangle = 0$ and $\langle\phi|H'_{\text{prop}}|\phi\rangle = 0$. The question is: *Can we attain zero for both expressions simultaneously, i.e. do G and H'_{prop} share a common null vector?* Unfortunately, the answer is no.

Exercise 6.54. Show that $\text{Null}(H'_{\text{prop}}) = (\mathbb{C}^2)_A^{\otimes n} \otimes (\mathbb{C}^2)_B^{\otimes p(n)} \otimes (\mathbb{C}^2)_C^{\otimes q(n)} \otimes |\gamma\rangle_D$.

Exercise 6.55. Show that $\text{Null}(G)$ is the direct sum of 3 orthogonal spaces, i.e. $\text{Null}(G) = N_1 \oplus N_2 \oplus N_2$ for

$$\begin{aligned} N_1 &:= |x\rangle_A \otimes (\mathbb{C}^2)_B^{\otimes p(n)} \otimes |0\cdots 0\rangle_C \otimes |0\rangle_D \\ N_2 &:= (\mathbb{C}^2)_A^{\otimes n} \otimes (\mathbb{C}^2)_B^{\otimes p(n)} \otimes (\mathbb{C}^2)_C^{\otimes q(n)} \otimes \text{Span}(|1\rangle, \ldots, |m-1\rangle)_D \\ N_3 &:= \{|\eta\rangle \mid V|\eta\rangle \text{ has qubit } C_1 \text{ set to } |1\rangle\}_{A,B,C} \otimes |m\rangle_D \end{aligned}$$

Exercise 6.56. Conclude that $\text{Null}(H'_{\text{prop}}) \cap \text{Null}(G) = \emptyset$, as claimed.

We thus have that for any choice of $|\phi\rangle$, at most one of $\langle\phi|G|\phi\rangle$ and $\langle\phi|H'_{\text{prop}}|\phi\rangle$ can be zero; thus minimizing $\langle\phi|G|\phi\rangle + \langle\phi|H'_{\text{prop}}|\phi\rangle$ is a balancing act between "not upsetting" either G or H'_{prop} too much. Intuitively, the "minimum joint unhappiness" experienced by G and H'_{prop} relative to a state $|\phi\rangle$ should depend on "how close" their null spaces are — if there is a $|\phi\rangle$ that is "close" to both $\text{Null}(G)$ and $\text{Null}(H'_{\text{prop}})$, then we might expect to minimize $\langle\phi|G|\phi\rangle + \langle\phi|H'_{\text{prop}}|\phi\rangle$ "well". Conversely, if $\text{Null}(G)$ and $\text{Null}(H'_{\text{prop}})$ are "far", then we might expect $\lambda_{\min}(G + H'_{\text{prop}})$ to be "large". This is precisely the intuition captured by the Geometric Lemma (whose proof uses elementary Linear Algebra, and which we omit for brevity).

Lemma 6.57 (Geometric Lemma)**.** *Let $A_1, A_2 \succeq 0$, and let v lower bound the minimum* non-zero *eigenvalues of both A_1 and A_2. Then,*

$$\lambda_{\min}(A_1 + A_2) \geq 2v \sin^2 \frac{\angle(\text{Null}(A_1), \text{Null}(A_2))}{2},$$

where the angle *between spaces $\mathcal{X}$ and $\mathcal{Y}$ is defined as* $\angle(\mathcal{X}, \mathcal{Y}) := \arccos\left[\max_{\substack{|x\rangle\in\mathcal{X}, |y\rangle\in\mathcal{Y} \\ \||x\rangle\|_2 = \||y\rangle\|_2 = 1}} |\langle x|y\rangle|\right]$.

Thus, the correct notion of "closeness" for $\text{Null}(G)$ and $\text{Null}(H'_{\text{prop}})$ is the *angle* between the two spaces.

Exercise 6.58. Suppose $\mathrm{Null}(A_1) \cap \mathrm{Null}(A_2) \neq \emptyset$. Why is the lower bound given by the Geometric Lemma trivial in this case, and why is this expected?

We now have a clear approach for trying to show $\lambda_{\min}(G + H'_{\mathrm{prop}}) \geq \beta$ — apply the Geometric Lemma with $A_1 = G$ and $A_2 = H'_{\mathrm{prop}}$. It just remains to figure out v and $\angle(\mathrm{Null}(G), \mathrm{Null}(H'_{\mathrm{prop}}))$.

Exercise 6.59. Prove that the smallest non-zero eigenvalue of G is 1. (Hint: Recall H_{in} and H_{out} are commuting projectors.)

Exercise 6.60. Use the formula for the eigenvalues of Λ to show that the smallest non-zero eigenvalue of H'_{prop} is $2(1 - \cos(\pi/(m+1))) \geq \pi^2/(m+1)^2$. (Hint: Taylor series.)

By the two exercises above, we may set $v = \pi^2/(m+1)^2$. The next lemma suffices to finish the proof of Lemma 6.40.

Lemma 6.61. *It holds that*

$$\sin^2 \frac{\angle(\mathrm{Null}(G), \mathrm{Null}(H'_{\mathrm{prop}}))}{2} \geq \frac{1 - \sqrt{\epsilon}}{4(m+1)}.$$

Proof. Since $\sin^2 x = 1 - \cos^2 x$, it suffices to show the bound

$$\cos^2 \angle(\mathrm{Null}(G), \mathrm{Null}(H'_{\mathrm{prop}})) \leq 1 - \frac{1 - \sqrt{\epsilon}}{m+1}. \tag{6.2}$$

Exercise 6.62. Prove that Lemma 6.61 follows from Equation (6.2).

To show Equation (6.2), we must upper bound

$$\max_{\substack{|x\rangle \in \mathrm{Null}(G), |y\rangle \in \mathrm{Null}(H'_{\mathrm{prop}}) \\ \| |x\rangle \|_2 = \| |y\rangle \|_2 = 1}} |\langle x|y\rangle|^2 = \max_{\substack{|x\rangle \in \mathrm{Null}(G), |y\rangle \in \mathrm{Null}(H'_{\mathrm{prop}}) \\ \| |x\rangle \|_2 = \| |y\rangle \|_2 = 1}} \langle y|x\rangle\langle x|y\rangle = \max_{\substack{|y\rangle \in \mathrm{Null}(H'_{\mathrm{prop}}) \\ \| |y\rangle \|_2 = 1}} \langle y|\Pi_{\mathrm{Null}(G)}|y\rangle,$$

for $\Pi_{\mathcal{X}}$ the projector onto a space $\mathcal{X}$. Since $\mathrm{Null}(G) = N_1 \oplus N_2 \oplus N_3$, we may write $\Pi_{\mathrm{Null}(G)} = \Pi_{N_1} + \Pi_{N_2} + \Pi_{N_3}$. Handling Π_{N_2} can be done directly.

Exercise 6.63. Prove that $\max_{\substack{|y\rangle \in \mathrm{Null}(H'_{\mathrm{prop}}) \\ \| |y\rangle \|_2 = 1}} \langle y|\Pi_{N_2}|y\rangle = \frac{m-1}{m+1}$.

To next relate our bound to ϵ, which encodes the probability of acceptance, we must consider $\Pi_{N_1} + \Pi_{N_3}$ jointly. Defining $\mathcal{X} := |x\rangle_A \otimes (\mathbb{C}^2)_B^{\otimes p(n)} \otimes |0 \cdots 0\rangle_C$ and $\mathcal{Y} := \{|\eta\rangle \mid V|\eta\rangle \text{ has qubit } C_1 \text{ set to } |1\rangle\}_{A,B,C}$, and recalling that any $|y\rangle \in \mathrm{Null}(H_{\mathrm{prop}})$ has form $|\eta\rangle_{A,B,C} \otimes |\gamma\rangle_D$, we have

$$\langle y|\Pi_{N_1} + \Pi_{N_3}|y\rangle = \langle y|(\Pi_{\mathcal{X}} \otimes |0\rangle\langle 0|_D + \Pi_{\mathcal{Y}} \otimes |m\rangle\langle m|_D)|y\rangle = \frac{1}{m+1}\langle \eta|(\Pi_{\mathcal{X}} + \Pi_{\mathcal{Y}})|\eta\rangle.$$

In other words, maximizing the left hand side has been reduced to upper bounding $\lambda_{\max}(\Pi_{\mathcal{X}} + \Pi_{\mathcal{Y}})$. Luckily, the main statement derived in the proof of the Geometric Lemma (which, recall, we've omitted) gives us exactly the requisite tool for this.

Fact 6.64. *For spaces $\mathcal{A}$ and $\mathcal{B}$, $\lambda_{\max}(\Pi_{\mathcal{A}} + \Pi_{\mathcal{B}}) \leq 1 + \cos(\angle(\mathcal{A}, \mathcal{B}))$.*

Exercise 6.65. Use Fact 6.64 to prove that $\cos^2(\angle(\mathcal{X}, \mathcal{Y})) \leq \epsilon$. Conclude that $\langle y|\Pi_{N_1} + \Pi_{N_3}|y\rangle \leq \frac{1+\sqrt{\epsilon}}{m+1}$.

Exercise 6.66. Combine the bounds we have computed to conclude that Equation (6.2) holds, thus completing the proof of Lemma 6.61. □

Exercise 6.67. Combine our choice of v and Lemma 6.61 to obtain Lemma 6.40.

□

7 Quantum-Classical Merlin Arthur (QCMA) and Ground State Connectivity

"I have called this principle, by which each slight variation, if useful, is preserved, by the term of Natural Selection."
— Charles Darwin

Introduction. In Lecture 5, we introduced Quantum Merlin Arthur (QMA) as the *de facto* quantum generalization of NP, which verified a quantum proof $|\psi\rangle$ with a quantum verifier. It is not clear at all, however, whether a *quantum* proof is required to capture the full power of QMA. For even though an arbitrary quantum proof $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ can be a "complicated" quantum state, a QMA verifier is restricted to be a *polynomial-size* quantum circuit. Can such a limited verifier even "distinguish" between "complicated" proofs $|\psi\rangle$ and "simpler" approximations $|\widetilde{\psi}\rangle$ thereof (where by "simpler" we roughly mean that unlike $|\psi\rangle$, $|\widetilde{\psi}\rangle$ has a succinct classical description)? In other words, is a classical proof as good as a quantum one for the purposes of polynomial-time quantum verification?

In this lecture, we explore this question via the complexity class QCMA, which is QMA but with a classical proof. Whereas QCMA $\subseteq$ QMA holds trivially, it is not at all clear whether the reverse containment should hold. In other words, in line with the opening quote of this lecture, we do not yet know whether the "variation" of allowing proofs to be quantum yields something "useful" (meaning more verification power). In this sense, "Natural Selection" is yet to play its hand in the study of "quantum NP".

This lecture is organized as follows. We begin in Section 7.1 by defining QCMA. Section 7.2 studies the QCMA-complete problem of *Ground State Connectivity*, which gives some insight into the types of classical proofs which may be useful to quantum verifiers. The proof of QCMA-completeness (Section 7.2.1) will again utilize the history state construction of the previous lecture, and its soundness analysis requires a tool known as the Traversal Lemma (Section 7.2.1). The tightness of this lemma is discussed in the closing section of this lecture, Section 7.2.1.

7.1 Quantum-Classical Merlin Arthur (QCMA)

Sandwiched between PromiseMA and QMA is QCMA (more accurately, PromiseQCMA) (i.e. PromiseMA $\subseteq$ QCMA $\subseteq$ QMA), a natural complexity class defined as follows.

Definition 7.1 (Quantum-Classical Merlin Arthur (QCMA))**.** *A promise problem $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in QCMA if there exists a P-uniform quantum circuit family $\{Q_n\}$ and polynomials $p, q : \mathbb{N} \to \mathbb{N}$ satisfying the following properties. For any input $x \in \{0,1\}^n$, Q_n takes in $n + p(n) + q(n)$ qubits as input, consisting of the input x on register A, $p(n)$ qubits initialized to a "classical proof" $|y\rangle \in \{0,1\}^{p(n)}$ on register B, and $q(n)$ ancilla qubits initialized to $|0\rangle$ on register C. The first qubit of register C, denoted C_1, is the designated output qubit, a measurement of which in the standard basis after applying Q_n yields the following:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, there exists proof $y \in \{0,1\}^{p(n)}$, such that Q_n accepts with probability at least $2/3$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then for all proofs $y \in \{0,1\}^{p(n)}$, Q_n accepts with probability at most $1/3$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, Q_n may accept or reject arbitrarily.*

Exercise 7.2. What is the only difference between QMA and QCMA?

As with BQP and QMA, the completeness and soundness errors can be made exponentially small via parallel repetition of the verification protocol and a majority vote. However, as with PromiseMA, it turns out that without loss of generality, one may assume QCMA has *perfect* completeness, i.e. in the YES case, there exists a proof y accepted with certainty. An analogous statement for QMA remains an open question.

Exercise 7.3. For QMA, we distinguished between weak and strong error reduction. Why does strong error reduction for QCMA hold trivially?

Exercise 7.4. Note that the proof register B in Definition 7.1 is expected to contain a standard basis state $|y\rangle$ for $y \in \{0,1\}^{p(n)}$. Since Q_n is a *quantum* circuit, B is a register of $p(n)$ *qubits*. This means that in the NO case, a cheating prover can in principle send an arbitrary *entangled* state $|\psi\rangle$ in register B. Why does this not ruin the soundness property of Definition 7.1? (Hint: Without loss of generality, we may assume that before running the actual verification, Q_n makes a certain measurement. Which measurement should Q_n make, and how can Q_n simulate this measurement unitarily?)

7.2 Ground State Connectivity

We now study a physically motivated complete problem for QCMA, which gives some insight into what type of *classical* proof might be useful to a *quantum* verifier. As with the Quantum Cook-Levin theorem, the problem arises in the setting of ground space properties of local Hamiltonians, $H = \sum_i H_i$. In contrast to k-LH, however, we shift our attention away from the *ground state energy* of H and to the structural properties of the *ground space* of H. For inspiration, we look to the classical study of *reconfiguration problems*.

Reconfiguration problems. Given a 3-SAT formula $\phi : \{0,1\}^n \to \{0,1\}$, computing different properties of ϕ can have different complexities. For example, we know from the Cook-Levin theorem that deciding whether the solution space for ϕ is *non-empty* (i.e. does ϕ have a satisfying assignment?) is NP-complete. If we instead wish to count the *size* of the solution space (i.e. the number of satisfying assignments to ϕ), this is much harder; it is #P-complete. We may also ask about the *structure* of the solution space — for example, is it *connected*? This turns out to be either in P, NP-complete or PSPACE-complete, depending on how the question is phrased.

Let us formalize what we mean by "connected" (we state it using ket notation to highlight the generalization to the quantum setting later). Given as input a Boolean formula $\phi : \{0,1\}^n \to \{0,1\}$, two satisfying assignments $x, y \in \{0,1\}^n$, and length parameter 1^m for $m \in \mathbb{N}$, we say $|x\rangle$ and $|y\rangle$ are *connected* with respect to ϕ if there exists a sequence of length at most $N \leq m$

bit flips $(X_{i_1}, X_{i_2}, \ldots, X_{i_N})$ for $i_k \in [m]$ (where Pauli X_i is applied to qubit i) satisfying two properties:

1. (Intermediate states are in solution space) For all $k \in [N]$ and intermediate states $|x_k\rangle := X_{i_k} \cdots X_{i_1}|x\rangle$, $\phi(x_k) = 1$.
2. (Final state is target state) $X_{i_N} \cdots X_{i_1}|x\rangle = |y\rangle$.

In other words, is there a sequence of at most m bit flips we can apply to map x to y, such that each intermediate state attained is also a solution to ϕ?

Exercise 7.5. Let $\phi = (x_1 \vee \overline{x_2})$ and $x = 00$ and $y = 11$. Are x and y connected with respect to ϕ?

Exercise 7.6. Give a Boolean formula ϕ (not necessarily in CNF form) and solutions x, y which are *not* connected with respect to ϕ.

It is important to note that the number of bit flips m needed for mapping x to y in this manner need *not* be at most n; in fact, it can scale as $O(2^n)$. This is due to property 1 above; the naive greedy sequence of bit flips mapping x to y might take us temporarily *out* of the solution space. For this reason, if we drop the upper bound m in the input, the problem of determining if a 3-SAT formula is connected is PSPACE-complete. In the formulation above (i.e. with parameter m), however, the problem is NP-complete.

Exercise 7.7. Why is deciding if a 3-SAT formula is connected according to our definition above in NP?

Reconfiguration in the quantum setting. By generalizing the reconfiguration problem for 3-SAT to the quantum setting, we arrive at the main problem to be studied in this section. As before, we are interested in the structure of the solution space, where "solution space" now refers to the ground space of a local Hamiltonian.

Definition 7.8 (Ground State Connectivity (GSCON))**.** *Fix an inverse polynomial $\Delta : \mathbb{N} \to \mathbb{R}^+$.*

- *Input parameters:*
  1. *k-local Hamiltonian $H = \sum_i H_i$ acting on n qubits with $H_i \in \mathrm{Herm}(\mathbb{C}^2)^{\otimes k}$ satisfying $\| H_i \|_\infty \leq 1$.*
  2. *Thresholds $\eta_1, \eta_2, \eta_3, \eta_4 \in \mathbb{R}$ such that $\eta_2 - \eta_1 \geq \Delta$ and $\eta_4 - \eta_3 \geq \Delta$, and 1^m for $m \in \mathbb{N}$.*
  3. *Polynomial size quantum circuits U_ψ and U_ϕ generating "starting" and "target" states $|\psi\rangle$ and $|\phi\rangle$ (starting from $|0\rangle^{\otimes n}$), respectively, satisfying $\langle\psi|H|\psi\rangle \leq \eta_1$ and $\langle\phi|H|\phi\rangle \leq \eta_1$.*
- *Output:*
  1. *If there exists a sequence of 2-qubit unitaries $(U_i)_{i=1}^m \in \mathrm{U}\left(\mathbb{C}^2\right)^{\times m}$ such that:*
     a) *(Intermediate states remain in low energy space) For all $i \in [m]$ and intermediate states $|\psi_i\rangle := U_i \cdots U_2 U_1 |\psi\rangle$, one has $\langle\psi_i|H|\psi_i\rangle \leq \eta_1$, and*

b) (Final state close to target state) $\| U_m \cdots U_1 |\psi\rangle - |\phi\rangle \|_2 \leq \eta_3$,

then output YES.

2. *If for all 2-qubit sequences of unitaries* $(U_i)_{i=1}^m \in \mathrm{U}\left(\mathbb{C}^2\right)^{\times m}$, *either:*

   a) (Intermediate state obtains high energy) There exists $i \in [m]$ *and an intermediate state* $|\psi_i\rangle := U_i \cdots U_2 U_1 |\psi\rangle$, *such that* $\langle\psi_i|H|\psi_i\rangle \geq \eta_2$, *or*

   b) (Final state far from target state) $\| U_m \cdots U_1 |\psi\rangle - |\phi\rangle \|_2 \geq \eta_4$,

   then output NO.

Intuition. Roughly, GSCON says: Given two ground states[1] $|\psi\rangle$ and $|\phi\rangle$ of a k-local Hamiltonian H, are $|\psi\rangle$ and $|\phi\rangle$ *connected* through the ground space of H? In other words, is there a sequence of 2-qubit gates mapping $|\psi\rangle$ to $|\phi\rangle$, such that all intermediate states encountered are also ground states? This turns out to have an important physical motivation in the quantum setting — it asks whether the ground space of H has an *energy barrier* preventing one ground state from being mapped to another via short circuits (while remaining in the low energy space throughout the computation).

7.2.1 QCMA-completeness of GSCON

Theorem 7.9. *There exists a polynomial* r *such that* GSCON *is* QCMA*-complete for* $m \in O(r(n))$ *and* $k \geq 7$, *where* n *denotes the number of qubits* H *acts on.*

Proof. For brevity, we sketch containment in QCMA. We give a full proof of QCMA-hardness.

Containment in QCMA. Containment in QCMA holds for any $m \in O(\mathrm{poly}(n))$, and is intuitively straightforward (hence we only sketch it here) — the prover sends[2] a classical description of the polynomially many 2-qubit gates $U_1, \ldots, U_m$ to the verifier as a proof. Since the verifier can prepare both the start and final states $|\psi\rangle$ and $|\phi\rangle$ efficiently via U_ψ and U_ϕ (also given as input), it can check the energy of each intermediate state $|\psi_i\rangle$ against H using the protocol from the proof of the Cook-Levin theorem. Finally, to check if $|\psi_m\rangle \approx |\phi\rangle$, the verifier prepares both states and applies the SWAP test, which we now briefly discuss, as it is a useful tool to have in one's toolkit.

Swap test. Given physical copies of states $|\psi\rangle$ and $|\phi\rangle$, can we efficiently test if $|\psi\rangle \approx |\phi\rangle$? The answer is *yes*, and goes via the SWAP test, pictured in Figure 7.1. This test outputs 0 with probability $(1 + |\langle\psi|\phi\rangle|^2)/2$, thus allowing us to estimate $\| |\psi\rangle - |\phi\rangle \|_2$.

QCMA-hardness. Let $A = (A_{\mathrm{yes}}, A_{\mathrm{no}}, A_{\mathrm{inv}})$ denote a QCMA promise problem. Let $x \in \{0,1\}^n$ be an input, with corresponding QCMA verifier $V = V_N \cdots V_1 \in (\mathbb{C}^2)^{\otimes n} \otimes (\mathbb{C}^2)^{\otimes p(n)} \otimes (\mathbb{C}^2)^{\otimes q(n)}$. Recall V is a uniformly generated quantum verification circuit consisting of 1- and 2-qubit unitary gates, acting on registers A (n qubits containing the input x), B ($p(n)$ qubits containing the proof $|\psi\rangle$), and C ($q(n)$ ancilla qubits initialized to all zeroes). We make two assumptions about V without loss of generality: (1) The completeness and soundness parameters for V are

[1] More accurately, the definition of GSCON discusses low-energy states, which need not be ground states. However, by using the fact that QCMA satisfies perfect completeness without loss of generality, one can show QCMA-completeness of GSCON even when all states involved are ground states of H.

[2] More accurately, since arbitrary 2-qubit gates cannot be specified exactly to finite precision, the prover sends elements from an appropriate notion of an "ϵ-net" over 2-qubit unitaries.

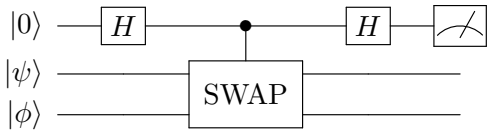


Figure 7.1: The circuit for the SWAP test. The SWAP gate has action $|x\rangle|y\rangle \mapsto |y\rangle|x\rangle$ for any standard basis states $|x\rangle, |y\rangle$. Note the inputs $|\psi\rangle$ and $|\phi\rangle$ are in tensor product. The "output" is the measurement result on the first wire.

$1-\epsilon$ and ϵ, so that $1-2\epsilon \in \Omega(1/\operatorname{poly}(n))$, and (2) V begins by measuring its proof in the standard basis (this trick was alluded to in a previous exercise; it forces a cheating prover to send a classical proof, and can be simulated unitarily via the principle of deferred measurement). Our goal is to construct an instance $(H, \eta_1, \eta_2, \eta_3\,\eta_4, m, U_\psi, U_\phi)$ with 7-local Hamiltonian H such that, if $x \in A_{\text{yes}}$, then there exists a sequence $U_1, \ldots, U_m$ of 2-qubit gates mapping $|\psi\rangle$ to $|\phi\rangle$ through the low energy space of H, and if $x \in A_{\text{no}}$, then any such unitary sequence must either leave the low-energy space of H at some point or map $|\psi\rangle$ "far" from $|\phi\rangle$.

The construction. We now state the construction, which at first glance does not seem to do anything interesting. The YES case in the correctness analysis will reveal the intuition as to why this works.

Let H^{CL} denote the 5-local Hamiltonian obtained from V the Quantum Cook-Levin Theorem's circuit-to-Hamiltonian construction. We define H to act on a *Hamiltonian* register denoted h (consisting of subregisters A, B, C, and D, where D is the unary-encoded clock register) and 3-qubit *GO* register denoted G. Specifically, $H \in \operatorname{Herm}(\mathbb{C}^2)_h^{\otimes(n+p(n)+q(n)+N)} \otimes (\mathbb{C}^2)_G^{\otimes 3}$.

Exercise 7.10. What are the N qubits in the h register used for? Why are there N of them?

Define

$$H := H_h^{\text{CL}} \otimes P_G \qquad \text{for} \qquad P := I - |000\rangle\langle 000| - |111\rangle\langle 111|. \tag{7.1}$$

Exercise 7.11. Note that P as written is 3-local. Show how to write P equivalently as a 2-local Hamiltonian.

By the exercise above, we have that H is 7-local. Define the initial and final states as

$$|\psi\rangle := |0\rangle^{\otimes(n+p(n)+q(n)+N)}|0\rangle^{\otimes 3} \qquad \text{and} \qquad |\phi\rangle := |0\rangle^{\otimes(n+p(n)+q(n)+N)}|1\rangle^{\otimes 3}, \tag{7.2}$$

which have trivial poly-size preparation circuits U_ψ and U_ϕ. Finally, let W denote a unitary circuit of size $|W|$ which prepares the history state of H given classical proof y. Define $m := 2(p(n) + |W| + 1)$.

To complete the construction, set $\eta_3 = 0$, $\eta_4 = 1/4$, $\eta_1 = \alpha$, and $\eta_2 = \beta/(16m^2)$, where $\alpha := \epsilon/(N+1)$ and $\beta := \pi^2(1-\sqrt{\epsilon})/(2(N+1)^3)$ come from the Quantum Cook-Levin theorem's circuit-to-Hamiltonian construction. Note that if we apply weak error reduction to V so that ϵ is exponentially close to zero, then the gap $\Delta \in \Omega(1/N^5)$.

Correctness for YES case. Suppose $x \in A_{\text{yes}}$, i.e. there exists a proof $y \in \{0,1\}^{p(n)}$ accepted by V. We demonstrate a sequence $(U_i)_{i=1}^m$ of 2-qubit unitaries mapping $|\psi\rangle$ to $|\phi\rangle$ through the ground space of H.

Intuition. To see the intuition, we first need an exercise.

Exercise 7.12. Prove that $|\psi\rangle$ and $|\phi\rangle$ lie in the null space of H, i.e. $H|\psi\rangle = H|\phi\rangle$. Use the fact that $H^{\text{CL}} \succeq 0$ to conclude that $|\psi\rangle$ and $|\phi\rangle$ hence lie in the ground space of H, and in particular have energy at most α.

Thus, $|\psi\rangle$ and $|\phi\rangle$ both lie in the null space of H, and are identical except for the pesky 3 qubits in the GO register, which are set to $|000\rangle$ and $|111\rangle$, respectively. To map $|\psi\rangle$ to $|\phi\rangle$ via 2-local gates, the obvious idea is hence to flip the GO qubits from 000 to 111. The problem is that we cannot flip more than two qubits at a time — so after flipping (say) the first two GO qubits, the G register reads $|110\rangle$, and now we are in the support of P_G in the definition of H. This means that H^{CL} is now "turned on" and checks the h register for a history state. Since we are in the YES case, there *is* a good history state we can use, and moreover, since we are dealing with QCMA, this history state can be prepared from the all zeroes initial state via a polynomial-size (though not necessarily P-uniformly generated) circuit.

Exercise 7.13. Given a classical proof $y \in \{0,1\}^{p(n)}$, give a polynomial-size sequence of 2-qubit gates which maps the all-zeroes state to the history state $|\psi_{\text{hist}}\rangle$ in time polynomial in n. Why does this not necessarily work for QMA, i.e. given a copy of a quantum proof $|\eta\rangle \in (\mathbb{C}^2)^{\otimes p(n)}$ in place of y?

The honest prover's actions. Recall the h register is broken up into four subregisters, A (input), B (proof), C (ancilla), and D (clock). The sequence of 2-qubit gates $(U_i)_{i=1}^m$ is as follows:

1. Apply Pauli X gates to h_B to prepare classical proof y, i.e., map $|0\rangle^{\otimes p(n)}$ to $|y\rangle$.
2. Apply W to h to prepare the history state $|\text{hist}_y\rangle$ of H^{CL}.
3. Apply $(X \otimes X \otimes I)_G$ to "initiate" checking of $|\text{hist}_y\rangle$.
4. Apply $(I \otimes I \otimes X)_G$ to "complete" checking of $|\text{hist}_y\rangle$.
5. Apply $W^\dagger$ to h to uncompute $|\text{hist}_y\rangle$.
6. Apply X gates to h_B to map the initial proof $|y\rangle$ back to $|0\rangle^{\otimes p(n)}$.

Exercise 7.14. Why is the length of the sequence above at most $m = 2(p(n) + |W| + 1)$, as desired?

Exercise 7.15. Verify that the sequence (U_i) correctly maps $|\psi\rangle$ to $|\phi\rangle$.

Exercise 7.16. As in Definition 7.8, recall the ith intermediate state is defined $|\psi_i\rangle := U_i \cdots U_1|\psi\rangle$. There is precise only one $i \in [6]$ above such that $|\psi_i\rangle$ is *not* in the null space of H — which i is this? Prove that for this i, $\langle\psi_i|H|\psi_i\rangle \leq \eta_1$, as desired.

Proof of correctness for NO case. Suppose $x \in A_{\text{no}}$, i.e. all proofs $y \in \{0,1\}^{p(n)}$ are rejected by V with probability at least $1-\epsilon$. Intuitively, we cannot proceed as in the YES case now because H^{CL} does *not* have a low-energy history state (indeed, all its eigenvalues are at least β). Thus, the moment we "switch on" the H^{CL} check in Step 3, we are in trouble. The only way a cheating prover can try to bypass this problem is to somehow try to switch all GO qubits from 000 to 111 *without* having significant support on the orthogonal space spanned by $\{|001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle\}$; note that this statement is non-trivial because the prover is not restricted to simply performing Pauli X gates. Thus, our main task for the NO case is to prove that this is impossible. The main tool for this is the following lemma, whose proof is given in Section 7.2.1.

Lemma 7.17 (Traversal Lemma)**.** *Let $S, T \subseteq (\mathbb{C}^2)^{\otimes n}$ be k-orthogonal subspaces. Fix arbitrary states $|v\rangle \in S$ and $|w\rangle \in T$, and consider a sequence of k-qubit unitaries $(U_i)_{i=1}^m$ such that*

$$\| \, |w\rangle - U_m \cdots U_1 |v\rangle \, \|_2 \leq \delta$$

for some $0 \leq \delta < 1/2$. Define $|v_i\rangle := U_i \cdots U_1 |v\rangle$ and $P := I - \Pi_S - \Pi_T$. Then, there exists $i \in [m]$ such that

$$\langle v_i | P | v_i \rangle \geq \left(\frac{1-2\delta}{2m} \right)^2.$$

Intuitively, this lemma says basically what we are looking for — that for certain types of subspaces S and T, any local unitary mapping from S to T must at some point "leave" $S \oplus T$. The "type" of subspaces this applies to are defined next.

Definition 7.18 (k-orthogonal states and subspaces)**.** *For $k \geq 1$, a pair of states $|v\rangle, |w\rangle \in (\mathbb{C}^d)^{\otimes n}$ is* k-orthogonal *if for all k-qubit unitaries U, we have $\langle w|U|v\rangle = 0$. We call subspaces $S, T \subseteq (\mathbb{C}^2)^{\otimes n}$* k-orthogonal *if any pair of vectors $|v\rangle \in S$ and $|w\rangle \in T$ are k-orthogonal.*

Exercise 7.19. Prove that $|000\rangle$ and $|111\rangle$ are 2-orthogonal. Are they 3-orthogonal?

We can now complete the proof of correctness for the NO case. We know the smallest eigenvalue of H^{CL} is at least β. Let S and T denote the $+1$ eigenspaces of $I_h \otimes |000\rangle\langle 000|_G$ and $I_h \otimes |111\rangle\langle 111|_G$, respectively.

Exercise 7.20. Characterize S and T.

Exercise 7.21. Show that S and T are 2-orthogonal subspaces, and that $|\psi\rangle \in S$ and $|\phi\rangle \in T$.

By the exercises above, for any sequence of two-qubit unitaries $(U_i)_{i=1}^m$, either $\| \, |\psi_m\rangle - |\phi\rangle \, \|_2 \geq 1/4 = \eta_4$ (in which case we have a NO instance of GSCON and we are done), or we can apply the Traversal Lemma (Lemma 7.17) with $\delta = 1/4$ to conclude that there exists an $i \in [m]$ such that

$$\langle \psi_i | P' | \psi_i \rangle \geq \left(\frac{1}{4m} \right)^2 = \frac{\eta_2}{\beta},$$

where recall $|\psi_i\rangle := U_i \cdots U_1 |\psi\rangle$ and we define $P' = I - \Pi_S - \Pi_T = I_h \otimes P$. We conclude that

$$\langle \psi_i | H | \psi_i \rangle = \langle \psi_i | H^{\text{CL}} \otimes P | \psi_i \rangle \geq \beta \langle \psi_i | I_h \otimes P | \psi_i \rangle = \beta \langle \psi_i | P' | \psi_i \rangle \geq \eta_2,$$

where the first inequality follows since $H^{\text{CL}} \succeq \beta I$.

□

Proof of Traversal Lemma

To conclude the proof of Theorem 7.9, it remains to show the Traversal Lemma, which we reproduce below for convenience.

Lemma 7.17. *Let $S, T \subseteq (\mathbb{C}^2)^{\otimes n}$ be k-orthogonal subspaces. Fix arbitrary states $|v\rangle \in S$ and $|w\rangle \in T$, and consider a sequence of k-qubit unitaries $(U_i)_{i=1}^m$ such that*

$$\| \, |w\rangle - U_m \cdots U_1 |v\rangle \, \|_2 \leq \delta$$

for some $0 \leq \delta < 1/2$. Define $|v_i\rangle := U_i \cdots U_1 |v\rangle$ and $P := I - \Pi_S - \Pi_T$. Then, there exists $i \in [m]$ such that

$$\langle v_i | P | v_i \rangle \geq \left(\frac{1 - 2\delta}{2m} \right)^2 .$$

The proof of Lemma 7.17 requires the well-known "Gentle Measurement Lemma", which quantifies an intuitively expected behavior: If a measurement outcome Π has high probability of occurring for state ρ, then we expect the postmeasurement state (proportional to) $\Pi\rho\Pi$ to be approximately ρ (i.e. the measurement should not disturb the state much). We state this lemma below first, and then prove Lemma 7.17.

Lemma 7.22 (Gentle Measurement Lemma)**.** *Let $\rho \in \mathrm{L}\left(\mathbb{C}^d\right)$ be a density operator and $O \preceq \Pi \preceq I$ a projective measurement operator for $\Pi \in \mathrm{L}\left(\mathbb{C}^d\right)$, such that $\mathrm{Tr}(\Pi\rho) \geq 1 - \epsilon$. Then, $\| \, \rho - \Pi\rho\Pi \, \|_{\mathrm{tr}} \leq 2\sqrt{\epsilon}$.*

Note that $\Pi\rho\Pi$ above is *not* necessarily normalized.

Proof of Lemma 7.17. We give a proof by contradiction. Suppose that for all $i \in [m]$, the expectations satisfy $\langle v_i | P | v_i \rangle < \kappa := [(1 - 2\delta)/(2m)]^2$. Consider the following thought experiment inspired by the quantum Zeno effect[3]. Imagine that after each U_i is applied, we measure $|v_i\rangle$ using the projective measurement $(\Pi, I - \Pi)$ for $\Pi := I - P$, and postselect on obtaining outcome Π. Define the following two sequences:

- $|v_i'\rangle := \Pi |v_i\rangle$ for $i \in [m]$,
- $|v_1''\rangle := |v_1'\rangle$ and $|v_i''\rangle := \Pi \, U_i |v_{i-1}''\rangle$ for $i \in \{2, \ldots, m\}$.

Note that $|v_i'\rangle$ and $|v_i''\rangle$ are not necessarily normalized.

To set up our contradiction, we first prove by induction on i that

$$\left\| \, |v_i\rangle\langle v_i| - |v_i''\rangle\langle v_i''| \, \right\|_{\mathrm{tr}} < 2i\sqrt{\kappa}. \tag{7.3}$$

For the base case $i = 1$, we have $|v_1''\rangle = |v_1'\rangle$. Then, since $\langle v_1 | P | v_1 \rangle < \kappa$, we know that $\mathrm{Tr}(\Pi |v_1\rangle\langle v_1|) > 1 - \kappa$.

[3] Roughly, the quantum Zeno effect is the phenomenon that a quantum system which is continuously observed never evolves.

Exercise 7.23. Use the Gentle Measurement Lemma (Lemma 7.22) to conclude

$$\left\| |v_1\rangle\langle v_1| - |v_1''\rangle\langle v_1''| \right\|_{\mathrm{tr}} < 2\sqrt{\kappa}, \tag{7.4}$$

as required for the base case.

For the inductive case, assume Equation (7.3) holds for $1 \leq i \leq j-1$. We prove it holds for $i = j$. Specifically,

$$\begin{aligned}
\left\| |v_j\rangle\langle v_j| - |v_j''\rangle\langle v_j''| \right\|_{\mathrm{tr}} &\leq \left\| |v_j\rangle\langle v_j| - |v_j'\rangle\langle v_j'| \right\|_{\mathrm{tr}} + \left\| |v_j'\rangle\langle v_j'| - |v_j''\rangle\langle v_j''| \right\|_{\mathrm{tr}} \\
&< 2\sqrt{\kappa} + \left\| |v_j'\rangle\langle v_j'| - |v_j''\rangle\langle v_j''| \right\|_{\mathrm{tr}} \\
&= 2\sqrt{\kappa} + \left\| \Pi U_j \left(|v_{j-1}\rangle\langle v_{j-1}| - |v_{j-1}''\rangle\langle v_{j-1}''| \right) U_j^\dagger \Pi \right\|_{\mathrm{tr}} \\
&\leq 2\sqrt{\kappa} + \left\| |v_{j-1}\rangle\langle v_{j-1}| - |v_{j-1}''\rangle\langle v_{j-1}''| \right\|_{\mathrm{tr}} \\
&< 2\sqrt{\kappa} + 2(j-1)\sqrt{\kappa} \\
&= 2j\sqrt{\kappa},
\end{aligned} \tag{7.5}$$

where the first statement follows from the triangle inequality, the second from the Gentle Measurement Lemma, the fourth from the facts that the Schatten p-norms are invariant under isometries and that $\| ABC \|_p \leq \| A \|_\infty \| B \|_p \| C \|_\infty$, and the fifth from the induction hypothesis. This establishes Equality (7.3).

Exercise 7.24. Use the fact that $\| |v\rangle\langle v| - |w\rangle\langle w| \|_{\mathrm{tr}} \leq 2 \| |v\rangle - |w\rangle \|_2$ for unit vectors $|v\rangle, |w\rangle$ to conclude that

$$\left\| |v_m''\rangle\langle v_m''| - |w\rangle\langle w| \right\|_{\mathrm{tr}} < 1, \tag{7.6}$$

We are now ready to obtain the desired contradiction. To do so, observe that since $|v\rangle \in S$, and since S and T are k-orthogonal subspaces, we have that for all $i \in [m]$, $|v_i''\rangle \in S$ (i.e., if S is 1-dimensional, this is the Zeno effect). Thus, we have $\langle v_m''|w\rangle = 0$, implying that

$$\left\| |v_m''\rangle\langle v_m''| - |w\rangle\langle w| \right\|_{\mathrm{tr}} = 1 + \left\| |v_m''\rangle \right\|_2 \geq 1.$$

This contradicts Equation (7.6), as desired. □

Tightness of the Traversal Lemma

The proof of QCMA-completeness of GSCON relied crucially on the Traversal Lemma, and so it is natural to ask whether the lemma is tight. Namely, in Lemma 7.17, the lower bound on $\langle v_i|P|v_i\rangle$ scales as $\Theta(1/m^2)$ (for m the number of unitaries and for fixed δ). This suggests that one can better "avoid" the subspace P projects onto if one uses a longer sequence of local unitaries. Indeed, it turns out that, at least in some cases, this behavior is possible; thus, a dependence on m is necessary in Lemma 7.17.

Theorem 7.25. *We assume the notation of Lemma 7.17. Fix any $0 < \Delta < 1/2$, and consider 2-orthogonal states $|v\rangle = |000\rangle$ and $|w\rangle = |111\rangle$, with $P := I - |v\rangle\langle v| - |w\rangle\langle w|$. Then, there exists a sequence of m 2-local unitary operations mapping $|v\rangle$ to $|w\rangle$ through intermediate states $|v_i\rangle$, each of which satisfy $\langle v_i|P|v_i\rangle \leq \Delta$, and where $m \in O(1/\Delta^2)$.*

Proof intuition. We omit the proof for brevity, but the idea behind it is based on the following rough analogy: Suppose one wishes to map the point $(1, 1)$ (corresponding to $|000\rangle$) in the 2D Euclidean plane to $(-1, -1)$ (corresponding to $|111\rangle$) via a sequence of moves with the following two restrictions: (1) For each current point (x, y), the next move must leave precisely one of x or y invariant (analogous to 2-local unitaries acting on a 3-qubit state), and (2) the Euclidean distance between (x, y) and the line through $(1, 1)$ and $(-1, -1)$ never exceeds Δ (analogous to the overlap with P not exceeding Δ). In other words, we wish to stay close to a diagonal line while making only horizontal and vertical moves. This can be achieved by making a sequence of "small" moves resembling a "staircase". The smaller the size of each "step" in the staircase, the better we approximate the line, at the expense of requiring more moves (analogous to increasing the number of unitaries, m). This is the basic premise behind the proof of Theorem 7.25 — giving a formal proof takes some work, as the back-and-forth shuffling of amplitude with the application of each local gate U_i needs to be carefully managed.

8 Quantum Merlin Arthur with Unentangled Proofs (QMA(2))

"Nothing strengthens authority so much as silence."
— Leonardo da Vinci

NOTE: THIS CHAPTER REMAINS UNDER CONSTRUCTION

Introduction. In Chapter 7, we considered our first alternative definition of "quantum NP" beyond QMA, QCMA. QCMA was rather natural — simply use a classical proof in place of QMA's quantum proof. But there are other variations of QMA one can consider, not all of which are equally benign. Enter QMA(2), proposed in 2003 just a year after QCMA, and for which a quantum proof is promised to be in *tensor product* across a prespecified cut L vs R, i.e. $|\psi\rangle_A \otimes |\phi\rangle_B$. *A priori*, it is not at all clear where there is anything interesting here — after all, entanglement appears at the heart of the power of quantum computing, so why should forcing *unentanglement* be of use?

Yet, over the last two decades, concrete answers about QMA(2) have been few and far between. On the one hand, hints of its "potential greatness" appeared early on, with the surprising fact that QMA(2) can verify NP via just *logarithmic*-size unentangled proofs. On the other hand, establishing even basic properties for QMA(2), such as *weak* error reduction (i.e. via polynomially many copies of the proof), took major effort. In contrast, in the same timespan, QCMA was discovered to have a number of natural complete problems, perfect completeness, even hardness of approximation!

As alluded to by this chapter's opening quote, this relative silence regarding QMA(2) has seeded the belief that QMA(2) is indeed more powerful than QMA. In fact, the most "mainstream" bound known on QMA(2) remains, embarassingly, the trivial one:

$$\text{QCMA} \subseteq \text{QMA} \subseteq \text{QMA(2)} \subseteq \text{NEXP}. \tag{8.1}$$

Recall that this is in sharp contrast to QMA, which is in PP.

Organization. We begin in Section 8.1 by defining QMA(k), i.e. QMA(2) with k unentangled provers. Section 8.2 discusses the Product State test, from which one obtains both QMA(2) = QMA(k) and weak error reduction. In Section 8.3, we show how QMA(2) can verify NP via logarithmic-size unentangled proofs. Finally, in Section 8.4 we discuss upper bounds on QMA(2), leading to quantum generalizations of the Polynomial Time Hierarchy.

8.1 QMA with unentangled provers (QMA(k))

We begin by defining QMA(k), which has k unentangled provers.

Definition 8.1 (QMA with unentangled provers (QMA(k))). *A promise problem* $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ *is in* QMA(k) *if there exists a P-uniform quantum circuit family* $\{Q_n\}$ *and polynomials* $p, q : \mathbb{N} \to \mathbb{N}$ *satisfying the following properties. For any input* $x \in \{0,1\}^n$, Q_n *takes in* $n + k \cdot p(n) + q(n)$ *qubits as input, consisting of the input* x *on register* A, *quantum proof* $|\psi_1\rangle_{B_1} \otimes \cdots \otimes |\psi_k\rangle_{B_k} \in ((\mathbb{C}^2)^{\otimes p(n)})^{\otimes k}$ *on registers* $B_1 \otimes \cdots \otimes B_k$, *and* $q(n)$ *ancilla qubits initialized to* $|0\rangle$ *on register* C. *The first qubit of register* C, *denoted* C_1, *is the designated output qubit, a measurement of which in the standard basis after applying* Q_n *yields the following:*

- *(Completeness/YES case) If* $x \in A_{\text{yes}}$, *there exists proof* $|\psi_1\rangle_{B_1} \otimes \cdots \otimes |\psi_k\rangle_{B_k} \in ((\mathbb{C}^2)^{\otimes p(n)})^{\otimes k}$, *such that* Q_n *accepts with probability at least* $2/3$.
- *(Soundness/NO case) If* $x \in A_{\text{no}}$, *then for all proofs* $|\psi_1\rangle_{B_1} \otimes \cdots \otimes |\psi_k\rangle_{B_k} \in ((\mathbb{C}^2)^{\otimes p(n)})^{\otimes k}$, Q_n *accepts with probability at most* $1/3$.
- *(Invalid case) If* $x \in A_{\text{inv}}$, Q_n *may accept or reject arbitrarily.*

We assume $1 \leq k \leq \text{poly}(n)$, *and define* $B := B_1 \otimes \cdots \otimes B_k$.

Throughout this chapter, unless noted otherwise, we use the term "product state" to refer to proofs which are product in the sense of Definition 8.1.

Exercise 8.2. What is the only difference between QMA and QMA(k)?

Pure versus mixed proofs. For QMA proof systems, we saw that without loss of generality, the optimal proof is a pure state, i.e. allowing mixed state proofs adds no verification power. We now ask the same question for QMA(2), except we need to be slightly careful: *Which set of mixed states does one consider?* Since proofs for QMA(k) are product states, the natural choice is to consider all *probabilistic mixtures* of pure product states on $B_1 \otimes \cdots \otimes B_k$, i.e. the convex hull of pure product states. Formally, define the set of separable states

$$S_k := \left\{ \sum_j p_j |\psi_{j,1}\rangle\langle\psi_{j,1}|_{B_1} \otimes \cdots \otimes |\psi_{j,k}\rangle\langle\psi_{j,k}|_{B_k} \mid p_j \geq 0, \ \sum_j p_j = 1, \right.$$
$$\left. |\psi_{j,i}\rangle \text{ is a unit vector for } 1 \leq i \leq k \right\}.$$

Exercise 8.3. Prove that S_k is a compact set, i.e. is closed and bounded.

Exercise 8.4. What are the extreme points of S_k, i.e. elements of S_k which cannot be written as convex combinations of other elements of S_k?

Next, as done in Equation (5.2) for QMA, we may derive the POVM encoding the QMA(k) verifier's "accept" outcome on input x as

$$P_x := \langle x|_A \otimes I_B \otimes \langle 0 \cdots 0|_C Q_n^\dagger |1\rangle\langle 1|_{C_1} Q_n |x\rangle_A \otimes I_B \otimes |0 \cdots 0\rangle_C. \tag{8.2}$$

Thus, the "mixed state" version of QMA(k) encodes maximizing

$$\max_{\rho_{B_1 \otimes \cdots \otimes B_k} \in S_k} \text{Tr}(P_x \rho_{B_1 \otimes \cdots \otimes B_k}). \tag{8.3}$$

This is an optimization of a linear function, $\text{Tr}(P_x \rho_{B_1 \otimes \cdots \otimes B_k})$, over a compact set, S_k. Such optimizations are known to obtain their optimal value on an extreme point of the set, which for S_k is (Exercise 8.4) the set of pure products states in $B_1 \otimes \cdots \otimes B_k$. Thus, we conclude that mixed state proofs confer no additional power to QMA(k) proof systems.

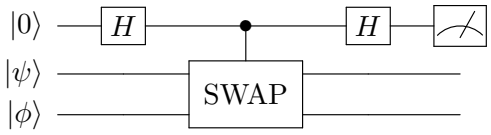


Figure 8.1: A reproduction of the SWAP test circuit from Figure 7.1, for convenience.

QMA versus QMA(k): A first hint at the latter's power. Deriving Equation (8.3) has a second benefit — recall that in contrast, for a QMA POVM P_x, the optimal proof maximizes (Equation (5.3))

$$\max_{\text{unit vectors } |\psi\rangle \in (\mathbb{C}^2)^{\otimes p(n)}} \langle \psi | P_x | \psi \rangle = \lambda_{\max}(P_x),$$

i.e. QMA is just an eigenvalue calculation for P_x. Of course, the catch is that the dimension of P_x is *exponential*. Nevertheless, given an explicit classical matrix encoding of P_x, one could compute $\lambda_{\max}(P_x)$ in time polynomial in the dimension. In contrast, a similar statement cannot be made for QMA(k) — the maximization of Equation (8.3) is *not* an eigenvalue problem. Such optimization problems (with P_x generalized by an arbitrary Hermitian matrix M, given explicitly classically as input) fall under the Best Separable State (BSS) problem, which is known to be NP-hard to within inverse polynomial (with respect to dimension) additive error, even for $k = 2$. In other words, whereas QMA encodes an efficiently solvable (with respect to dimension) eigenvalue optimization, QMA(2) encodes an NP-hard optimization problem. This is our first clue that perhaps there is more than meets the eye to QMA(k).

8.2 The product state test: QMA(2) and weak error reduction

With a formal definition of QMA(k) in hand, we turn to the central question: *What can we do with unentanglement, that we cannot do with entanglement?* The answer is — the SWAP test, introduced in Chapter 7, and which we begin by revisiting below. We then use the SWAP test to build the Product State test, one of the central tools in the QMA(2) toolkit, in Section 8.2.1. Section 8.2.2 discusses how QMA(k) = QMA(2) and weak error reduction follow from the Product State test.

Revisiting the SWAP test. For brevity, define $\mathcal{B} := (\mathbb{C}^2)^{\otimes n}$. In Chapter 7, we saw that given a pair of states in tensor product, $|\psi\rangle_A \otimes |\phi\rangle_B \in \mathcal{B} \otimes \mathcal{B}$, the SWAP test (Figure 8.1) outputs 0 on its first wire with probability $(1 + |\langle \psi | \phi \rangle|^2)/2$. By requesting multiple copies of $|\psi\rangle_A |\phi\rangle_B$, this allowed us to estimate the overlap $|\langle \psi | \phi \rangle|^2$. For our discussion here, however, we require a deeper understanding of this test — what does it *really* do, i.e. which two-outcome projection does it implement given an arbitrary, possibly entangled input $|\psi\rangle_{AB}$?

Exercise 8.5. Suppose the circuit of Figure 8.1 receives entangled state $|\psi\rangle_{AB}$ on the registers entering the SWAP gate, and that measuring the first wire yields 0. What property might one expect the resulting post-selected state on system AB to have? (Hint: Could it be entangled? If yes, what would be the "entangled analog" of having "both halves" of $|\psi\rangle_{AB}$ being "equal"?)

In short, the two-outcome projective measurement the SWAP test simulates is a projection onto the *symmetric subspace* of AB. To define this, we first express the SWAP operator acting

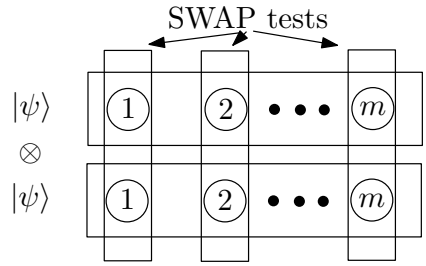


Figure 8.2: Given two copies of an m-partite state $|\psi\rangle$, the Product State test performs a SWAP test between the pair of ith systems from both copies, for all $i \in [m]$.

on $A \otimes B = \mathcal{B} \otimes \mathcal{B}$ as

$$\text{SWAP} = \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} |i\rangle\langle j|_A \otimes |j\rangle\langle i|_B.$$

The SWAP test now corresponds to projective measurement $\Pi = \{\Pi_{\text{sym}}, \Pi_{\text{asym}}\}$ (Π_{sym} for measurement result 0, Π_{asym} for 1), defined as

$$\Pi_{\text{sym}} := \frac{1}{2}(I + \text{SWAP})_{AB} \qquad \Pi_{\text{asym}} := \frac{1}{2}(I - \text{SWAP})_{AB}.$$

In words, Π_{sym} and Π_{asym} are the projectors onto the symmetric and anti-symmetric subspaces, respectively.

Exercise 8.6. Suppose we run the swap test on single qubit inputs $|0\rangle_A$ and $|1\rangle_B$. What is our post-measurement state upon measuring 0? How about upon measuring 1?

Exercise 8.7. Can you explicitly characterize the set of states onto which Π_{sym} projects? How about Π_{asym}? (Hint: What properties do 1-eigenvectors of each projector possess?)

Exercise 8.8. Show that the SWAP test indeed corresponds to projective measurement $\Pi = \{\Pi_{\text{sym}}, \Pi_{\text{asym}}\}$.

In sum, nothing prevents the SWAP test from being run on an entangled input $|\psi\rangle_{AB}$, it's just that a *tensor product* input $|\psi\rangle_A \otimes |\phi\rangle_B$ is symmetric under exchange of A and B if and only if $|\psi\rangle = |\phi\rangle$.

8.2.1 From SWAP test to the Product State test

With the SWAP test loaded back into RAM, the Product State test is now depicted in Figure 8.2. It proceeds as follows: Suppose an untrusted prover claims that an m-partite state $|\psi\rangle \in \mathcal{B}^{\otimes m}$ is in tensor product between all m copies of $\mathcal{B}$. In Figure 8.2, this means that in the top horizontal box of subsystems labelled 1 through m, the prover claims each subsystem $i \in [m]$ is in tensor product with all $j \neq i$. To test the prover's claim, the Product State test says one simply needs two *unentangled* copies of $|\psi\rangle$, i.e. the top horizontal box of Figure 8.2 is in tensor product with the bottom horizontal box. For all $i \in [m]$, the verifier runs the SWAP test between the ith subsystems of both copies of $|\psi\rangle$ (depicted by the vertical boxes in Figure 8.2), and accepts if and only if all SWAP tests output 0.

Exercise 8.9. Convince yourself that, formally, the Product State test performs two-outcome projective measurement $\Pi = \left\{\Pi_{\text{sym}}^{\otimes m}, I - \Pi_{\text{sym}}^{\otimes m}\right\}$, where the ith copy of Π_{sym} acts on the ith vertical box in Figure 8.2, where $i \in [m]$.

Exercise 8.10. Suppose $|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_m\rangle$ for all $|\phi_i\rangle \in \mathcal{B}$. Given two copies of $|\psi\rangle$ as input, with what probability does the Product State test accept?
Thus, an honest prover sending a tensor product state $|\psi\rangle$ is accepted with certainty. The question is: *What about soundness? With what probability can a cheating prover pass the test?*

Analyzing the Product State test. Intuitively, how well a state $|\psi\rangle$ passes the Product State should depend on "how entangled" it is across its m subsystems. In general, however, quantifying "how entangled" a multipartite system is a nightmare — in fact, already on 4-qubit systems, the number of distinct "types" (defined formally via SLOCC equivalence classes, which are beyond this lecture) of entanglement is infinite! Luckily, to analyze the Product State test, we need not distinguish all possible types of entanglement, but just quantify how close to *product* $|\psi\rangle$ is. The following geometric notion of "closeness to product" thus suffices:

$$\delta_\psi := \max_{\text{Product state } |\phi\rangle \in \mathcal{B}^{\otimes m}} |\langle\psi|\phi\rangle|^2$$

The soundness of the test now follows from the next theorem.

Theorem 8.11 (Soundness of Product State test)**.** *Given two copies of an m-partite state, $|\psi\rangle \in \mathcal{B}^{\otimes m}$, the Product State test passes with probability at most $\frac{1}{3}\delta_\psi^2 + \frac{2}{3}$.*

Proof. The proof[1] idea is simple — apply the Schmidt decomposition recursively with respect to the $\mathcal{B}$ systems in $|\psi\rangle$ recursively, each time using the fact that if the SWAP test succeeds on a copy of $\mathcal{B}$, then that copy of $\mathcal{B}$ is projected onto the symmetric subspace.

Formally, the proof proceeds via induction on m, where recall $|\psi\rangle \in \mathcal{B}^{\otimes m}$. A key observation is that □

8.2.2 Applications of the Product State test

8.3 Verifying NP with logarithmic-size unentangled proofs

8.4 Beyond QMA(2) — Quantum Polynomial Hierarchies

[1] This simplified proof is due to [Soleimanifar and Wright, 2022].

9 Quantum Interactive Proofs (QIP), semidefinite programs, and multiplicative weights

"There is one thing stronger than all the armies in the world, and that is an idea whose time has come."
— Victor Hugo

Introduction. In this course, we have considered the power of quantum verification systems in the presence of various types of communication: No communication (BQP), one-way classical communication (QCMA), and one-way quantum communication (QMA). A natural extension of these studies is to ask: *What happens when the communication is interactive, i.e. back and forth between the prover and verifier?* This is roughly the class QIP, and the high-level aim of this lecture is to study the result that QIP = PSPACE.

Along the way, we will encounter two elegant ideas which have found surprisingly fruitful uses in quantum information (as the opening quote of this section suggests, it is these ideas whose "time has come"). The first is the notion of *semidefinite programs (SDP)*, which are a fairly broad class of optimization problems typically solvable in polynomial-time, and which have become rather ubiquitous in quantum information. The second is the *multiplicative weights* (MW) method, which in some sense is the oldest trick in the book in terms of how humans handle unpredictable situations (at least according to the present instructor). Remarkably, both these tools come together to prove that polynomially many messages of interaction with a quantum verifier yields precisely PSPACE.

Organization. We begin in Section 9.1 with the multiplicative weights algorithm; while the context we present it in is unrelated to QIP, our discussion will nevertheless hopefully give you an intuition as to how the method arises. Section 9.2 then introduces quantum interactive proofs, semidefinite programs, and how the former can be phrased in terms of the latter. Section 9.3 finally gives the proof combining all the previous ingredients to show that QIP = PSPACE. We stress that this lecture is full of deep and fundamental ideas and tools; the MW method, SDPs, and interactive proofs alone can each fill many lectures many times over.

9.1 The Multiplicative Weights algorithm

In a nutshell, the Multiplicative Weights (MW) algorithm captures the age-old idea that *"if it worked once, it's likely to work again"*. Typically, the algorithm is introduced in the context of the stock market and stock advisors, but let us focus here on a more troublesome bunch: Professors. Suppose you are a first-year student entering a Computer Science program at your university, and at a loss as to which of many advanced courses to choose. Confused, you assemble an A-team of professors, labelled 1 through n, to guide you through this process. In your first semester, you ask: "Professors, professors, heed my call, which is the fairest course of them

all?" Each professor gives you an answer; but since you're new to the school, you have no idea whose advice to take. So in round 1, you pick some professor $i_1 \in [n]$ uniformly at random, and follow his/her advice. Once the semester is over, you reflect: *Was Professor i's advice good or bad?* If the advice was good, then once semester 2 starts and you repeat this process, you will naturally be more likely to follow Professor i's advice again. The question is: *How should you update your probability of following Professor i's advice at the end of each semester?* It turns out the right answer is "multiplicatively", and this is precisely where MW gets its name.

Formal setup and algorithm. Imagine we have a set E of n experts, and T rounds of some process for which we wish to take the experts' advice into account. In each round $t \in [T]$, we have a probability distribution p^t over E, such that we pick and follow (only) E_i's advice in round t with probability p_i^t. We imagine that the environment now assigns a "cost" to each expert i's choice in round t, denoted $-1 \leq c_i^t \leq 1$. Absolutely no assumptions are made on how these costs are set by the environment; they may even be set adversarially.

The MW algorithm is an *online* algorithm, meaning it tries to make the best choices at each step *without* access to future environmental data (i.e. costs c_i^t for future rounds t). Ideally, the goal would be to pick the "best expert" from the beginning, i.e. the one attaining minimum cost

$$\min_{i \in [n]} \sum_{t=1}^{T} c_i^t,$$

and follow this expert in each round. This is naturally impossible, but remarkably we can get "close" to this optimum, *even if the environment acts adversarially.* To formalize this, let $c^t \in [-1,1]^n$ and $p^t \in [0,1]^n$ denote the vectors encoding the cost and probability for all experts in a round t. (Thus, the odds of picking expert i in round t are $p_i^t \in [0,1]$, and the cost of doing so is $c_i^t \in [-1,1]$.) The *expected* cost of using distribution p^t in round t is hence

$$C_t := \sum_{i=1}^{n} p_i^t c_i^t = \left\langle c^t, p^t \right\rangle,$$

for $\langle a, b \rangle$ the inner product of a and b. The expected cost over *all* rounds is thus $C := \sum_{t=1}^{T} C_t = \sum_{t=1}^{T} \left\langle c^t, p^t \right\rangle$. Before stating the algorithm, let us state what it gives us.

Theorem 9.1. *Fix $0 < \epsilon \leq 1/2$. Then, for any expert E_i, after T rounds the MW algorithm obtains expected cost*

$$C \leq \sum_{t=1}^{T} c_i^t + \left[\epsilon \sum_{t=1}^{T} \left| c_i^t \right| + \frac{\ln n}{\epsilon} \right].$$

Remarks. We stress that the right hand side of the bound above holds for *any expert E_i*; thus, the MW algorithm is "close" to the best expert in terms of performance. The term in square brackets is the *additive error* term; it depends on the magnitude of the costs $\left| c_i^t \right|$ incurred by E_i.

The algorithm itself is now stated as Algorithm 1 (yes, it really is this simple).

Exercise 9.2. Why is there a division involved in Step 2(a) of Algorithm 1?

Proof of Theorem 9.1. As done in amortized analysis, it turns out to be useful to study a "potential function" as a "reference point": $\Phi^t := \sum_i w_i^t$. By giving upper and lower bounds on Φ, we will obtain the claim.

Algorithm 1 The Multiplicative Weights algorithm.

1. Fix $0 < \epsilon \leq 1/2$, and give each expert E_i "weight" $w_i^1 = 1$.
2. For $t = 1, \ldots, T$:
   a) Pick expert E_i with probability $w_i^t / (\sum_i w_i^t) = w_i^t / \Phi^t$.
   b) Obtain costs c^t for round t from the environment.
   c) Update weight of expert E_i as $w_i^{t+1} := \begin{cases} w_i^t (1-\epsilon)^{c_i^t} & \text{if } c_i^t \geq 0 \\ w_i^t (1+\epsilon)^{-c_i^t} & \text{if } c_i^t < 0 \end{cases}$.

Upper bound. By definition,

$$\Phi^{t+1} = \sum_i w_i^{t+1} \leq \sum_i w_i^t (1 - \epsilon c_i^t) = \Phi^t (1 - \epsilon \left\langle c^t, p^t \right\rangle) \leq \Phi^t e^{-\epsilon \left\langle c^t, p^t \right\rangle},$$

where the first inequality follows from the next exercise.

Exercise 9.3. Prove that $(1 \mp \epsilon)^{\pm x} \leq (1 - \epsilon x)$ when $x \in [0, 1]$ and $x \in [-1, 0]$, respectively. Use these facts to obtain the first inequality above.

Exercise 9.4. Use the fact that $p_i^t = w_i^t / \Phi^t$ to show the second equality above.

Exercise 9.5. Prove that $e^{-x} \geq 1 - x$ for $x \in [-1, 1]$. Use this to prove that last inequality above. (Hint: You need to use an assumption we made on the costs c_i^t.)

Since by definition $\Phi^1 := n$, it follows that after T steps, $\Phi^{T+1} \leq n e^{-\epsilon \sum_{t=1}^T \left\langle c^t, p^t \right\rangle}$, our desired upper bound.

Lower bound. Since the weights $w_i^t \geq 0$, we have that for any E_i, $\Phi^{t+1} \geq w_i^{t+1}$. Combining the upper bound above with the following exercise now yields the claim.

Exercise 9.6. Prove that for $0 \leq \epsilon \leq 1/2$, $\ln(1/(1-\epsilon)) \leq \epsilon + \epsilon^2$ and $\ln(1+\epsilon) \geq \epsilon - \epsilon^2$.

Exercise 9.7. Use the exercise above to prove that $\ln(w_i^{t+1}) \geq -\epsilon \sum_i c_i^t - \epsilon^2 \sum_i \left| c_i^t \right|$. Combine this with our upper bound to obtain the claim. □

This completes our discussion of the basic MW method. For a gentle introduction to its extension to the matrix setting, the reader is referred to the thesis of Kale, available at `http://www.satyenkale.com/papers/thesis.pdf`. In the interest of time, we shall instead jump right into QIP and semidefinite programs.

9.2 QIP and semidefinite programs

Ultimately, our goal is to use a matrix variant of the MW algorithm to show that QIP = PSPACE. To do so, we first require a definition of *quantum interactive proofs*, and subsequently an approach for embedding their maximum acceptance probability into a *semidefinite program*. These are given in Sections 9.2.1 and 9.2.2, respectively.

9.2.1 Quantum interactive proofs

Roughly, classical interactive proofs are the natural extension of NP to the setting in which the prover (which remains computationally unbounded) can now exchange polynomially many rounds of communication with the verifier (which remains polynomially bounded). Quantumly, the idea is analogous: The quantum prover (which remains computationally unbounded, with the exception of obeying the laws of quantum mechanics) now exchanges polynomially many rounds of quantum communiction with the verifier (which is still computationally bounded). Slightly more formally, we imagine the prover and verifier send a common quantum "message register" M back and forth, and each take turns applying a local unitary circuit on M and their private workspace.

Formal definition. To make this intuition formal, we must generalize our definition of a QMA verifier to an m-round quantum verifier (and analogously, an m-round quantum prover).

Definition 9.8. *(m-round quantum verifier) An m-round quantum verifier is a P-uniform circuit family $Q = \{Q_{n,1}, \dots Q_{n,m}\}$, acting on three registers: An input register A containing $x \in \{0,1\}^n$, a message register M consisting of $p(n)$ qubits, and an ancilla or "private" register V consisting of $q(n)$ qubits, for some polynomials $p, q : \mathbb{N} \to \mathbb{N}$. We imagine the verifier acts in "rounds", applying circuit $Q_{n,i}$ in round $i \in [m]$. Before round 1, the message and private registers are initialized to all zeroes.*

Definition 9.9. *(m-round quantum prover) An m-round quantum prover is defined identically to an m-round verifier, except the circuit family $Q = \{Q_{n,1}, \dots Q_{n,m}\}$ need not be P-uniform (indeed, the circuits can in be superpolynomial in size). To help distinguish between verifier and prover, for the latter we label the private register as P for "prover" (versus V for "verifier").*

Exercise 9.10. Why does dropping the P-uniformity condition in Definition 9.9 capture the notion of a prover which is limited only by the laws of quantum mechanics?

For brevity, we henceforth drop the input size n when referring to circuits $Q_{n,i}$. We can now model an interactive protocol between an m-round verifier and m-round prover in the natural way: Letting $\{Q_i\}$ and $\{R_i\}$ denote their respective circuits, the interaction is given by (for simplicity, we omit the A register, as we may assume without loss of generality that its contents remain fixed to the input x for both parties):

$$(Q_m)_{V,M}(R_m)_{M,P} \cdots (Q_1)_{V,M}(R_1)_{M,P}|0\cdots 0\rangle_V|0\cdots 0\rangle_M|0\cdots 0\rangle_P.$$

Just as in QMA, the verifier now measures a designated output qubit of register V, say V_1, and accepts if and only if the output is 1.

Exercise 9.11. Suppose we wish the interactive protocol to instead begin with a message from the verifier to the prover; how can we model that in the setup above?

Exercise 9.12. Why can we assume without loss of generality that the verifier acts last?

We may now formally define the class QIP.

Definition 9.13. *(Quantum Interactive Proof Systems (QIP)) A promise problem $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ is in QIP if there exists a polynomial $m : \mathbb{N} \to \mathbb{N}$ and m-round quantum verifier satisfying the following for any input $x \in \{0,1\}^n$:*

- *(Completeness/YES case) If $x \in A_{\text{yes}}$, there exists an m-round quantum prover causing the verifier to accept with probability at least $2/3$.*
- *(Soundness/NO case) If $x \in A_{\text{no}}$, then for all m-round quantum provers, the verifier accepts with probability at most $1/3$.*
- *(Invalid case) If $x \in A_{\text{inv}}$, the verifier may accept or reject arbitrarily.*

Magic: Only a constant number of rounds are needed. Remarkably, and in strong contrast to classical interactive proofs, it turns out quantumly that the use of polynomially many rounds of communication in QIP is overkill — it suffices to use just 2 rounds:

$$(Q_2)_{V,M}(R_2)_{M,P}(Q_1)_{V,M}(R_1)_{M,P}|0\cdots 0\rangle_V|0\cdots 0\rangle_M|0\cdots 0\rangle_P.$$

In fact, we may even assume the only message from the verifier to the prover (modelled by Q_1) is an unbiased coin flip (i.e. not conditioned on the action of R_1). This is known as a "Quantum Arthur-Merlin" game, and in the context of QIP, we may assume the accompanying completeness and soundness parameters are 1 and $1/2 + \epsilon$ for any fixed $\epsilon > 0$, respectively. For the remainder of this lecture, we shall henceforth use this Quantum Arthur-Merlin characterization of QIP.

Exercise 9.14. Since Q_1 above is just an unbiased coin flip independent of R_1, at first glance it may seem one can simply remove R_1 altogether. Why might this be a bad idea?

9.2.2 Semidefinite programming

Linear programming and its generalization, semidefinite programming, are both misnomers: Neither of them has anything to do "programming" in the usual computer science sense. Both actually refer to a fairly broad class of optimization problems with a wide variety of applications. Generally, a linear program (LP) attempts to maximize a given linear function $f : \mathbb{R}^n \to \mathbb{R}$, subject to a set of given linear inequality constraints. The main syntactic difference in generalizing from an LP to a semidefinite program (SDP) is that now the variables are not vectors in $\mathbb{R}^n$, but Hermitian *matrices* in $\mathcal{L}(\mathbb{C}^n)$, and hence the correct notion of inequality is the generalized inequality $\succeq$ denoting the positive semidefinite ordering (i.e. $A \succeq B$ if and only if $A - B$ is positive semidefinite).

Exercise 9.15. Is $I \succeq 2I$? How about $I \succeq X$ for Pauli X? $I \succeq 2X$? If $A \succeq B$ and $C \succeq D$, is $A + C \succeq B + D$?

Exercise 9.16. If $A \succeq 0$ and $B \succeq 0$, is it always true that $AB \succeq 0$? (Hint: A necessary condition for AB to be positive semidefinite is for it to be Hermitian.)

Standard form for SDPs. We begin by stating the standard form for SDPs we shall use. To do so, we need three things: (1) A *cost matrix* $C \in \text{Herm}(\mathcal{X})$, (2) a *constraint matrix* $D \in \text{Herm}(\mathcal{Y})$, and (3) a linear *constraint map* $\Psi : \text{Herm}(\mathcal{X}) \to \text{Herm}(\mathcal{Y})$. Here, $\mathcal{X}$ and $\mathcal{Y}$ are fixed complex vector spaces. The (primal) semidefinite program is then given by:

Primal problem (P)		Dual problem (D)	
supremum:	$\mathrm{Tr}(CX)$	infimum:	$\mathrm{Tr}(DY)$
subject to:	$\Psi(X) \preceq D,$	subject to:	$\Psi^*(Y) \succeq C,$
	$X \succeq 0,$		$Y \succeq 0.$

This may look cryptic at first sight, so let us first break down some facts about the primal problem, P:

- The *variable* being optimized over is $X \in \mathrm{Herm}(\mathcal{X})$. The *feasible region* is the set of all "valid assignments", i.e. all X satisfying the given constraints, $\Psi(X) \preceq D$ and $X \succeq 0$.
- The *objective function* being maximized, $\mathrm{Tr}(CX)$, is linear in X. In analogy with Section 9.1, we may use inner product notation $\langle C, X\rangle := \mathrm{Tr}(C^\dagger X)$ (recall C is Hermitian in our setting).

  Exercise 9.17. Prove that the objective function is indeed linear in X.

  Exercise 9.18. The function $\langle A, B\rangle := \mathrm{Tr}(A^\dagger B)$ is sometimes called the Hilbert-Schmidt inner product for matrices. Why does this name make sense? (Hint: How is this function really just an example of the usual vector inner product?)

- The map Ψ must be *Hermiticity preserving*, i.e. map Hermitian operators to Hermitian operators.

  Exercise 9.19. Why is the requirement that Ψ be Hermiticity preserving necessary?

- Once an SDP is in the standard form P above, one can formulate the corresponding *dual* problem, D, which is also an SDP. We shall say more about duality shortly, but for now let us define the *adjoint* map $\Psi^* : \mathrm{Herm}(\mathcal{Y}) \to \mathrm{Herm}(\mathcal{X})$; it is the unique linear map satisfying $\langle A, \Psi(B)\rangle = \langle \Psi^*(A), B\rangle$ for all $A \in \mathcal{L}(\mathcal{Y}), B \in \mathcal{L}(\mathcal{X})$.

  Exercise 9.20. Let $\Psi : \mathrm{L}(\mathbb{C}^n) \to \mathbb{C}$ be the trace function. What is Ψ^*?

The topic of semidefinite programming, and its generalization to convex optimization, fills up entire textbooks. Here, we shall aim to convey the basic intuition and facts needed for studying QIP via some examples and key statements.

Examples and optimal solutions. As with most things in life, you have actually been using SDPs without knowing it. The simplest example of an SDP which you know is the computation of the largest eigenvalue of a Hermitian matrix $C \in \mathrm{Herm}(\mathbb{C}^n)$, which can be written:

Primal problem (P)		Dual problem (D)	
supremum:	$\mathrm{Tr}(CX)$	infimum:	y
subject to:	$\mathrm{Tr}(X) \leq 1$	subject to:	$y \cdot I \succeq C,$
	$X \succeq 0,$		

Exercise 9.21. What space does dual variable y live in?

Exercise 9.22. Technically, our standard definition of the dual problem required $y \geq 0$. How can we rewrite D above to be in standard form? (Hint: Any $a \in \mathbb{R}$ can be written $a = b - c$ for some $b, c \geq 0$.)

Two comments: (1) It is not necessary to always put an SDP into standard form; this is more for convenience, as it makes computation of the dual SDP easier. (Certain numerical SDP solvers may also request it.) Generally speaking, an SDP is any optimization (min or max) of a linear function, subject to linear inequality constraints and non-negativity constraints (with respect to $\succeq$). (2) The use of *supremum* in P above is unnecessary; in this case, the optimal value is attained and is precisely $\lambda_{\max}(C)$. More generally, however, this is not true, as the following example demonstrates.

$$\begin{aligned} \text{infimum:} \quad & x \\ \text{subject to:} \quad & \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \succeq 0 \\ & x, y \in \mathbb{R} \end{aligned}$$

Exercise 9.23. Prove that $x = 0$ is not in the feasible region. Conclude that 0 as an objective function value is not attainable. (Hint: Use the Determinant Test, which states for a 2×2 matrix that $A \succeq 0$ if and only if $A_{11}, A_{22} \geq 0$ and $\det(A) \geq 0$.)

Exercise 9.24. Prove that for *any* $x > 0$, there exists a choice of y so that (x, y) is a feasible solution to the dual problem. Conclude that the use of infimum is necessary in this example.

Duality theory. You may be wondering why we've been dragging around the dual problem for each primal problem we've stated. Let p and d denote the optimal values for a primal and corresponding dual SDP, respectively. These values satisfy an amazing property known as *weak duality*:

$$p \leq d.$$

This immediately gives a powerful use for SDPs. Suppose you have some maximization problem Π which is difficult to solve analytically (for example, this might encode the optimal cheating probability for a cryptographic protocol). If you can formulate Π (or a relaxation of it) as an SDP, then it is "easy" to give an upper bound on Π — any feasible solution to the dual SDP will, by weak duality, yield *some* upper bound on Π. Note, crucially, that this does *not* require solving an SDP; one can often make a clever guess as to what a good dual solution should be.

Of course, the natural question is whether this upper bound can be made tight, i.e. is it true that $p = d$? This is called *strong duality*. In general, strong duality unfortunately does not hold. However, a simple sufficient condition for strong duality is Slater's constraint qualification, which states that if (say for the primal) there is a *strictly feasible* solution X (i.e. $\Phi(X) \prec D$ and $X \succ 0$), then strong duality holds.

Exercise 9.25. Consider the non-standard dual program below. Show that it does not satisfy strong duality.

$$\begin{aligned} \text{infimum:} \quad & x \\ \text{subject to:} \quad & \begin{pmatrix} 0 & x & 0 \\ x & y & 0 \\ 0 & 0 & x+1 \end{pmatrix} \succeq 0 \\ & x, y \in \mathbb{R} \end{aligned}$$

Runtime. It is a common fallacy that "SDPs can be solved in polynomial time". While the spirit of the statement is true, in order to actually attain a poly-time solution, two constraints must be met: The feasible region must be contained in a ball of radius R, and must contain a ball of radius r. The runtime of the Ellipsoid Algorithm is then polynomial in the input size (i.e. encodings of C, Ψ, D; we assume this encoding size scales at least as the dimension of the space C acts on), $\log R$, $\log(1/r)$, and $\log(1/\epsilon)$, where ϵ is the additive error in the optimal objective function value we are willing to tolerate. In practice, these conditions are typically met. Note also that while the runtime of the Ellipsoid Algorithm is often cited in theoretical algorithmic results relying on SDPs, in practice more stable and modern methods such as Interior Point Methods are deployed. In this lecture, we will see an alternative method for solving certain SDPs; the matrix multiplicative weights method.

9.2.3 Quantum interactive proofs as SDPs

Having introduced quantum interactive proofs and SDPs, we can now formulate the former as an example of the latter. As stated in Section 9.2.1, we shall assume a 3-message Quantum Arthur-Merlin protocol, which suffices to capture QIP. The setup is as follows:

1. The prover (Merlin) sends the verifier (Arthur) a density operator σ in register M_1 (for "message 1").

2. Arthur has a pair of measurements $\{P_0, I - P_0\}$ and $\{P_1, I - P_1\}$ in mind, for $0 \preceq P_0, P_1 \preceq I$ and acting on joint space $M_1 \otimes M_2$. Arthur chooses a uniformly random bit $b \in \{0, 1\}$, and sends it to Merlin.

3. Merlin sends quantum register M_2 (for "message 2") to Arthur.

4. Arthur performs measurement $\{P_b, I - P_b\}$ on the message registers $M_1 \otimes M_2$, and accepts if and only if the outcome is P_b.

Formally, one can model the acceptance POVM for Arthur's random measurement via operator

$$Q := |0\rangle\langle 0|_C \otimes (P_0)_{M_1,M_2} + |1\rangle\langle 1|_C \otimes (P_1)_{M_1,M_2}.$$

To see why, note that conditioned on bit flip b, Merlin prepares joint state $\rho_b \in \mathcal{L}(M_1 \otimes M_2)$. In other words, the density operator prepared by Merlin is

$$X := \frac{1}{2}|0\rangle\langle 0|_C \otimes (\rho_0)_{M_1,M_2} + \frac{1}{2}|1\rangle\langle 1|_C \otimes (\rho_1)_{M_1,M_2}. \tag{9.1}$$

Exercise 9.26. Show that the probability of Arthur accepting is $\mathrm{Tr}(QX) = \frac{1}{2}(\mathrm{Tr}(P_0\rho_0) + \mathrm{Tr}(P_1\rho_1))$.

Of course, ρ_0 and ρ_1 cannot be *arbitrary* — in step 1 of the protocol, Merlin committed some state σ on space M_1, which remains untouched throughout the remainder of the protocol. Thus, it must be that both possible end states ρ_0 and ρ_1 agrees on this "commitment space" M_1, i.e.

$$\mathrm{Tr}_{M_2}(\rho_0) = \mathrm{Tr}_{M_2}(\rho_1) = \sigma.$$

Exercise 9.27. Consider any pair of purifications $|\psi_0\rangle_{AB}, |\psi_1\rangle_{AB}$ of some density operator σ_B. Prove that there exists a unitary U_A such that $(U_A \otimes I_B)|\psi_0\rangle\langle\psi_0|_{AB}(U_A^\dagger \otimes I_B) = |\psi_1\rangle\langle\psi_1|_{AB}$. Conclude that the restriction to fixing σ on the commitment space M_1 above is without loss of generality (i.e. for any pair of pure states $|\psi_0\rangle$ and $|\psi_1\rangle$ Merlin wants to prepare on $M_1 \otimes M_2$, he may do so (possibly inefficiently), as long as $|\psi_0\rangle$ and $|\psi_1\rangle$ agree on their reduced state σ on M_1).

With these observations in hand, we can state the primal and dual SDPs capturing our interactive protocol, where for convenience we have moved the factor of $1/2$ from X to Q:

Primal problem (P)

maximize:	$\mathrm{Tr}(QX)$
subject to:	$\mathrm{Tr}_{M_2}(X) \preceq I_C \otimes \sigma_{M_1}$
	$\mathrm{Tr}(\sigma_{M_1}) = 1$
	$\sigma_{M_1} \succeq 0$
	$X_{C,M_1,M_2} \succeq 0$

Dual problem (D)

minimum:	$\| \mathrm{Tr}_C(Y_{C,M_1}) \|_\infty$
subject to:	$Y_{C,M_1} \otimes I_{M_2} \succeq Q,$
	$Y_{C,M_1} \succeq 0.$

Again, let us break down the primal SDP:

- The constraints $\mathrm{Tr}(\sigma) = 1$ and $\sigma \succeq 0$ ensure σ is a density operator.
- Ideally, we wish to force X to be of the form in Equation (9.1) (but without the $1/2$), which is block diagonal with respect to C:

  $$X = \begin{pmatrix} \rho_0 & 0 \\ 0 & \rho_1 \end{pmatrix}.$$

  In principle, we could explicitly enforce this by having separate variables for ρ_0 and ρ_1; however, since Q is also block diagonal with respect to C, the off-diagonal blocks of X do not alter the value of the objective function. Thus, without loss of generality the optimal X sets the off-diagonal blocks to 0 (this technically requires the following exercise).

  Exercise 9.28. Prove that if $\begin{pmatrix} A & B \\ B^\dagger & C \end{pmatrix} \succeq 0$, then $\begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix} \succeq 0$. Why is this exercise needed in assuming the off-diagonal blocks of X are 0?

  Exercise 9.29. Write Q in block form with respect to register C, and confirm it is block-diagonal.

- We must enforce that ρ_0 and ρ_1 have reduced state σ on M_1. To see why the SDP captures this, we require the following exercise.

Exercise 9.30. Prove in Equation (9.1) that tracing out M_2 yields $I_C \otimes \sigma_{M_1}$, assuming ρ_0 and ρ_1 have reduced state σ on M_1 (and omitting the 1/2 factor).

This almost explains the last primal SDP constraint, $\mathrm{Tr}_{M_2}(X) \preceq I_C \otimes \sigma_{M_1}$ — here we have an inequality without loss of generality (as opposed to an equality), since any feasible solution to the inequality can be "boosted" to make the inequality tight, while only increasing the objective function value.

Exercise 9.31. Prove the claim above: Suppose $\mathrm{Tr}_{M_2}(X) \preceq I_C \otimes \sigma_{M_1}$ but $\mathrm{Tr}_{M_2}(X) \neq I_C \otimes \sigma_{M_1}$. Give a new operator X' such that $\mathrm{Tr}_{M_2}(X) = I_C \otimes \sigma_{M_1}$ and $\mathrm{Tr}(QX') \geq \mathrm{Tr}(QX)$. (Hint: By assumption, $I_C \otimes \sigma_{M_1} - \mathrm{Tr}_{M_2}(X) \succeq 0$.)

- Finally, we have quietly replaced our use of supremum with maximum.

Exercise 9.32. Prove that Slater's constraint qualification holds, implying strong duality holds for P.

Some final massaging. Just as applying a unitary change of basis in our analysis of the Quantum Cook-Levin theorem helped with its analysis, here it turns out to be useful to also perform an appropriate change of variables. The final SDP we obtain is

Primal problem (P)

$$\begin{aligned} \text{maximize:} \quad & \mathrm{Tr}(X) \\ \text{subject to:} \quad & \Phi(X) \preceq I_C \otimes \sigma_{M_1} \\ & \mathrm{Tr}(\sigma) = 1 \\ & \sigma_{M_1} \succeq 0 \\ & X_{C,M_1,M_2} \succeq 0 \end{aligned}$$

Dual problem (D)

$$\begin{aligned} \text{minimum:} \quad & \| \mathrm{Tr}_C(Y) \|_\infty \\ \text{subject to:} \quad & \Phi^*(Y_{C,M_1}) \succeq I_{C,M_1,M_2} \\ & Y_{C,M_1} \succeq 0, \end{aligned}$$

where we define $\Phi : \mathcal{L}(C \otimes M_1 \otimes M_2) \to \mathcal{L}(C \otimes M_1)$ as

$$\Phi(X) := \mathrm{Tr}_{M_2}(Q^{-1/2} X Q^{-1/2}), \tag{9.2}$$

with adjoint map $\Phi^* : \mathcal{L}(C \otimes M_1) \to \mathcal{L}(C \otimes M_1 \otimes M_2)$ as $\Phi^*(Y) = Q^{-1/2}(Y \otimes I_{M_2})Q^{-1/2}$. Henceforth, P and D will always refer to this primal and dual problem, respectively.

9.3 QIP = PSPACE

We are now in a position to show that QIP = PSPACE. One direction of this equality is "trivial", in that QIP $\supseteq$ IP, for IP the classical analogue of QIP, which is known to equal PSPACE. Thus, the non-trivial direction is the containment QIP $\subseteq$ PSPACE.

A bit of interpretation and context. On the one hand, the statement QIP = PSPACE = IP is somewhat disappointing, in that as far as (single-prover) interactive proofs are concerned, quantum resources add no power. On the other hand, it may be interpreted as saying that interactive proofs are themselves so powerful that additional resources such as quantum computation add

nothing new to the picture. It is worth noting, however, that remarkably, this state of affairs is only the case for *single-prover* interactive proofs. If we move to multiple prover interactive proofs (i.e. MIP, with multiple provers who may not communicate with each other once the protocol starts), it is known that classically MIP = NEXP, whereas quantumly $\text{MIP}^* = \text{RE}$ (you read that right; MIP^* can verify even the Halting problem, which is undecidable)! Here, MIP^* is MIP but where the provers are allowed to share entanglement before the protocol starts.

We begin by stating the algorithm which allows us to put QIP in PSPACE in Section 9.3.1. Correctness is shown in Section 9.3.2.

9.3.1 The algorithm

Before stating the algorithm, we sketch why it will imply containment of QIP in PSPACE.

Connection to PSPACE. In Section 9.2.3, we showed how to exactly capture the acceptance probability of a 3-message Quantum Arthur-Merlin game (and hence QIP) via an SDP P. In principle, one can then try to apply the Ellipsoid method to solve P, which would require time polynomial in the dimension of the matrices involved, such as Q. Unfortunately, Q has dimension *exponential* in the number of qubits, n, and so the best the Ellipsoid method could give us is containment in EXP.

Exercise 9.33. Why is the dimension of Q exponential in n?

We hence need an alternate approach for solving P. To begin, what we certainly *can* do in PSPACE is produce explicit descriptions of the matrix Q.

Exercise 9.34. Convince yourself that there exists a (uniformly generated) circuit of size exponential in n and depth polynomial in n which outputs the full classical description of matrix Q (ignoring issues of precision in representing individual entries of Q here). The class of decision problems decidable by such circuits is called NC(poly); crucially for us, NC(poly) = PSPACE.

Next, given a matrix M of dimension $d \times d$, it turns out one can also compute common matrix operations on M using circuits of size $\text{poly}(d)$ and depth $\text{polylog}(d)$; this includes, for example, matrix powers (M^k), matrix exponentials (e^M), and spectral decompositions. (When d is polynomial in the input size, the corresponding complexity class is called NC, and formally requires all circuits to be generated by a log-space TM. Intuitively, NC is the log-depth analogue of P.) This means that after computing the explicit matrix representation for Q in PSPACE above, we can then do things like take the spectral decomposition of Q in PSPACE as well. In other words, if we could just come up with an algorithm for solving SDP P which relies solely on the application of common matrix operations to Q, then we could put the entire thing into PSPACE. This is precisely what the matrix analogue of the multiplicative weights method allows us to do.

Statement of the algorithm. The idea for solving SDP P via the MW method is analogous to Section 9.1. Let us restate the SDP and its dual for convenience first.

Primal problem (P)

$$\begin{aligned} \text{maximize:} \quad & \mathrm{Tr}(X) \\ \text{subject to:} \quad & \Phi(X) \preceq I_C \otimes \sigma_{M_1} \\ & \mathrm{Tr}(\sigma) = 1 \\ & \sigma_{M_1} \succeq 0 \\ & X_{C,M_1,M_2} \succeq 0 \end{aligned}$$

Dual problem (D)

$$\begin{aligned} \text{minimum:} \quad & \| \mathrm{Tr}_C(Y) \|_\infty \\ \text{subject to:} \quad & \Phi^*(Y_{C,M_1}) \succeq I_{C,M_1,M_2} \\ & Y_{C,M_1} \succeq 0, \end{aligned}$$

for $\Phi(X) := \mathrm{Tr}_{M_2}(Q^{-1/2} X Q^{-1/2})$, $\Phi^*(Y) = Q^{-1/2}(Y \otimes I_{M_2})Q^{-1/2}$, and

$$Q := \frac{1}{2}\left(|0\rangle\langle 0|_C \otimes (P_0)_{M_1,M_2} + |1\rangle\langle 1|_C \otimes (P_1)_{M_1,M_2}\right).$$

There are two primal variables in play: X and σ. We have no idea what they should be set to, so just as in Section 9.1, we start with a random guess by setting both to the maximally mixed state. In each iteration, we check "how badly" the constraint $I_C \otimes \sigma_{M_1} - \Phi(X) \succeq 0$ is violated, and update our guesses for X and σ accordingly. For clarity, the actual implementation, given below, differs somewhat from this, but this is the basic spirit. The variables ρ and ζ roughly play the roles of X and σ, respectively (the actual choices of X and σ for P will be slight modifications of the final values of ρ and ζ).

Algorithm 2 Multiplicative Weights for QIP

1. For brevity, define $N = \dim(C \otimes M_1 \otimes M_2)$ and $M = \dim(M_1)$.
2. Set parameters $\gamma = 4/3$, $\epsilon = 1/64$, $\delta = \epsilon/(2 \left\| Q^{-1} \right\|_\infty)$, $T = \lceil 4 \log N/(\epsilon^3 \delta) \rceil$.
3. Set initial states $\rho_0 = W_0/N$ for $W_0 = I_{C \otimes M_1 \otimes M_2}$ and $\zeta_0 = Z_0/M$ for $Z_0 = I_{M_1}$.
4. For $t = 0, \ldots, T$:
   a) (Check constraint violation) Let Π_t project onto the space spanned by the eigenvectors of $\Phi(\rho_t) - \gamma I_C \otimes \zeta_t$ with non-negative eigenvalues. Set $\beta_t = \mathrm{Tr}(\Phi(\rho_t)\Pi_t)$.
   b) If $\beta_t \leq \epsilon$, accept.
   c) (Update current solution) Set

$$\begin{aligned} \rho_{t+1} &= W_{t+1}/\mathrm{Tr}(W_{t+1}) \qquad \text{for} \qquad W_{t+1} = e^{-\epsilon\delta \sum_{j=0}^{t} \Phi^*\left(\frac{1}{\beta_j}\Pi_j\right)} \\ \zeta_{t+1} &= Z_{t+1}/\mathrm{Tr}(Z_{t+1}) \qquad \text{for} \qquad Z_{t+1} = e^{\epsilon\delta \sum_{j=0}^{t} \mathrm{Tr}_C\left(\frac{1}{\beta_j}\Pi_j\right)}. \end{aligned}$$

5. Reject.

It is worth stressing that Algorithm 2 does *not* directly output a solution (X, σ) to SDP P. Instead, given its output (ρ, ζ), what we can do is the following: If Algorithm 2 accepts, then we can extract a "good" feasible solution (X, σ) for P from (ρ, ζ) which convinces us we are in a YES case. Conversely, if Algorithm 2 rejects, we can construct a "good" feasible *dual* solution Y for D which convinces us we are in a NO case (recall that any dual feasible solution upper bounds the value of the primal SDP from Section 9.2.2).

9.3.2 Correctness

We now show correctness of Algorithm 2. Let $\mathcal{A} = (A_{\text{yes}}, A_{\text{no}}, A_{\text{inv}})$ denote a QIP promise problem, which recall has a 3-message Quantum Arthur-Merlin game with verifier V. For any input $x \in \{0,1\}^*$, we may assume that if $x \in A_{\text{yes}}$, V accepts with certainty, and if $x \in A_{\text{no}}$, V accepts with probability at most $1/2 + \epsilon$ for (say) $\epsilon = 1/64$. Thus, if $x \in A_{\text{yes}}$, the primal SDP P and (by strong duality) dual SDP D in Section 9.2.3 achieve value 1, and if $x \in A_{\text{no}}$, they achieve at most $1/2 + 1/64$.

Theorem 9.35. *If Algorithm 2 accepts, then P has optimal value strictly larger than $5/8$. If Algorithm 2 rejects, then P has optimal value strictly smaller than $7/8$.*

Exercise 9.36. Why does Theorem 9.35 show that Algorithm 2 correctly decides our QIP instance x?

We break the proof down into lemmas for the YES and NO cases.

Lemma 9.37. *(YES case) If Algorithm 2 accepts, then P has optimal value strictly larger than $5/8$.*

Proof. Let ρ, ζ, Π, and β denote the final values to $\rho_t, \zeta_t, \Pi_t, \beta_t$ set by Algorithm 2. We claim that

$$X = \frac{1}{\gamma + 2\beta}\rho \qquad\qquad \sigma = \frac{1}{\gamma + 2\beta}\left[\gamma\zeta + 2\text{Tr}_C(\Pi\Phi(\rho)\Pi)\right]$$

is a feasible solution to P with value strictly larger than $5/8$.

Exercise 9.38. Prove that the objective function value, $\text{Tr}(X)$, is indeed strictly larger than $5/8$.

Exercise 9.39. Show that σ is a density operator.

It thus remains to show that the constraint $\Phi(X) \preceq I_C \otimes \sigma_{M_1}$ is satisfied. For this, rearrange

$$\Phi(X) - \gamma I_C \otimes \zeta \preceq \Pi(\Phi(X) - \gamma I_C \otimes \zeta)\Pi \preceq \Pi\Phi(X)\Pi \preceq 2I_C \otimes \text{Tr}_C(\Pi\Phi(X)\Pi),$$

where the first inequality holds by the definition of Π, the second since $\Pi(I_C \otimes \zeta)\Pi \succeq 0$, and the third by the fact that $M_{AB} \preceq 2I_A \otimes \text{Tr}_A(M)$ for any $M_{AB} \succeq 0$ and $A = \mathbb{C}^2$. □

Lemma 9.40. *(NO case) If Algorithm 2 rejects, then P has optimal value strictly smaller than $7/8$.*

Proof. We claim that a dual feasible solution to D is

$$Y = \frac{1 + 2\epsilon}{T}\sum_{t=0}^{T-1}\frac{1}{\beta_t}\Pi_t,$$

and that Y attains dual objective function value $\| \text{Tr}_C(Y) \|_\infty < 7/8$; the lemma then holds by weak duality. As the proofs of both claims use a similar approach, we show only feasibility here.

Y **is dual feasible.** The proof approach is reminiscent of that of Theorem 9.1 (the classical MW algorithm); it turns out the correct "potential function" to use now, however, is $\mathrm{Tr}(W_T)$. We show upper and lower bounds on $\mathrm{Tr}(W_T)$ to establish that $\Phi^*(Y_{C,M_1}) \succeq I_{C,M_1,M_2}$. Equivalently, we show $\lambda_{\min}(\Phi^*(Y_{C,M_1})) \geq 1$.

Exercise 9.41. Show that $Y \succeq 0$.

Lower bound on $\mathrm{Tr}(W_T)$*.* We begin with the lower bound, which is simpler, and follows from the same principle as in the classical lower bound of Theorem 9.1 — the sum of non-negative numbers $\sum_i w_i$ is at least as large as any one number w_i.

Exercise 9.42. Prove that

$$\mathrm{Tr}(W_T) = \mathrm{Tr}\left(e^{-\epsilon\delta\sum_{j=0}^{T-1}\Phi^*\left(\frac{1}{\beta_j}\Pi_j\right)}\right) \geq e^{-\epsilon\delta\lambda_{\min}\left(\Phi^*\left(\sum_{j=0}^{T-1}\frac{1}{\beta_j}\Pi_j\right)\right)}. \tag{9.3}$$

Upper bound on $\mathrm{Tr}(W_T)$*.* We would like to upper bound

$$\mathrm{Tr}(W_T) = \mathrm{Tr}\left(e^{-\epsilon\delta\sum_{j=0}^{T-1}\Phi^*\left(\frac{1}{\beta_j}\Pi_j\right)}\right)$$

using an inductive approach similar to the upper bound in the proof of Theorem 9.1. The problem is that the exponents $\Phi^*(\Pi_j/\beta_j)$ do not pairwise commute; thus we cannot simply factor out the last term $\exp(-\epsilon\delta\Phi^*(\frac{1}{\beta_{T-1}}\Pi_{T-1}))$ and apply the induction hypothesis. Luckily, since we are interested in the *trace* of W_T, we are saved by the Golden-Thompson inequality, which states that

$$\mathrm{Tr}\left(e^{A+B}\right) \leq \mathrm{Tr}\left(e^A e^B\right) \text{ for Hermitian } A, B.$$

Exercise 9.43. For any $t \in \{1, \ldots, T\}$, use the Golden-Thompson inequality to show that

$$\mathrm{Tr}(W_t) \leq \mathrm{Tr}(W_{t-1}e^{-\epsilon\delta\Phi^*(\Pi_{t-1}/\beta_{t-1})}). \tag{9.4}$$

Now that we've isolated the last term in the exponential, we wish to bring its argument $\delta\Phi^*(\Pi_{t-1}/\beta_{t-1})$ down out of the exponent. For this, we use the fact that for any Hermitian M satisfying $0 \preceq M \preceq I$, and every real $r > 0$,

$$e^{rM} \preceq I + re^r M \quad \text{and} \quad e^{-rM} \preceq I - re^{-r}M.$$

From this, we immediately have that

$$e^{-\epsilon[\delta\Phi^*(\Pi_{t-1}/\beta_{t-1})]} \preceq I - \epsilon\delta e^{-\epsilon}\Phi^*(\Pi_{t-1}/\beta_{t-1}), \tag{9.5}$$

assuming the result of the following exercise.

Exercise 9.44. Prove that $\| \delta\Phi^*(\Pi_{t-1}/\beta_{t-1}) \|_\infty \leq 1$. (Hint: Use submultiplicativity of the spectral norm to show that $\| \Phi^*(\Pi_{t-1}) \|_\infty \leq \left\| Q^{-1} \right\|_\infty$.)

Exercise 9.45. Use Equation (9.5) and the fact that $W_{t-1} \succeq 0$ to conclude that

$$\mathrm{Tr}(W_t) \leq \mathrm{Tr}(W_{t-1})(1-\epsilon\delta e^{-\epsilon}\mathrm{Tr}\left[\rho_{t-1}\Phi^*(\Pi_{t-1}/\beta_{t-1})\right]) = \mathrm{Tr}(W_{t-1})\left(1-\epsilon\delta e^{-\epsilon}\frac{\mathrm{Tr}\left[\Phi(\rho_{t-1})\Pi_{t-1}\right]}{\beta_{t-1}}\right).$$

(Hint: Use Exercise 9.44 to justify applying Equation (9.5) to Equation (9.4), expand out the resulting expression, and then look up the definition of ρ_{t-1}.)

Combining this with the fact that by definition, $\beta_{t-1} = \mathrm{Tr}\left[\Phi(\rho_{t-1})\Pi_{t-1}\right]$, and that for all real r, $1 + r \leq e^r$, we conclude

$$\mathrm{Tr}(W_t) \leq \mathrm{Tr}(W_{t-1})e^{-\epsilon\delta e^{-\epsilon}}.$$

Exercise 9.46. Use the fact that $\mathrm{Tr}(W_0) = N$ to obtain our final upper bound,

$$\mathrm{Tr}(W_T) \leq e^{-T\epsilon\delta e^{-\epsilon}+\log N}. \tag{9.6}$$

Combining upper and lower bounds. Combining Equations (9.3) and (9.6), we have that

$$\lambda_{\min}\left(\Phi^*\left(\sum_{j=0}^{T-1}\frac{1}{\beta_j}\Pi_j\right)\right) \geq Te^{-\epsilon} - \frac{\log N}{\epsilon\delta}.$$

Exercise 9.47. Recalling that $Y = \frac{1+2\epsilon}{T}\sum_{t=0}^{T-1}\frac{1}{\beta_t}\Pi_t$, use the fact that $e^{-\epsilon} - \epsilon^2/4 > 1 - \epsilon$ to complete the proof of dual feasibility by concluding that $\lambda_{\min}(\Phi^*(Y)) > 1$. (Note the argument to Φ^* is now Y.) □

In conclusion, we have covered a rather long journey for this set of lectures notes, spanning the MW method, SDPs, interactive proofs, and finally the proof that QIP = PSPACE. Each of these is a fundamental tool or result in its own right; that they fit together to tell an elegant story is nothing short of remarkable.

10 Boson Sampling

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."
— Richard P. Feynmann

Introduction. We began in Lecture 1 by stating that the field of quantum computation is at a critical crossroads, with one of the following statements being necessarily false: The Extended Church-Turing Thesis is true, integer factorization does not have a polynomial time classical algorithm, or large-scale universal quantum computers can be built. Since this crossroads arose due to the discovery of Shor's quantum polynomial-time factoring algorithm, a natural goal is to try and show the Extended Church-Turing Thesis is false by running Shor's algorithm on a "large enough" quantum computer.

Unfortunately, there are caveats to this. First, even if Shor's algorithm could be implemented experimentally, this does not rule out the second statement — that perhaps there is an efficient *classical* algorithm for factoring. More worrisome is the fact that we are arguably not close to a functioning universal quantum computer capable of breaking today's RSA keys. For example, to a theoretician, a quantum random walk on the line is a rather basic construct; yet, implementing such a walk *efficiently* (i.e. resources scaling polynomially with the length of the walk) in an actual photonic system is highly non-trivial, requiring ideas such as *time multiplexing*.

Luckily, if our goal is to disprove the Extended Church-Turing Thesis, we do not necessarily need a *universal* quantum computer. Rather, a sufficiently restricted quantum model may still be able to solve "hard" problems, and yet be implementable on a large scale via near-term "noisy intermediate scale quantum devices" (NISQ). This quest for an experimental demonstration of quantum computational speedup has fallen under the moniker of "quantum supremacy", with multiple candidate approaches to date: The Instantaneous Quantum Polynomial-Time (IQP) model, random circuit sampling, and the deterministic quantum computation with one quantum bit (DQC1) model. Here, however, we shall focus on a framework which has elicited a particularly beautiful collaboration between the computer science and physics communities: *Boson sampling*.

Organization. We begin in Section 10.1 with an introduction to non-interacting bosonic systems. Section 10.2 describes the connection between such systems and computation of the matrix permanent. Using this background, Section 10.3 defines the Boson Sampling problem. Finally, Sections 10.3.1 and 10.3.2 discuss intractability of exact and approximate Boson Sampling for classical computers.

10.1 Of hedgehogs and photons

The basic premise of Boson sampling is to use *non-interacting Bosonic systems* to implement a computational task which is "easy" quantumly, yet provably "hard" classically. For our purposes, "bosons" will be "photons", and to begin with, we will equate "photons" with "hedgehogs".

The hedgehog model of computing. Suppose we have n identical hedgehogs, and $m \geq n$ burrows (numbered 1 to m). The hedgehog model is as follows:

1. Before nightfall, the first n burrows contain precisely 1 hedgehog each[1].
2. During the night, each hedgehog can move from its current burrow to any other. Some rules for this:
   - Parties are allowed, i.e. a burrow can host multiple hedgehogs.
   - No hedgehogs are created or destroyed in this process, i.e. we have conservation of hedgehogs.
3. When the night ends, we measure: How many hedgehogs are in each burrow?

To formalize this model, we can work in the *hedgehog number basis*, which is different from the usual standard basis for qubit systems. Namely, to specify the positions of all n hedgehogs, we use basis state

$$|S\rangle = |s_1 s_2 \cdots s_m\rangle$$

where $s_i \in \{0, \ldots, n\}$ denotes the number of hedgehogs in burrow i. The s_i are called "occupation numbers", and this basis the "occupation number basis".

Exercise 10.1. Why are we only concerned with the *number* of hedgehogs per burrow? (Hint: Which keyword used above essentially says this is the only defining characteristic of the hedgehogs?)

The set of all such valid basis states is denoted

$$\Phi_{m,n} := \left\{ (s_1, \ldots, s_m) \mid s_i \in \{0, \ldots, n\} \text{ and } \sum_{i=1}^{m} s_i = n \right\}. \tag{10.1}$$

Exercise 10.2. Why is the summation condition required in the definition of $\Phi_{m,n}$?

Of course, we're not just dealing with any old hedgehogs, but quantum hedgehogs; thus, we allow superpositions over hedgehog states:

$$|\psi\rangle = \sum_{S \in \Phi_{m,n}} \alpha_S |S\rangle,$$

where as usual $\sum_S |\alpha_S|^2 = 1$. A crucial point to note here is that unlike with m qubit systems, the vector space we are working in is *not* a tensor product of m systems of dimension n; we revisit this shortly.

From hedgehogs to photons. To move from the world of hedgehogs to photons, we make two simple substitutions: Replace the word "hedgehog" with "photon", and "burrow" with "mode" (for this lecture, a "mode" can be thought of as a spatial mode, meaning a "location" of a photon). We can now rephrase our discussion above formally in the setting of photons:

[1] Most hedgehog species are nocturnal. They are also very cute.

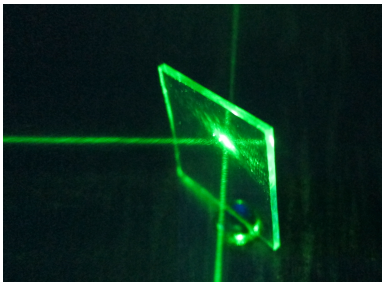


Figure 10.1: As this is the closest we will get to experiment in this course, it is worth seeing an actual piece of hardware: Depicted above is a beamsplitter designed to reflect 80% of incoming light, and transmit the remaining 20%. Intuitively, a beamsplitter implements a Hadamard gate. (Source: `https://commons.wikimedia.org/wiki/File:Flat_metal-coated_beamsplitter.png`.)

1. At the start of the experiment, our apparatus has n photons in the first n modes, and the remaining $m-n$ modes are empty, i.e. our start state is

$$|1_n\rangle := |1^n 0^{m-n}\rangle \in \Phi_{m,n}.$$

2. Formalizing the set of allowed operations (i.e. how the hedgehogs choose to switch burrows) is trickier, as we are working in a Hilbert space without a tensor product structure (in contrast to qubit systems). To see the full picture takes two steps: (1) We first look at the "idealized" case in which we have 1 photon and m modes; this will be analogous to modeling an m-dimensional qudit. The unitaries U in this case will hence be $m \times m$. (2) We then show how to map any $m \times m$ unitary U up to the full space $\Phi_{m,n}$ spans, which requires an understanding of how U acts on *multi*-photon configurations.

   Single photon configurations. Denote the subset of single photon configurations as $\Phi_{m,1} =: \{|i\rangle\} \subset \Phi_{m,n}$, i.e. $|i\rangle$ has $s_i = 1$ for some $i \in [m]$ and $s_i = 0$ otherwise. Restricted to this space, one can think of the entire system as a single m-dimensional qudit, with the ith "standard basis state" given by $|i\rangle = |0^{i-1}10^{m-i}\rangle$ (i.e. imagine encoding the basis states in unary, not binary). The set of allowed operations on this space, as expected, is the set of all $m \times m$ unitary matrices U.

   What makes optical setups appealing is that any such U can be written $U = U_T \cdots U_1$ for $T \in O(m^2)$, where each U_k is an $m \times m$ unitary or *optical element* falling into one of two classes: *Phase shifters* and *beam splitters*. These optical elements are relatively easy to implement in a lab; see Figure 10.1. Restricted to the single photon basis $\Phi_{m,1}$, each U_k acts non-trivially only on some pair of modes i and j, i.e. on unary basis states $|i\rangle$ and $|j\rangle$, and hence can be represented as a 2×2 unitary

$$U_k = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

   where the rows are labelled by $|i\rangle$ and $|j\rangle$, respectively. On $\Phi_{m,1}$, U_k acts as expected:

$$|i\rangle \mapsto a|i\rangle + c|j\rangle, \qquad |j\rangle \mapsto b|i\rangle + d|j\rangle, \qquad |k\rangle \mapsto |k\rangle \text{ for any } k \neq i, j. \tag{10.2}$$

Exercise 10.3. Consider the Pauli X_{12} gate applied to modes 1 and 2. Then, the 2×2 optical element has matrix representation (restricted to $\mathrm{Span}(|1\rangle, |2\rangle)$)

$$X_{12} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

What is $X_{12}|1\rangle$, $X_{12}|2\rangle$, and $X_{12}|k\rangle$ for $k > 2$?

Exercise 10.4. Write down the full 3×3 matrix representation for X_{12} with respect to the $\Phi_{m,1}$ basis when $m = 3$.

Restricted to this $\Phi_{m,1}$ basis, phase shifters and beamsplitters have intuitively simple representations (for $\theta \in \mathbb{R}$):

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Thus, phase shifters are essentially phase gates, and beamsplitters are analogous to Hadamard gates.

Exercise 10.5. How does a phase shifter applied to modes i and j act on basis states $|i\rangle, |j\rangle \in \Phi_{m,1}$? How about a beam splitter?

Multi-photon configurations. Focusing on single-photon configurations gave an intuitive sense of what phase shifters and beamsplitters do, but in reality our full system of n photons in m modes is not m-dimensional, but $M = |\Phi_{m,n}|$-dimensional.

Exercise 10.6. Show that $M = \binom{m+n-1}{n}$.

Given any $m \times m$ unitary U (which recall can be implemented with phase shifters and beamsplitters), we hence need a way of mapping U to the larger M-dimensional space to understand its action on *all* basis states in $\Phi_{m,n}$, as opposed to just $\Phi_{m,1}$. (In other words, how does U_k act on modes which contain multiple photons, such as $|20\rangle$?) This mapping $\varphi : \mathcal{U}(\mathbb{C}^m) \to \mathcal{U}(\mathbb{C}^M)$ turns out to be a homomorphism, meaning for us that it obeys $\varphi(U) = \varphi(U_T) \cdots \varphi(U_1)$. Thus, it suffices understand its action on the 2×2 optical elements U_k, which is:

$$\langle st|\varphi(U_k)|uv\rangle = 0 \qquad \text{if} \quad s + t \neq u + v \tag{10.3}$$

$$\langle st|\varphi(U_k)|uv\rangle = \sqrt{\frac{u!v!}{s!t!}} \sum_{\substack{p+q=u \\ p \leq s \\ q \leq t}} \binom{s}{p} \binom{t}{q} a^p b^{s-p} c^q d^{t-q} \qquad \text{if} \quad s + t = u + v. \tag{10.4}$$

Exercise 10.7. Why is Equation (10.3) equal to 0? (Hint: Which property of the hedgehogs must we preserve?)

Exercise 10.8. Setting $a = d = 0$ and $b = c = 1$, confirm that Equations (10.3) and (10.4) correctly recover the action of Pauli X when restricted to $\Phi_{m,1}$.

Exercise 10.9. How does a phase shifter act on basis state $|20\rangle \in \Phi_{2,2}$?

Exercise 10.10. What is the overlap onto $|11\rangle \in \Phi_{2,2}$ if we start with $|20\rangle \in \Phi_{2,0}$ and apply $U_k = X$? How about the overlap onto $|02\rangle$? What does this suggest intuitively about how X acts on multiple photons?

Putting it all together. In sum, given any desired $m \times m$ unitary U (including ones which will later encode hard problems), in an optical setup one can implement U by a sequence of $O(m^2)$ phase shifters and beamsplitters, and the effective action of this U on the larger M-dimensional Hilbert space is prescribed by $\varphi(U) = \varphi(U_T) \cdots \varphi(U_1)$.

3. At the end of the experiment, we measure with respect to basis $\Phi_{m,n}$ to see which modes the photons are in. Let D_U denote the distribution obtained, assuming the experiment implemented $m \times m$ unitary U. Then, the probability of observing configuration S is

$$\Pr[S \in \Phi_{m,n}] = |\langle S|\varphi(U)|1^n\rangle|^2 .$$

10.2 Connection to the matrix permanent

To now connect our optics setup to hard computational problems, we return to our hedgehog model of computing, and study a related thought experiment. In this experiment, we have n indistinguishable hedgehogs and n burrows. The rules are as follows:

1. Before nightfall, burrow $i \in [n]$ contains precisely one hedgehog, the hedgehog labelled i.
2. During the night, hedgehog i moves to burrow j with probability a_{ij}.
3. When the night ends, we ask: What is the probability that each burrow contains precisely one hedgehog?

Let us derive a formula for this probability, for which we simply have to *count* all configurations the hedgehogs could end up in. For starters, observe that the probability that the ith hedgehog remains in the ith burrow is just $a_{11} \cdots a_{nn}$.

Exercise 10.11. What is the probability that for all i, hedgehog i moves to burrow $(i \mod n)+1$?

Exercise 10.12. Show that the probability that each burrow contains precisely one hedgehog is

$$\sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} =: \mathrm{Per}(A), \tag{10.5}$$

where S_n is the set of permutations acting on n elements, and A is the $n \times n$ matrix with entries a_{ij}.

Brief aside on the permanent. The quantity in Equation (10.5) is the *permanent* of matrix A, and has seen considerable attention for at least two centuries now (being mentioned in the 1812 memoirs of Binet and Cauchy). It looks remarkably like a related quantity — the *determinant* of A, whose formula is identical except each term in the sum is multiplied by the sign of the permutation σ. Yet, these two apples most certainly did not fall from the same tree — while the determinant is efficiently computable, the permanent is #P-hard to compute exactly, even if A consists only of entries from $\{0, 1\}$. The best known general algorithm for $\mathrm{Per}(A)$ is Ryser's algorithm, which requires $\Theta(n2^n)$ arithmetic operations. We do catch a break when A has only non-negative entries: In this case, there is a *fully polynomial-time randomized approximation scheme (FPRAS)* for approximating $\mathrm{Per}(A)$, i.e. for any inverse polynomial error ϵ, there is a polynomial-time randomized algorithm outputting $\mathrm{Per}(A)$ up to relative error $(1 \pm \epsilon)$. While this setting does apply to our hedgehog model above, it will crucially *not* apply for the type of matrices which arise through boson sampling.

Connection to optics. Recall that in our optics setup, any $m \times m$ unitary U can be performed (restricted to the single photon space) via a sequence of phaseshifters and beamsplitters. The key point is that if we run our optics experiment starting in configuration $|T\rangle \in \Phi_{m,n}$ and apply U, then one can show that the probability of observing end configuration $|S\rangle \in \Phi_{m,n}$ is given by

$$|\langle S|\varphi(U)|T\rangle|^2 = \frac{|\mathrm{Per}(U_{ST})|^2}{s_1! \cdots s_m! t_1! \cdots t_m!}, \tag{10.6}$$

for $|S\rangle = |s_1 \cdots s_m\rangle$, $|T\rangle = |t_1 \cdots t_m\rangle$, and U_{ST} defined via the following two-step process (this is necessary because we must account for the action of U on multi-photon configurations via φ):

1. Map U to U_T by listing t_i copies of column i of U.
2. Map U_T to U_{ST} by listing s_i copies of row i of U_T.

This process is best demonstrated with an example, for which we set $m = 3$, $n = 2$, $S = |200\rangle$ and $T = |110\rangle$:

$$U = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \qquad U_T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad U_{ST} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

Exercise 10.13. Show that U_{ST} is an $n \times n$ matrix, for n the number of photons.

Exercise 10.14. Show that if $|T\rangle = |1^n 0^{m-n}\rangle$, then U_T is U restricted to its first n columns.

Exercise 10.15. Show that if $|S\rangle = |T\rangle = |1^n 0^{m-n}\rangle$, then U_{ST} is the upper left $n \times n$ principal submatrix of U, i.e. the submatrix obtained by keeping the first n columns and rows of U.

In sum, if one could write down output probabilities of our photonic experiment, then one could compute permanents of matrices U_{ST}. In particular, as the last exercise above suggests, when $|S\rangle = |T\rangle = |1^n 0^{m-n}\rangle$, this boils down to the permanent of whichever $n \times n$ matrix A we are able to embed in the top-left block of U. Of course, there are some important questions to

be answered: Which types of matrices can we embed into unitaries in such a fashion? How do we convert the ability to sample to the ability to estimate output probabilities? Can an experimental quantum optics device, which will inherently be subject to noise and imperfection, itself perform such estimation? These questions, along with the connection between the permanent and photonics setups, are the starting point of boson sampling.

10.3 Boson Sampling

We are now in position to semi-formally define the task of Boson Sampling.

Definition 10.16 (Boson Sampling)**.**

- *Input: An $m \times m$ unitary matrix U.*
- *Output: Define distribution D_U as follows. Starting in configuration $|1^n\rangle \in \Phi_{m,n}$, we imagine running the optics setup outlined in Section 10.1 with unitary U. For any configuration $S \in \Phi_{m,n}$, the probability of observing output configuration S is*

$$\Pr_{D_U}[S] = \frac{|\mathrm{Per}(U_{S,1^n})|^2}{s_1! \cdots s_m!}.$$

The output of Boson Sampling is to sample configurations according to D_U.

Two remarks are in order: First, unlike all other computational problems we have seen in this course, Boson Sampling is *not* a decision or promise problem; rather, it is a *sampling* problem (i.e. the output is not a single bit, but a random sample over strings). Second, the "semi-formal" aspect of the definition above is that we have not specified the *precision* to which the sampling must be done (i.e. are we required to sample from D_U exactly? Approximately? Within what type of error?). These distinctions are crucial, and are discussed in the next two sections.

10.3.1 The exact case

The strongest variant of Boson Sampling would be to require the sampling to be *perfect* — i.e. one outputs samples from $\Phi_{m,n}$ exactly according to D_U. This is not experimentally realistic, as any physical setup is subject to noise. Nevertheless, in this case one can rigorously show the following result.

Theorem 10.17 (Exact classical Boson Sampling)**.** *Suppose there is a classical algorithm A which, given any $m \times m$ unitary U, solves the Boson Sampling problem exactly. We make no assumptions about A (e.g. it could be a* P*,* BPP*, or* PH *machine). Without loss of generality, we may view A as a deterministic machine which is fed a uniformly random string of bits r. By assumption, the sample produced by $A(U, r)$ is distributed exactly according to D_U. Then,*

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{NP}^A}.$$

Deer in the headlights: Interpreting Theorem 10.17 . Theorem 10.17 is a bit stunning at first sight, so let us carefully unpack it.

- What it *does* say is that if A is a BPP machine (or even a PH machine!), then

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{NP}^{\mathrm{BPP}}} \subseteq \mathrm{PH},$$

which would collapse PH. Thus, it is highly unlikely that exact Boson Sampling is efficiently simulatable classically.

- What it *does not* say is anything about whether a quantum computer can perform exact Boson Sampling. And therein lies the magic of the theorem — Theorem 10.17 does *not* prove that exact Boson Sampling is #P-hard. Rather, it shows that if there is an efficient *classical* algorithm for Boson Sampling, then PH collapses.

  Exercise 10.18. Why would it be presumably "bad" if Theorem 10.17 actually showed Boson Sampling is #P-hard? (Hint: What would this say about the ability of quantum computers to solve Boson Sampling?)

  The way Theorem 10.17 accomplishes this feat is by exploiting the fact that the randomness r in any classical machine A can be "extracted" from it. In other words, a classical algorithm A is without loss of generality deterministic, up to an input string of uniformly random bits r.

  Exercise 10.19. Why is it not clear how to similarly "extract the randomness" out of a quantum computation?

- While theoretically interesting, Theorem 10.17 unfortunately does not suffice to rule out the Extended Church Turing thesis, as even an optical setup realistically cannot perform exact Boson Sampling due to experimental noise. Thus, we must consider *approximate* Boson Sampling.

10.3.2 The approximate case

While things work out neatly in the exact case, the approximate case (which is the relevant one) is much messier; in particular, we do not have a rigorous statement along the lines of Theorem 10.17. Rather, there is a two-step agenda in place, the second part of which currently relies on (arguably reasonable) conjectures:

1. **What is currently proven:** If one can classically simulate "approximate" Boson Sampling, then one could compute the permanent "approximately" and "on average" (i.e. for "most inputs").
2. **What relies on conjecture:** Computing the permanent "approximately" and "on average" is #P-hard.

Taken together, this agenda would yield that efficient classical simulation (i.e. in BPP) of "reasonable" Boson Sampling setups (i.e. allowing reasonable error, and taking into account "average-case" inputs as opposed to extremal worst-case inputs[2]) is likely impossible. Again, the agenda does *not* imply that simulating Boson Sampling approximately and on average is #P-hard, but rather that any *classical* algorithm for such a simulation can be bootstrapped to solve #P-hard problems. Finally, let us stress that it is not clear whether even a *quantum* computer can solve approximate Boson Sampling on average — the biggest challenge is arguably the requirement for single photon sources (i.e. to prepare the initial state $|1^n\rangle$). Addressing this question is not the purpose of today's lecture.

[2]This distinction is crucial. A classic example is the canonical NP-complete problem 3-SAT; while intractable in the worst case, many (if not most) instances of 3-SAT can be solved efficiently in practice using heuristics.

Formalizing the agenda. In the remainder of this lecture, we shall sketch how Step 1 of the agenda above is formalized and shown. The computational problem capturing approximate permanent computation on average is the following.

Definition 10.20 (Gaussian Permanent Estimation (GPE))**.**

- *Input: (1) A random matrix $X \in \mathcal{L}(\mathbb{C}^n)$, each entry of which is distributed independently according to the standard Gaussian distribution $\mathcal{N}(0,1)$. (2) Precision parameter $\epsilon > 0$, specified in unary. (3) Success probability parameter $\delta > 0$, specified in unary.*
- *Output: With probability at least $1 - \delta$ (with respect to the randomness in choosing X), output a value $z \in \mathbb{R}$ satisfying*

$$|\mathrm{Per}(X)|^2 - \epsilon n! \leq z \leq |\mathrm{Per}(X)|^2 + \epsilon n!.$$

Exercise 10.21. Which parameter above captures the notion of solving for the permanent "approximately"? Which parameter captures "on average"?

The main theorem of this lecture is the following, which states that if efficient classical simulation of approximate Boson Sampling on average is possible, then GPE is in PH.

Theorem 10.22 (Main Theorem)**.** *Let D_U be the Boson Sampling distribution from Definition 10.16 for $m \times m$ input unitary U. Suppose there exists a classical algorithm A which, given precision parameter $\epsilon > 0$ in unary, outputs a sample from distribution D'_U such that $|D_U - D'_U| \leq \epsilon$ in time polynomial in m and $1/\epsilon$. Then,*

$$\mathrm{GPE} \in \mathrm{BPP}^{\mathrm{NP}^A}.$$

For clarity, $|D_U - D'_U|$ denotes the total variation distance between distributions D_U and D'_U.

Let us also formalize what we mean by a "classical algorithm A" in Theorem 10.22 above. Roughly, we shall think of A as an approximate Boson Sampling *oracle*, i.e. we will not care about its internal workings (other than it being deterministic), but just its input/output behavior.

Definition 10.23 (Approximate Boson Sampling oracle)**.** *A deterministic algorithm A, to be treated as an oracle, which takes in as input:*

- *an $m \times m$ unitary matrix U,*
- *a precision parameter $\epsilon > 0$ encoded in unary,*
- *and "random" string $r \in \{0,1\}^{\mathrm{poly}(n)}$.*

Let $D_A(U, \epsilon)$ denote, for any fixed U and ϵ, the distribution over outputs of A when r is uniformly random. Then, A outputs samples from $\Phi_{m,n}$ distributed according to distribution $D_A(U, \epsilon)$ such that, for all U, ϵ,

$$|D_A(U, \epsilon) - D_U| \leq \epsilon.$$

Proof of Theorem 10.22

We are now ready to move to the final stage of this lecture; giving a proof sketch of Theorem 10.22. Again, let us stress that this theorem only says that classical simulation of approximate Boson Sampling solves a problem related to computation of the permanent, GPE. It does *not* tell us whether GPE is hard to begin with (this would be the job of Step 2 of the agenda of Section 10.3.2, which currently relies on conjectures), nor does it say anything about whether quantum computers can simulate approximate Boson Sampling.

Proof sketch of Theorem 10.22. Let X, ϵ, δ be inputs to GPE, where recall X is $n \times n$. We wish to bootstrap a (classical) approximate Boson Sampling oracle A to approximate $|\mathrm{Per}(X)|^2$ within additive error $\pm\epsilon n!$, with success probability at least $1 - \delta$ over the random choice of X, in class $\mathrm{FBPP}^{\mathrm{NP}^A}$. (To be technically correct, we use the base class FBPP here in place of BPP; the former is the function analogue of BPP which is allowed to output strings longer than length 1.) For this, we will need two technical ingredients:

1. The Hiding Lemma.
2. Stockmeyer's approximate counting algorithm.

We will introduce these when the time comes; for now, let us begin with the "naive" approach for solving GPE using A.

The "naive" approach. As suggested at the end of Section 10.2, we will attempt to embed $n \times n$ matrix X in the top left corner of an $m \times m$ unitary U, so that simulating Boson Sampling on U will output configuration $|1^n\rangle \in \Phi_{m,n}$ with probability precisely $|\mathrm{Per}(X)|^2$. This gives us the ability to sample according to $|\mathrm{Per}(X)|^2$ — to then convert this into the ability to estimate the scalar $|\mathrm{Per}(X)|^2$ itself, we apply Stockmeyer's algorithm.

Exercise 10.24. Given the ability to run an experiment which accepts with probability $0 < p < 1$, what is the naive way to estimate the scalar p? Why does this approach not necessarily work to estimate $|\mathrm{Per}(X)|^2$ above (i.e. why do we seem to need Stockmeyer's algorithm)?

To begin, recall that X is $n \times n$. Let our Boson Sampling setup have n photons and $m \gg n$ modes (e.g. $m = O(\frac{1}{\delta} n^5 \log^2 \frac{n}{\delta})$ suffices). By rescaling our input as $X' := X/\sqrt{m}$, our task is to estimate

$$\left|\mathrm{Per}(X')\right|^2 = \frac{1}{m^n} |\mathrm{Per}(X)|^2$$

within additive error $\epsilon n!/m^n$. We proceed as follows:

1. Embed X' as the top-left $n \times n$ principal submatrix of a unitary U (we ignore how this is done for the moment).
2. Run the Boson Sampling oracle A with input (U, ϵ, r) for uniformly random r and inverse polynomial ϵ. This allows us to sample from distribution D'_U such that $|D_U - D'_U| \leq \epsilon$. Now, if it were true that $D_U = D'_U$, then we would be in luck, since the probability of observing precisely one photon in each of the first n modes is

$$\mathrm{Pr}_{D_U}[1^n] = \frac{|\mathrm{Per}(U_{1^n,1^n})|^2}{(1!)^n (0!)^{m-n}} = \left|\mathrm{Per}(X')\right|^2 .$$

   Given the ability to sample outcome 1^n with probability $|\mathrm{Per}(X')|^2$, we can now convert this to the ability to approximate the scalar $|\mathrm{Per}(X')|^2$ itself via the following theorem.

Theorem 10.25 (Stockmeyer's approximate counting)**.** *Given Boolean function $f : \{0,1\}^n \to \{0,1\}$, let p be the probability that a uniformly random input x causes f to accept, i.e.*

$$p = \Pr_{x \in \{0,1\}^n}[f(x) = 1] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x).$$

For any error tolerance $g \geq 1 + \frac{1}{\text{poly}(n)}$, p can be estimated within multiplicative error g in $\text{FBPP}^{\text{NP}^f}$.

Exercise 10.26. In an earlier exercise, we said the naive approach for exploiting the ability to run an experiment which accepts with probability $0 < p < 1$ into an estimation of p itself did not work. Why does Stockmeyer's algorithm get around this problem? (Hint: Does Stockmeyer's algorithm restrict p in any essential way?)

Reflection. It is now clear why Theorem 10.22 works only for *classical* algorithms; namely, any classical oracle implementation is just a Turing machine encoding a Boolean function $f : \{0,1\}^n \to \{0,1\}$ (up to additional inputs to the machine, such as the random string r). This allows us to apply Stockmeyer's algorithm to count the number of satisfying assignments to f, and hence to estimate p. If A were instead to be *quantum*, an analogous statement is not known[3].

The problem with the naive approach. In our discussion of Stockmeyer's algorithm above, we assumed the Boson Sampling oracle is *perfect*, i.e. $D_U = D'_U$. However, recall that in our setup, A only satisfies $|D_U - D'_U| \leq \epsilon$. So we must ask: *How badly does this error affect our desired sampling outcome, $S = 1^n$?* Intuitively, since $\epsilon \in O(1/\text{poly}(n))$, and since there are exponentially many (i.e. $\binom{m}{n}$ for $m \gg n$) possible experimental outcomes or configurations in $\Phi_{m,n}$, the *expected* error per configuration is exponentially small in n. However, we are not interested in *most* configurations — we are only interested in the output configuration $S = 1^n$. And it is entirely possible, since we make no assumptions about A, that *all* of the ϵ sampling error A makes is concentrated on this one outcome 1^n we care about (i.e. all other outcomes $S' \neq 1^n \in \Phi_{m,n}$ are sampled perfectly by A). This is a huge problem — $\Pr_{D_U}[1^n]$ could be exponentially small in n, whereas ϵ is as large as inverse polynomial in n, potentially wiping out our estimate of the former.

The final ingredient: The Hiding Lemma. To resolve this problem, let us revisit why the sampling outcome $S = 1^n$ encoded the permanent of X' in the first place.

Exercise 10.27. Argue that $S = 1^n$ encodes $|\text{Per}\, X'|^2$ only because we embedded X into the top-left $n \times n$ principal submatrix of unitary U. In other words, suppose we instead embed X (e.g.) into rows 2 to $n+1$ and columns 1 to n of U — which output configuration $S \in \Phi_{m,n}$ would now encode $|\text{Per}\, X'|^2$?

As the exercise above suggests, the output configuration S we care about depends entirely on *where* in the matrix U we embed X'. Thus, if oracle A makes a large error on output

[3]Intuitively, the problem quantumly is that "computational paths" can destructively interfere and cancel out, so that the acceptance probability p of a quantum circuit is no longer the sum of a set of non-negative numbers, but the sum of both positive and negative numbers.

configuration 1^n, no problem — we simply encode X' elsewhere in U. Of course, *a priori* we have no idea on which configurations A makes an error. So the obvious thing to do is to embed X' in a *random* $n \times n$ submatrix of U. How to implement this, however, is not obvious — in particular, we need to do the embedding cleverly so that U looks completely random to A (i.e. A should have no way of distinguishing where in U the submatrix X' was hidden). By doing so, we may argue that *even if A acts adversarially*, it cannot corrupt the particular output configuration we care about with non-negligible probability. Thus, the error incurred by A can be neglected with high probability, and we can then apply the "naive" approach using Stockmeyer's algorithm above.

This "hiding/embedding" trick is accomplished by the Hiding Lemma.

Lemma 10.28 (Hiding Lemma)**.** *Fix n, $\delta > 0$, and $m \gg n$ (e.g. $m \in \Theta(\frac{1}{\delta} n^5 \log^2 \frac{n}{\delta})$ suffices). There exists a $\mathrm{BPP}^{\mathrm{NP}}$ algorithm B which, given an $n \times n$ matrix X with entries independently and identically distributed as $\mathcal{N}(0,1)$, behaves as follows: B succeeds with probability $1 - O(\delta)$ over the choice of X, and conditioned on succeeding, outputs a random $m \times m$ unitary U according to a distribution D_X, such that both of the following hold:*

1. *Assuming B succeeds with non-zero probability on X, we have that $X' = X/\sqrt{m}$ occurs as a uniformly random $n \times n$ submatrix of U.*

2. *The matrix U looks "uniformly random". (Formally, the distribution over $m \times m$ matrices produced by first drawing X, and then conditioning on B succeeding on input X, is precisely the Haar measure over $m \times m$ unitaries.)*

In words, the Hiding Lemma does exactly what we need — property (1) states that the matrix for which we wish to compute the permanent, X', is embedded in a *random* location of unitary U. If we could assume the approximate Boson Sampling oracle A is "honest", this alone might suffice. However, since we cannot assume anything about A, we have our failsafe — property (2) says that not only is X' randomly embedded into U, but that there is information theoretically no way to tell, given U alone, *where* X was hidden. (Here we are implicitly using the fact that if two distributions, or more generally density operators, have trace distance 0, then there is no possible measurement which distinguishes these operators with probability better than random guessing.) Thus, it can be shown that the $\epsilon \in O(1/\operatorname{poly}(n))$ error incurred by our approximate Boson sampling oracle is highly unlikely to affect the particular output configuration $S \in \Phi_{m,n}$ which corresponds to where X' was hidden. Thus, we can apply Stockmeyer's algorithm to the choice of S output by the Hiding Lemma to estimate X', yielding the claim of Theorem 10.22.

We close by remarking that the proof of the Hiding Lemma is rather technical, and thus omitted for the purposes of this lecture. However, it is not entirely surprising that random submatrices of Haar-random unitaries look "Gaussian" — a standard approach for sampling Haar-random unitaries is roughly to begin by picking all entries as i.i.d. Gaussian entries, and then adjusting all columns to be orthonormal via the Gram-Schmidt procedure. Indeed, this is precisely one of the reasons that GPE is defined according to Gaussian instances to begin with. (Of course, the proof of the Hiding Lemma remains non-trivial.)

□

11 BQP versus the Polynomial Hierarchy

> *"And it is also said," answered Frodo: "Go not to the Elves for counsel for they will answer both no and yes." "Is it indeed?" laughed Gildor. "Elves seldom give unguarded advice, for advice is a dangerous gift, even from the wise to the wise, and all courses may run ill."*
> — J. R. R. Tolkien, The Fellowship of the Ring

Introduction. In Assignment 3, we showed the Sipser-Gács-Lautemann Theorem, which stated that BPP $\subseteq$ PH (more precisely, BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$). A natural question is thus: Could BQP $\subseteq$ PH as well?

At first glance, this question appears wholly unconnected to our study of Boson Sampling from Chapter 10. A closer look, however, reveals similar magic at play: BPP $\subseteq$ PH is shown by leveraging the fact that randomness can be "extracted" from a randomized Turing machine; in this sense, a randomized Turing machine can be viewed as deterministic, instead taking a uniformly random string as input. Whether the inherent randomness in *quantum* circuits can similarly be extracted, however, is entirely unclear. Indeed, it was precisely this distinction between the classical and quantum models which was exploited in Boson Sampling to argue that if a *classical* (but not quantum!) algorithm could solve the approximate Boson Sampling problem, then PH is at risk[1] of collapsing. It hence seems this distinction between "classical and quantum randomness" has an important role to play in delineating the power of classical versus quantum computation. Indeed, for this reason, the techniques of the Sipser-Gács-Lautemann Theorem break down in the quantum setting, and BQP is generally believed *not* to lie in PH, in contrast to BPP.

Of course, proving BQP $\not\subseteq$ PH is difficult. In this lecture, we discuss arguably the next best thing: "Evidence" that BQP is not in PH in the form of an *oracle separation* between the classes. The word evidence is in quotes here, as recall oracle separations are *not* necessarily reliable evidence that a pair of classes are distinct. For example, there exist oracles A and B relative to which one can rigorously prove $\mathrm{P}^A = \mathrm{NP}^A$ and yet $\mathrm{P}^B \neq \mathrm{NP}^B$. This is the meaning of the opening quote of this lecture — oracle separations are like Elves; they may "answer" both no and yes, and it is not clear what the correct answer should be. (As an aside, what oracle separations *do* rigorously show is that any separation proof between (in this case) P and NP must be non-relativizing, i.e. must break down when oracles are added to the picture.) Nevertheless, as separating classes is typically difficult, oracle separations are often viewed as a desirable first step in this direction.

Organization. We begin in Section 11.1 by stating and parsing the key claim on which the lecture rests. This includes fleshing out a connection between bounded depth circuits and alternating quantifiers in Section 11.1.1. The remainder of the lecture shows the key claim: Section 11.2 states the distribution D required for the claim, and Sections 11.3 and 11.4 sketch

[1] We say "at risk" of collapsing, as recall in the setting of approximate Boson Sampling, part of the research agenda currently relies on conjectures.

its proof. This lecture uses some useful tools such as Fourier analysis to study Boolean functions, which are worth delving into in their own right. A nice reference for the latter is `https://arxiv.org/abs/1205.0314` (Analysis of Boolean Functions, lecture notes of Ryan O'Donnell with a guest lecture by Per Austrin, scribed by Li-Yang Tan).

11.1 The key claim

We begin by stating and parsing the key claim on which the lecture rests. Throughout this lecture, we set $N := 2^n$ for n the input parameter of interest (i.e. N is exponentially large).

Theorem 11.1. *Let U_{2N} denote the uniform distribution over $\{\pm 1\}^{2N}$. There exists an explicit distribution D over $\{\pm 1\}^{2N}$ satisfying both of the following:*

1. *(Distinguishing is easy quantumly) There exists a quantum algorithm able to distinguish U_{2N} from D*
   - *with advantage $\Omega(1/\log N)$, and*
   - *in time $O(\log N)$ using 1 input query.*
2. *(Distinguishing is hard classically) Any Boolean circuit of size quasipolynomial in N and of constant depth cannot distinguish U_{2N} from D with advantage better than $O(\mathrm{polylog}(N)/\sqrt{N})$.*

Exercise 11.2. An important step in digesting technical material is to understand what you don't understand — which terms in Theorem 11.1 above require clarification in order to make the theorem a formal statement?

As suggested by the exercise above, there are a few terms here which require clarification:

- *What is a Boolean circuit in this context?* By "Boolean circuit", we mean a classical circuit consisting of unbounded fan-in[2] AND and OR gates, as well as NOT gates. The circuit has a single output wire. Its *size* is the number of gates, and its *depth* is the length of the longest path in edges from any input wire to the output wire. (Here, we are implicitly viewing the circuit as a directed acyclic graph from input wires to output wires.)
- *What do we mean by "advantage"?* Let D, D' be distributions over a finite set X. Then, we say a (classical or quantum) algorithm A distinguishes between D and D' with advantage ϵ if
$$|\Pr_{X\sim D}[A \text{ accepts } x] - \Pr_{X\sim D'}[A \text{ accepts } x]| = \epsilon.$$
- *How are the distributions U_{2N} and D accessed?* We stated above that N is exponentially large in n; thus, each random sample $x \in \{\pm 1\}^{2N}$ is an exponentially large string. To make this a meaningful input model, we hence grant algorithms *oracle access* to string x:
  - A classical algorithm is allowed to perform the mapping $i \mapsto x_i$ for unit cost, for $i \in [\log(2N)]$ and x_i the ith bit (in the ± 1 basis) of x.
  - A quantum algorithm is allowed to coherently perform the mapping $|i\rangle \mapsto x_i|i\rangle$ for unit cost. Here, we are implicitly using phase kickback to inject $x_i \in \{\pm 1\}$ as a phase. The mapping also works with any "garbage" in an ancilla register, i.e. $|i\rangle|g\rangle \mapsto x_i|i\rangle|g\rangle$ for any state $|g\rangle$.

[2] By unbounded fan-in, we mean each AND and OR gate can have multiple input wires, as opposed to just 2.

- *What counts as "quasipolynomial" size in N?* Typically, quasipolynomial in N refers to quantities such as $O(2^{\log^c N})$ for constant c.

- *What in the world does Theorem 11.1 have to do with* PH*?* As the title of this lecture suggests, we are interested in giving an oracle-based task which can be solved in BQP but not in PH. Yet, the statement of Theorem 11.1 says nothing about PH or even alternating quantifiers — rather, it is a no-go statement for quasipolynomial size bounded depth circuits. The missing link between these two ideas is worthy of a discussion in its own right, which we now move to.

11.1.1 The connection between bounded depth circuits and alternating quantifiers

We first recall the definition of PH.

Definition 11.3 (Polynomial Hierarchy (PH) and Σ_k^p)**.** *A language $L \subseteq \{0,1\}^*$ is in Σ_k^p, if there exists a polynomial-time uniformly generated Turing machine M which takes in input $x \in \{0,1\}^n$, k proofs $y_1, \ldots, y_k \in \{0,1\}^{n^c}$ for $c, k \in O(1)$, and acts as follows:*

$$x \in L \quad \Rightarrow \quad \exists y_1 \forall y_2 \exists y_3 \cdots Q_k y_k \text{ such that } M \text{ accepts } (x, y_1, \ldots, y_k), \tag{11.1}$$

$$x \notin L \quad \Rightarrow \quad \forall y_1 \exists y_2 \forall y_3 \cdots \overline{Q}_k y_k \text{ such that } M \text{ rejects } (x, y_1, \ldots, y_k), \tag{11.2}$$

where $Q_k = \exists$ ($Q_k = \forall$) if i is odd (even). We define $\mathrm{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_k^p$*, where* $\Sigma_0^p = \mathrm{P}$.

In the setting of *oracle* problems, there is a slick connection between bounded depth circuits and computations of the above form, which we state and prove in full generality below. To set this up, fix $N = 2^n$ for input parameter n, and consider an arbitrary Boolean function $f : \{0,1\}^N \to \{0,1\}$. Since any input $x \in \{0,1\}^N$ to f is exponentially long, we assume we are given only access via some oracle O_x to x, i.e. the ability to map $i \mapsto x_i$ for unit cost for any $i \in [N]$.

Lemma 11.4. *Suppose there exists a Σ_k^p machine M which is given as input (1) a unary string 1^n and (2) $x \in \{0,1\}^N$ given implicitly via access to oracle O_x, and computes output $f(x) \in \{0,1\}$. Then, there exists a Boolean circuit of depth $k+1$ and size $O(2^{\log^{c'} N})$ for some $c' \in O(1)$ which, given access to oracle O_x, computes $f(x)$.*

It is worth stressing above that M only receives 1^n as an explicit input; the "actual" input x on which f is to be evaluated is "stored off-site" in the oracle O_x.

Proof of Lemma 11.4. Let M be the Σ_k^p machine computing $f(x)$ with oracle access to $x \in \{0,1\}^N$. Without loss of generality, we may assume M makes only a single query to x, as per the following exercise.

Exercise 11.5. Show that by adding two additional alternating quantifiers, one can simulate M making a polynomial number of calls to f with some M' making only a single call to f. (Hint: Think about computational paths which branch each time an oracle query answer bit is received. Use the $\exists$ and $\forall$ quantifiers to "pick out and enforce" the "correct" computational branching process.) Can you reduce it to requiring just one additional alternating quantifier?

High-level idea. We shall build an AND-OR tree T whose nodes are unbounded fan-in AND and OR gates. The leaves are the input bits; by applying the AND and OR gates each time we move up a level from the leaves, we arrive at the root, which shall be an OR gate. The output bit of the root shall be 1 if and only if $f(x) = 1$. The depth of the tree shall be $k + 1$, and its size $O(2^{\log^{c'} N})$; hence, we will have a circuit computing $f(x)$ with properties stated in the claim.

The construction. We view the action of M via its computational branches. Let us start with the first existentially quantified proof, $y_1 \in \{0,1\}^{n^c}$. This induces a branching in M over 2^{n^c} computational paths (one per possible proof y_1), and M accepts if at least one of these branches accepts. Hence, in T we "represent y_1" via an OR gate at the root which takes in 2^{n^c} wires and outputs a single wire. We can now recursively apply the same idea for each successive proof y_i, except whenever we have $Q_i = \forall$, we instead put in an AND gate into T (as opposed to an OR gate for $Q_i = \exists$). Finally, each leaf of T is reached by fixing a sequence of proofs $y_1, \ldots, y_k$. At any such leaf, we may assume M makes its single query to O_x, and subsequently decides to accept or reject. Specifically, M evaluates some polynomial-time function $g(y_1, \ldots, y_k) : (\{0,1\}^{n^c})^{\times k} \to [N]$ to obtain the index $i \in [N]$ on which it will query O_x. Upon obtaining x_i, it performs some final polynomial-time computation $g'(y_1, \ldots, y_k, x_i) : (\{0,1\}^{n^c})^{\times k} \times \{0,1\} \to \{0,1\}$ to decide whether to accept or reject. Equivalently, this can be viewed as: Conditioned on $y_1, \ldots, y_k$, M either returns x_i or $\overline{x_i}$.

Exercise 11.6. There is another option above: M could return a constant value independent of x_i. Why can we ignore this case without loss of generality? (Hint: Do we need a query to O_x in this case? If no query is needed, how can we trivially modify the tree and corresponding circuit to eliminate this leaf altogether?)

Now, if M returns x_i at this leaf, then the corresponding circuit simply reads input bit x_i here via a query to O_x. If M instead returns $\overline{x_i}$ at this leaf, the corresponding circuit first reads input x_i via O_x, and subsequently applies a NOT gate (which is also viewed as a node in T with one input and one output wire). This completes the construction.

Exercise 11.7. Prove that the tree T constructed has depth $k + 1$ and size $O(\sum_{i=0}^{k} (2^{n^c})^i) \in O(2^{\log^{c'} N})$ for some $c' \in O(1)$. Conclude there is a bounded depth circuit as stated in the claim which computes $f(x)$ given oracle access to x. □

With Lemma 11.4, we can close our discussion of the key claim of the lecture via the following exercise.

Exercise 11.8. Use Lemma 11.4 to answer our earlier question, "What in the world does Theorem 11.1 have to do with PH?". In other words, show that Theorem 11.1 implies that for any $k \in O(1)$, no Σ_k^p machine can distinguish between U_{2N} and D with advantage better than $O(\text{poly}(N)/\sqrt{N})$.

11.1.2 Outline for lecture

With Theorem 11.1 in place, the remainder of the lecture proceeds as follows:

- Specify the distribution D.

- Show that distinguishing D from U_{2N} is easy quantumly.
- Show that distinguishing D from U_{2N} is hard classically.

11.2 The distribution D

The distribution D in Theorem 11.1 is defined via a two-step process as follows. Set $n \in \mathbb{N}$ as our input parameter, $N = 2^n$, and $\epsilon = 1/(24\ln(N))$ (the precise value of ϵ is not relevant for our lecture, only that $\epsilon \in \Theta(1/\operatorname{poly}(n))$.

Step 1: Define a distribution G' over continuous space $\mathbb{R}^N \times \mathbb{R}^N$.

1. (Sample the first N real numbers) Sample $x_1, \ldots, x_N \in \mathbb{R}$ independently, each according to $\mathcal{N}(0,1)$. (This can roughly be viewed as choosing a Haar random vector in $\mathbb{R}^N$.) Denote $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$.
2. (Correlate the second N real numbers with x) Observing that $x \in \mathbb{R}^{2^n}$ is a column vector, set $y = H^{\otimes n}x$ for H the 2×2 Hadamard gate.
3. (Final output) Output vector $z = \sqrt{\epsilon}(x, y) \in \mathbb{R}^{2N}$.

Note that since ϵ is "small", with high probability $-1 \leq z_i \leq 1$. For simplicity in this lecture, we henceforth assume that indeed $-1 \leq z_i \leq 1$ for all $i \in [2N]$. (One can deal with z_i violating this via a further "truncation step", which complicates the analysis and does not affect its core intuition; we omit this here.)

Step 2: "Round" G' to a distribution D over discrete space $\{\pm 1\}^N \times \{\pm 1\}^N$. The distribution G' is over $\mathbb{R}^{2N}$, but Theorem 11.1 requires a distribution D over $\{\pm 1\}^{2N}$. Since we are assuming all $z_i \in [-1, 1]$, we can perform such a mapping to $\{\pm 1\}^{2N}$ using a now-standard idea in approximation algorithms; namely, "snap" z_i to whichever of -1 or 1 it is closer to with probability proportional to $|z_i|$. Formally:

1. (Snap to the Boolean hypercube) Independently for each $i \in [2N]$, set
$$z'_i := \pm 1 \text{ with probability} \frac{1 \pm z_i}{2}.$$
2. (Final output) Output the resulting string $z' \in \{\pm 1\}^N \times \{\pm 1\}^N$.

Brief intuition. The main idea behind the choice of G' is that "Gaussian distributions are nice to work with". Thus, ideally we would like to analyze G' rather than this clunky "discrete" object D, which is necessary mainly due to the query model (e.g. recall a quantum query injects $z'_i \in \{\pm 1\}$ as a phase). Luckily, due to the choice of D, it turns out that as far as expectation values are concerned, both D and G' are "equivalent" in the following sense.

Exercise 11.9. Prove that $E[z'_i] = z_i$ for all $i \in [2N]$, where recall z'_i (z_i) are the discrete (continuous) coordinates.

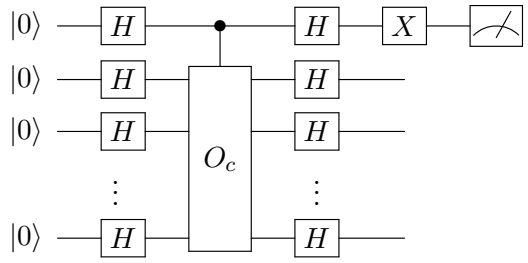


Figure 11.1: The quantum circuit V_n for distinguishing D from U_{2N}.

11.3 Distinguishing D from U_{2N} is easy quantumly

The construction. Given oracle access to input $z' \in \{\pm 1\}^{2N}$, our goal is to decide whether z' was drawn according to D or U_{2N}. To do this quantumly, we imagine that the oracle for accessing z', denoted O_z, is split into two oracles O_x and O_y, such that for any $i \in [N]$,

$$(\text{“x part”}) \quad O_x|i\rangle \mapsto z'_i|i\rangle \qquad \text{and} \qquad (\text{“y part”}) \quad O_y|i\rangle \mapsto z'_{N+i}|i\rangle.$$

Exercise 11.10. Show how to implement O_x and O_y given O_z.

The quantum circuit for distinguishing D from U_{2N} is given in Figure 11.1. It is denoted V_n, since it acts on $n+1$ wires; the first wire is a control denoted c, and wires 2 to $n+1$ are fed to the oracle O_c. The control mechanism is as follows: If $c = 0$, then O_c applies O_x, and if $c = 1$, O_c applies O_y.

Correctness. For the remainder of this section, set $z' = x'y'$ for $x', y' \in \{\pm 1\}^N$. The following theorem shows that V_n works as intended.

Theorem 11.11. *Suppose circuit V_n of Figure 11.1 outputs 1 when it measures 1 and outputs -1 when it measures 0 (i.e. we are measuring observable $-Z$). Then, the expected output of V_n on $z' \sim U_{2N}$ (respectively, $z' \sim D$) is 0 (respectively, ϵ).*

Exercise 11.12. Theorem 11.11 is in terms of expectation, but we defined “advantage” in Theorem 11.1 using probabilities. Why does the former immediately imply the quantum advantage claimed in Theorem 11.1? (Hint: No repetition of the protocol is needed. Use the fact that V_n has only two possible outcomes, ± 1.)

Proof of Theorem 11.11. We proceed in three steps.

Step 1: The probability with which V_n outputs 1.

Exercise 11.13. Show that V_n measures 1 in the control qubit with probability

$$\frac{1}{2}\left(1 + \left(\frac{1}{\sqrt{2^{3n}}} \sum_{i,j\in[N]} (-1)^{i\cdot j} x'_i y'_j\right)\right) = \frac{1}{2}\left(1 + \left(\frac{1}{N} \sum_{i,j\in[N]} (H^{\otimes n}(i,j)) x'_i y'_j\right)\right) =: \frac{1}{2}(1+\varphi(x',y')),$$

where $i \cdot j$ denotes the inner product modulo 2 of the bit strings i and j, and $H^{\otimes n}(i,j)$ is the (i,j)th entry of $H^{\otimes n}$. To characterize the acceptance probability of V_n, it hence suffices to analyze the multilinear polynomial $\varphi(x', y')$.

Exercise 11.14. Why is φ multilinear?

Step 2: Expected output of V_n for $z' \sim U_{2N}$.

Exercise 11.15. Prove that $E_{(x',y')\sim U_{2N}}[\varphi(x', y')] = 0$. (Hint: There are three tools in mathematics which should always be at the top of your toolbox; the Cauchy-Schwarz inequality, Taylor series, and the linearity of expectation.)

We hence conclude that when $z' \sim U_{2N}$, the circuit V_n outputs each possible answer with probability 1/2.

Step 3. Expected output of V_n for $z' \sim D$. Recall in Section 11.2 that we had the gall to call discrete-valued distribution D "clunky", but the multivariate Gaussian distribution G' "nice". Here we will see why. First, we claim that

$$E_{(x',y')\sim G'}[\varphi(x', y')] = \epsilon, \tag{11.3}$$

where note we are using G'.

Exercise 11.16. Prove that for any $i, j \in [N]$, $E_{(x',y')\sim G'}[x'_i y'_j] = \epsilon \frac{(-1)^{i \cdot j}}{\sqrt{N}}$.

Via the exercise above and the linearity of expectation, the claim of Equation (11.3) follows:

$$E_{(x',y')\sim G'}[\varphi(x', y')] = \frac{1}{N} \sum_{i,j \in [N]} \frac{(-1)^{i \cdot j}}{\sqrt{N}} E_{(x',y')\sim G'}[x'_i y'_j] = \frac{1}{N^2} \sum_{i,j \in [N]} \epsilon = \epsilon.$$

So now we know φ has the right expectation with respect to G'; but we need the expectation of φ relative to D to correctly capture the expected output of V_n. For this, we use the following remarkable lemma (whose proof is omitted).

Lemma 11.17. *For any multilinear function $F : \mathbb{R}^{2N} \to \mathbb{R}$,*

$$E_{z' \sim D}[F(z')] = E_{z \sim G'}[F(z)].$$

In words, if we process the input z' by a sufficiently restricted function F, namely multilinear F, then one cannot distinguish on expectation whether samples are drawn from D or G'.

Exercise 11.18. Combine Equation (11.3) and Lemma 11.17 to complete the proof of Theorem 11.11. What is the multilinear function F set to in our use of Lemma 11.17? □

11.4 Distinguishing D from U_{2N} is hard classically

We now wish to show that on expectation, any bounded depth circuit (as outlined in Theorem 11.1) cannot distinguish between samples z' drawn from D versus U_{2N}. A hint as to where we might wish to begin is given by Lemma 11.17, which recall says that on expectation under the action of any multilinear map F, D and U_{2N} are indistinguishable. Lucky for us, it is well-known that the action of any Boolean circuit is captured by a multilinear map, which we now review.

11.4.1 Boolean circuits and multilinear polynomials

Let C be an arbitrary circuit mapping $\{\pm 1\}^n$ to $\{\pm 1\}$. We can equivalently view C as a Boolean function from $\{\pm 1\}^n$ to $\{\pm 1\}$, to which the following lemma applies.

Lemma 11.19. *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a Boolean function. Then, f has a (unique) multilinear extension $g : \mathbb{R}^n \to \mathbb{R}$ such that $g(x) = f(x)$ when $x \in \{\pm 1\}^n$, and*

$$g(x) = \sum_{S \subseteq [n]} \widehat{g}(S) \cdot \prod_{i \in S} x_i,$$

for $\widehat{g}(S) \in \mathbb{R}$ the Fourier coefficients *of g.*

Proof sketch. The idea is to write g as a sum of 2^n terms, each of which encodes the product of an output $f(x)$ for some x times an "indicator function" which eliminates all terms other than x. The construction is best sketched via an example. Let $n = 3$, and consider input $x = (1, -1, 1)$. Then, we add to g the term

$$\left(\frac{1 + x_1}{2}\right)\left(\frac{1 - x_2}{2}\right)\left(\frac{1 + x_3}{2}\right) f(x).$$

The three terms in brackets serve as the "indicator function" alluded to above, which is uniquely specified by the string $x = (1, -1, 1)$.

Exercise 11.20. Verify that when $x = (1, -1, 1)$ is plugged into the equation above, the output is $f(x)$. What is the output when $x \in \{\pm 1\}^3$ but $x \neq (1, -1, 1)$?

Exercise 11.21. What would be the indicator function for, say, $x = (-1, -1, -1)$?

By adding to g a term of the above form for each possible input $x \in \{\pm 1\}^n$, expanding the brackets and simplifying the resulting expression, we obtain the final desired polynomial g, which may consist of exponentially many terms in general.

Exercise 11.22. Why is g multilinear? □

11.4.2 Tools and proof approach

In the remainder of our discussion, let us henceforth use C interchangeably to mean a classical circuit and its multilinear extension (Lemma 11.19), where the desired interpretation will hopefully be clear from context. Our goal is to show that if $C : \{\pm 1\}^{2N} \to \{\pm 1\}$ is constant

depth and has size $2^{\log^c N}$ for constant c, then C cannot distinguish on expectation between D and U_{2N}, i.e.

$$\left|E_{z' \sim D}[C(z')] - E_{u \sim U_{2N}}[C(u)]\right| \leq \frac{\text{polylog}(N)}{\sqrt{N}},$$

where recall $N = 2^n$ is exponentially large in the input parameter, n.

Basic proof approach. We begin with a simple observation.

Exercise 11.23. Prove that $E_{u \sim U_{2N}}[C(u)] = \widehat{C}(\emptyset)$, for $\widehat{C}(\emptyset)$ the Fourier coefficient of C corresponding to empty set $S = \emptyset$.

In words, the expectation of the circuit C under U_{2N} simply eliminates all higher order Fourier coefficients, leaving behind the constant term in the Fourier expansion of C, $\widehat{C}(\emptyset)$. Thus, it suffices for our goal to show

$$\left|E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset)\right| \leq \text{“small”}. \tag{11.4}$$

Our basic proof approach is hence to show that, on expectation, the contribution of higher order Fourier coefficients to $C(z')$ on inputs $z' \sim D$ is bounded.

Tools. To follow this basic approach, we require a pair of tools.

1. *Tail bounds on Fourier coefficients (proof omitted):*

   Lemma 11.24. *Let $C : \{\pm 1\}^{2N} \to \{\pm 1\}$ be a Boolean circuit of depth d and of size s. Then for any $k \in \mathbb{N}$, there exists $c \in O(1)$ such that*

   $$\sum_{S \subseteq [2N] \text{ s.t. } |S| = k} \left|\widehat{C}(S)\right| \leq (c \log s)^{(d-1)k}.$$

   In words, Lemma 11.24 intuitively says that for constant depth circuits, the low order (e.g. $k \in O(1)$) Fourier coefficients are bounded. Thus, this lemma alone gets us part of the way to Equation (11.4); the problem is the lemma does *not* work *a priori* for higher order coefficients.

2. *Random walks:* To use Lemma 11.24 to also bound the high-order Fourier coefficients of C, we use a trick — it turns out we can simulate drawing $z' \sim D$ with a random walk which takes “baby steps”. Applying Lemma 11.24 to each such “baby step” and applying the triangle inequality then does the trick.

11.4.3 Main theorem and proof sketch

We are now ready to state and sketch a proof of the main theorem, which is a quantitative version of Equation (11.4).

Theorem 11.25 (D fools bounded depth circuits)**.** *Let $C : \{\pm 1\}^{2N} \to \{\pm 1\}$ be a Boolean circuit of depth d and size s. Then*

$$\left|E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset)\right| \leq \frac{12\epsilon(c \log s)^{2(d-1)}}{\sqrt{N}}.$$

To prove Theorem 11.25, we require one final lemma, which is proven using the tail bounds of Lemma 11.24 (i.e. we will not directly use Lemma 11.24 otherwise in this lecture).

Lemma 11.26. *Fix $p \leq 1/4$. Let $C : \{\pm 1\}^{2N} \to \{\pm 1\}$ be a Boolean circuit of depth d and size s such that $\sqrt{\epsilon}p(c \log s)^{d-1} \leq \frac{1}{4}$. Then, for any $z_0 \in [-\frac{1}{2}, \frac{1}{2}]^{2N}$,*

$$|E_{z \sim G'}[C(z_0 + pz)] - C(z_0)| \leq \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}}.$$

Intuitively, Lemma 11.26 says that if in a random walk we evaluate C at our "current point" z_0, versus if we evaluate C on a slight perturbation of order p of z_0 (i.e. on $z_0 + pz$), then the ability for C to distinguish these inputs is damped quadratically in p. Thus, to simulate drawing $z' \sim D$, we shall run a random walk with baby steps of small order, i.e. $p = 1/\sqrt{N}$.

Proof sketch of Theorem 11.25. Assume without loss of generality that $\sqrt{\epsilon}(c \log s)^{d-1} \leq \frac{1}{4}N^{1/4}$, as otherwise the claim is vacuous. We run the following random walk for $t = N$ steps, with perturbation size $p = \frac{1}{\sqrt{N}}$:

1. Draw t samples $z^{(1)}, \ldots, z^{(t)} \sim G'$.
2. Set the output of "step i" of the random walk, for $i \in \{0, \ldots, t\}$, to be random variable

$$z^{\leq (i)} = p(z^{(1)} + \cdots + z^{(i)}).$$

One can show that $z^{\leq (t)} \sim G'$, i.e. the random walk exactly reproduces the distribution G'. (Namely, $z^{\leq (i)}$ and G' share the same expectation and covariance matrix.)

Now, for any step $i \in \{0, \ldots, t-1\}$, since each entry of $z^{\leq (i)}$ is Gaussian with variance scaling as $\sim p^2\epsilon$, we have with high probability that $z^{\leq (i)} \in [-1/2, 1/2]^{2N}$, as required for Lemma 11.26. Thus, by setting $z_0 = z^{\leq (i)}$ and calling Lemma 11.26 (i.e. this is where we use the tail bounds), we have that

$$\left| E[C(z^{\leq (i+1)})] - E[C(z^{\leq (i)})] \right| \leq \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}}. \tag{11.5}$$

Thus, the change in output of C, on expectation from step i to $i+1$ of the walk, is bounded by a factor scaling with $p^2 = 1/N$. Applying this recursively over all steps of the walk, we conclude:

$$\begin{aligned}
\left| E_{z' \sim D}[C(z')] - \widehat{C}(\emptyset) \right| &= \left| E_{z \sim G'}[C(z)] - \widehat{C}(\emptyset) \right| \\
&= \left| E[C(z^{\leq (t)})] - \widehat{C}(\emptyset) \right| \\
&\leq \sum_{i=0}^{t-1} \left| E[C(z^{\leq (i+1)})] - E[C(z^{\leq (i)})] \right| \\
&\leq t \frac{12\epsilon p^2 (c \log s)^{2(d-1)}}{\sqrt{N}} \\
&= \frac{12\epsilon (c \log s)^{2(d-1)}}{\sqrt{N}},
\end{aligned}$$

where the first statement follows from Lemma 11.17, the second since $z^{\leq (t)} \sim G'$, the third by the triangle inequality over all steps of the walk, and the fourth by Equation (11.5).

Exercise 11.27. The third statement above involves a term of form $\left|E[C(z^{\leq(1)})] - E[C(z^{\leq(0)})]\right|$, but we have not explicitly defined $z^{\leq(0)}$. Intuitively, we require $E[C(z^{\leq(0)})] = \widehat{C}(\emptyset)$. How should we define $z^{\leq(0)}$ for this requirement to hold? (Hint: Which input $x \in \mathbb{R}^{2N}$ reduces the multilinear extension C to its constant term, $\widehat{C}(\emptyset)$?) Why does this choice of $z^{\leq(0)}$ make sense given our definitions of $z^{\leq(i)}$ for $i \in [t]$? □

Bibliography