QUANTUM COMPUTING AND QUANTUM ERROR CORRECTION

Forest Kobayashi
Department of Mathematics, Harvey Mudd College, Claremont, CA 91711
April 25th, 2018

Abstract

Quantum computing has the potential to make problems that are classically "hard" for computers to solve (e.g., factoring products of large primes) computationally feasible [Sho95a]. However, in order to execute such quantum algorithms, quantum computers must be robust against *decoherence*, an undesirable phenomenon in which information in the quantum state is destroyed through interactions with the surroundings. Preventing decoherence completely is impossible, since any real-world quantum system cannot be perfectly isolated [Sch03]. However, there *are* methods for mitigating its effects, without performing measurements. These techniques are broadly referred to as Quantum Error Correction. In this paper, I'll give an overview of the theoretical basis of Quantum Computing, as well as an introduction to the motivations for developing error correcting codes, as well as some of the techniques that have been developed in the past two decades.

1. Introduction

In 1981, during a keynote speech delivered at the first conference on Physis and Computation, Richard Feynman observed that some quantum mechanical effects are impossible to simulate effectively on a classical computer [RP98]. As such, some researchers speculated that there was a loose sense in which quantum systems might have a higher information density than classical ones, and hence that a computer based on quantum systems might be able to perform computations more efficiently than a classical computer. In 1995, Peter Shor showed that indeed, on a quantum computer, integer factorization could be run in polynomial time (in particular, polynomial in $\log(N)$, where N is the size of the input), whereas the fastest classical algorithms run in sub-exponential time [Sho95a].

However, quantum computing presents many technological challenges. For reasons that will be explained later, in order for a quantum computer to function properly, the quantum system must be *coherent* — meaning there must exist definite phase relationships between the wavefunctions of the constituent particles. However, for any real-world coherent quantum system, over time, interactions with the surroundings will cause these definite phase relationships to degrade. This process is, as one might expect, known as *decoherence*.

Unfortunately, coherent quantum states are extremely fragile, with decoherence occuring on timescales of $\sim \hbar/k_B T$ [DNM09], [Unr94]. Hence, this provides weak upper limit for the time that a quantum calculation can take [Unr94]. Plugging in some numbers, we see that even for a temperature of 10µK, our computation time is limited to below 1µs. Hence, simply cooling the system until it is negligibly coupled to the surroundings would be impractical. So we must look elsewhere for a solution.

The idea is this: ok, preventing errors might be (for all intents and purposes) next to impossible. But what if instead of trying to prevent errors entirely, we just put in a good-faith effort to do so, and focus instead on correcting whatever errors we see at runtime? This presents challenges of its own: many of the error-correcting methods involved in classical computing rely on performing measurements of bits, which we cannot do in a quantum computer (measuring the state of the system would destroy the information we're using to compute).

But, it turns out that there *are* methods for measuring which errors have occurred (as well as correcting for them) that do not require direct measurement of the system's state! The study of these methods comprises the field of Quantum Error Correction, and is a relatively new field, with the seminal papers having been released in 1995 [Sho95b], [DNM09].

2. Mathematical Formulation

First, in order to be consistent with the literature (and hence, make my life easier), we will introduce *Dirac* notation, also known as *bra-ket* notation. But to appreciate the full power of this formalism, it is useful to first introduce some Math 70 material. If you have not had exposure to these concepts before, *don't panic!* I'll try to write the remainder of the paper such that it is optional. In particular, I will introduce *bra-ket* notation both from a formal mathematical perspective and from an intuitive physics perspective.

However, if you *have* had prior experience with inner products, operators, Hilbert space duals, etc., reading through the stuff below might give you a new perspective on the material we've covered this year, and could make the QEC material more intuitive. If you want to read just one part and skip the rest, I would glance briefly at the definition of operators (2.6), and theorems 2.1, 2.2, and 2.3. Again, I want to stress that there is basically *an entire course*'s worth of notation and definitions given below, so read on at your own risk.

Notational note: the symbol $\triangle$ will be used do denote the end of definitions, theorems, examples, etc.

2.1. The Linear Algebra

Inner Products, Hilbert Spaces, Operators, oh my! The linear algebraic underpinnings of Quantum Mechanics are quite rich, and to fully appreciate what the bra-ket notation means, I think it is important to take some time to establish its mathematical basis (pun intended).

I assume the reader is familiar with linear algebraic concepts such as a *basis*, but not much beyond that.

DEFINITION 2.1 (Inner Product). An inner product on a vector space V over a field $\mathbb{F}$ (basically, a set of scalars) is a map $\mathrm{Ip} : V \times V \to \mathbb{F}$ s.t. $\forall u, v, w \in V$, $\lambda \in \mathbb{F}$,

(a) $\mathrm{Ip}(v, v) \geq 0$
(b) $\mathrm{Ip}(v, v) = 0_{\mathbb{F}} \iff v = 0_V$
(c) $\mathrm{Ip}(u + v, w) = \mathrm{Ip}(u, w) + \mathrm{Ip}(v, w)$
(d) $\mathrm{Ip}(\lambda u, v) = \lambda \mathrm{Ip}(u, v)$
(e) $\mathrm{Ip}(u, v) = (\mathrm{Ip}(v, u))^\star$ (where $\star$ denotes the complex conjugate).

often, $\mathrm{Ip}(u, v)$ is denoted by $\langle u, v\rangle$, but to avoid confusion with bra-ket notation, we'll use the shorthand (u, v) instead. A vector space V together with a well-defined inner product is called an *inner product space.* △

Inner Products are useful, because they "act like" the dot product. In fact, any inner product can actually be viewed as applying a sufficient change of basis, then taking the dot product! So, why not just use the dot product then? Well, for "ordinary" vector spaces (e.g., $\mathbb{R}^2$), we can. But what if we have a vector space of polynomials? Do we take the dot product of the coordinate vectors? The answer isn't always clear. In cases such as those, we can use the inner product to fall back on Linear Algebra, a well-understood subject, to describe structure of things like Quantum Mechanics, which aren't necessarily as intuitive. The following two examples are from [Axl15].

EXAMPLE 2.1.1. Let $C[-1, 1]$ denote the set of continuous real-valued functions on the interval $[-1, 1]$. Then the following defines an inner product on $C[-1, 1]$:

$$(f, g) = \int_{-1}^{1} f(x)g(x)\ dx \qquad \triangle$$

EXAMPLE 2.1.2. Let $\mathbb{R}[x]$ be the set of polynomials in a variable x with coefficients in $\mathbb{R}$. Then the following defines an inner product on $\mathbb{R}[x]$:

$$(p, q) = \int_{0}^{\infty} p(x)q(x)e^{-x}\ dx. \qquad \triangle$$

We want to use the inner product to talk about common properties of vectors.

DEFINITION 2.2 (Norms!). Let V be an inner product space over a field $\mathbb{F}$. Then $\forall v \in V$, define the *norm* of v (denoted $\|v\|$), by

$$\|v\| = \sqrt{(v, v)}$$

i.e., the square root of the inner product of v with itself. If v has norm 1, it is said to be *normalized.* Note that $\|v\| = 0_{\mathbb{F}} \iff v = 0_V$, and $\forall \lambda \in \mathbb{F}$, $\|\lambda v\| = |\lambda|\|v\|$. △

DEFINITION 2.3 (Orthogonality). Let V be an inner product space over a field $\mathbb{F}$, and let $u, v \in V$. Then u, v are said to be *orthogonal* iff $(u, v) = 0_{\mathbb{F}}$. △

We now turn our attention to wavefunctions. We want to define some sort of inner product structure on $\mathcal{H}$, the space of wavefunctions for some quantum system. If there's any justice in the world, this should be tied to some sort of real, concrete physical idea. Indeed, this turns out to be the case. Because the wavefunctions we've been examining are all asymptotically 0, we can just take a slight modification of example 2.1.1, yielding the following:

DEFINITION 2.4 (Standard Inner Product on Wavefunctions). Let $\psi, \phi \in \mathcal{H}$, the set of wavefunctions in some n-dimensional space $\mathcal{V}$. Then define (ϕ, ψ) by

$$(\phi, \psi) = \int_{\mathcal{V}} \psi^\star \phi\ d\mathcal{V}.$$

in the case that $\mathcal{V}$ is one-dimensional, this reduces to the familiar quantity

$$(\phi, \psi) = \int_{-\infty}^{\infty} (\psi(x))^\star \phi(x)\ dx$$

I.e., the probability that performing a measurement on a state ϕ will yield a state ψ. △

We now introduce the idea of a Hilbert space. The motivation is as follows: say we want to talk about the angle between two vectors. Well, we could just take their dot product, and divide by the magnitudes, and apply a nice arccos to both sides. But what if our vector space has an infinite basis? How do we know that the dot product will actually be finite? To answer these sorts of questions, we introduce the abstraction of a Hilbert Space.

DEFINITION 2.5 (Hilbert Space). Let $\mathcal{H}$ be an inner product space over a field $\mathbb{F}$. Let $\{u_k\} \subseteq \mathcal{H}$ be an arbitrary sequence of vectors in $\mathcal{H}$. If

$$\sum_{k=0}^{\infty} \|u_k\| \leq \infty$$

implies that $\exists u \in \mathcal{H}$ s.t.

$$\left(\sum_{k=0}^{\infty} u_k\right) \to u$$

(i.e., that if the norms converge, the series converges), then we call $\mathcal{H}$ a *Hilbert Space.* For those of you who have taken a course in Analysis, note that a Hilbert space is really an inner product space where the metric induced by the norm ($d(u, v) := \|u - v\|$) makes $\mathcal{H}$ a *complete* metric space. △

Note that a Hilbert Space is a special sort of inner product space, and hence, anything we define or prove about inner product spaces extends to Hilbert Spaces as well.

DEFINITION 2.6 (Operators). Let V be an inner product space over a field $\mathbb{F}$. An *operator* on V is a linear transformation from V to itself. That is, an operator is a map $T : V \to V$ s.t. $\forall u, v \in V$, $\lambda \in \mathbb{F}$

(a) $T(u + v) = T(u) + T(v)$
(b) $T(\lambda v) = \lambda T(v)$.

operators can be represented by matrices (although, the matrix representation of an operator is not the same thing as an operator itself, as matrix representations encode an operator with respect to some particular basis). The set of operators on V themselves form a vector space, denoted by $\mathcal{L}(V)$. Operator addition is defined as one would expect, with $(S+T)(u) = S(u)+T(u)$. Multiplication is defined by function composition: $ST(u) = S(T(u))$. $\triangle$

Basically, an operator just takes elements of an inner product space, and shuffles them around in a particular way that respects the algebraic structure of the vector space. For those of you who have taken Math 171, you might recognize this as a particular sort of homomorphism.

DEFINITION 2.7 (Eigenvectors of Operators). Let V be an inner product space over a field $\mathbb{F}$, and let $T \in \mathcal{L}(V)$. Let $v \in V$. Then v is said to be an *eigenvector* of T iff $\exists \lambda \in \mathbb{F}$ s.t. $Tv = \lambda v$. $\triangle$

We now define *adjoints.* Admittedly, the definition is a bit abstract, and you might wonder whether adjoints for most operators even exist. Fret not, in the restricted context we'll be examining, when we represent operators by matrices, adjoints are just conjugate transposes.

DEFINITION 2.8 (Adjoint of an Operator). Let V be an inner product space over a field $\mathbb{F}$, and let $T \in \mathcal{L}(V)$. Then the *adjoint* of T, denoted $T^\dagger$, is also an operator on V, with the property that $\forall u, v \in V$,

$$(T(u), v) = (u, T^\dagger(v)). \qquad \triangle$$

for the reader's entertainment, we list some fun properties of the adjoint given in [Axl15]. Let V be an inner product space, and let $S, T \in \mathcal{L}(V)$. Then

(a) $(S + T)^\dagger = S^\dagger + T^\dagger$
(b) $\forall \lambda \in \mathbb{F}$, $(\lambda T)^\dagger = \lambda^\star T^\dagger$.
(c) $(T^\dagger)^\dagger = T$
(d) $I^\dagger = I$ (where I is the identity operator).
(e) $(ST)^\dagger = T^\dagger S^\dagger$
(f) $\ker(T^\dagger) = (\text{Image}(T))^\perp$ (where $\perp$ denotes an orthogonal complement).

Finally, we can talk about why this is relevant. In quantum mechanics, it turns out that certain families of operators are intimately tied to observables. In particular, these are *Hermetian*, or *self-adjoint* operators. These are defined as follows:

DEFINITION 2.9. An operator $T \in \mathcal{L}(V)$ is said to be *self-adjoint* or *hermetian* iff $T = T^\dagger$. That is, $\forall u, v \in V$, we have

$$(T(u), v) = (u, T(v)). \qquad \triangle$$

Hermetian operators are important for the following reasons [Axl15]:

THEOREM 2.1. *Let V be an inner product space, and let $T \in \mathcal{L}(V)$. Then if $T = T^\dagger$ (T is Hermetian), then every eigenvalue of T is real.* $\triangle$

We offer a short proof. Recall 2.1 (e) — for any inner product, $(u, v) = (v, u)^\star$ (where $\star$ denotes the complex conjugate).

Proof. Let $v \in V$ be a non-zero eigenvector of T, and suppose T is Hermetian. Then

$$\begin{aligned}
(Tv, v) &= (v, T^\dagger v) \\
(Tv, v) &= (v, Tv) \\
(\lambda v, v) &= (v, \lambda v) \\
(\lambda v, v) &= (\lambda v, v)^\star \\
\lambda(v, v) &= \lambda^\star (v, v)^\star \\
\lambda(v, v) &= \lambda^\star (v, v) \\
\lambda\|v\| &= \lambda^\star \|v\| \\
\lambda &= \lambda^\star
\end{aligned}$$

hence $\lambda \in \mathbb{R}$. ∎

This next theorem tells us that for Hermetian operators, measurement will obtain a real value.

THEOREM 2.2. *Suppose V is an complex inner product space, and $T \in \mathcal{L}(V)$. Then T is Hermetian iff $\forall v \in V$,*

$$(Tv, v) \in \mathbb{R}. \qquad \triangle$$

I'll perform a cardinal sin of mathematics, and not offer a proof (both for this theorem and the next one), because they are not terribly enlightening.

THEOREM 2.3. *Let $\mathcal{H}$ be a Hilbert space, and let $U \in \mathcal{L}$. Call an operator U* unitary *iff $UU^\dagger = I = U^\dagger U$, that is, if $U^{-1} = U^\dagger$. Then if U is unitary, $\forall u, v \in \mathcal{H}$, we have*

$$(U(u), U(v)) = (u, v)$$

$\triangle$

Because we define the (standard) norm in terms of the inner product, this has the consequence that unitary operators conserve probability. Further, since they are invertible operators, they define a bijection over $\mathcal{H}$.

We're almost to Dirac notation, I promise. We just have to introduce one more mathematical object: linear functionals.

DEFINITION 2.10. Let V be an inner product space over a field $\mathbb{F}$. Then a *linear functional* is a map $\varphi : V \to \mathbb{F}$ such that $\forall u, v \in V, \lambda \in \mathbb{F}$,

(a) $\varphi(u + v) = \varphi(u) + \varphi(v)$, and
(b) $\varphi(\lambda v) = \lambda\varphi(v)$. △

DEFINITION 2.11 (Dual Space). Let V be an inner product space over a field $\mathbb{F}$. Then the *algebraic dual space*, denoted V^*, is defined as the set of all linear functionals on V:

$$V^\star = \{\varphi : V \to \mathbb{F} \mid \varphi \text{ is linear.}\}$$

if we define addition and scalar multiplication in V^* by $\forall \varphi, \theta \in V^*$, $v \in V$, $\lambda \in \mathbb{F}$,

(a) $(\varphi + \theta)(v) = \varphi(v) + \theta(v)$
(b) $(\lambda\varphi)(v) = \lambda(\varphi(v))$

then V^* is itself a vector space, and is referred to as the *dual space* of V. △

This is primarily useful because of the following theorem:

THEOREM 2.4 (Riesz Representation Theorem). *Let $\mathcal{H}$ be a Hilbert space, and let H^* be its dual space. Let $\mathcal{H}^*$ be the subspace of H^* corresponding to* continuous *linear functionals. Then we have the following: $\forall u, v \in \mathcal{H}$, the functional*

$$\varphi_v(u) = (u, v)$$

is an element of $\mathcal{H}^*$. Furthermore, φ_v is unique. *For those interested in category theory, we can say $\mathcal{H}$ is anti-isomorphic to $\mathcal{H}^*$.* △

Proof. Terry Tao once told me this in a personal email, therefore it is true.[1] ■

2.2. THE DIRAC FORMALISM

Establishing this linear algebraic framework is quite useful, although a bit overwhelming at first. As it turns out, the set of wavefunctions for some particular quantum mechanical system form a Hilbert space $\mathcal{H}$ over $\mathbb{C}$. The probability of a wavefunction ϕ collapsing to a state ψ upon measurement is given by the inner product (ϕ, ψ). Observables such as the hamiltonian, position, momentum, or spin of a system correspond to linear operators on $\mathcal{H}$. Time evolution as well can be formalized as a unitary *time-evolution* operator acting on $\mathcal{H}$. And because we don't need to write down a specific basis to consider the effect of an operator A, it turns out we can do lots of quantum mechanics without ever touching a basis. Now, we put all this machinery to work in defining Dirac notation.

2.2.1. BRAS AND KETS: MATH VERSION

Let $\psi, \phi \in \mathcal{H}$. Then taking

$$(\phi, \psi) = \int_{\mathcal{V}} \psi^\star \phi \; d\mathcal{V}$$

gives us the probability of finding a state ϕ in the state ψ upon performing a measurement. But, as we saw in the introduction of a dual space, there is a sense in which the two sides of the inner product are meaningful alone. We introduce an innocuous change in notation. What if, instead of denoting the inner product by (ϕ, ψ), we denoted it by $\langle \psi \mid \phi \rangle$? So far so good, nothing really seems to be amiss yet, aside from the fact that we've flipped the order for some reason.[2] But what if we split it on the vertical line, and made these quantities separate?

$$(\phi, \psi) = \langle \psi \mid \phi \rangle = \langle \psi || \phi \rangle.$$

Is there a way in which this makes meaningful mathematical sense? It turns out there is.

In order for this statement to hold, we apply the Riesz representation theorem. By $\langle \psi |$, we really mean the linear functional $\varphi_\psi \in \mathcal{H}^*$ where $\varphi_\psi : \mathcal{H} \to \mathbb{C}$, and by $\langle \psi || \phi \rangle$, we mean $\varphi_\psi(\phi)$.[3] This is perfectly consistent with the inner product, and allows us to separate $\langle \psi \mid \phi \rangle$ above without deep sensations of guilt. The $\langle \psi |$ is called "the bra ψ", and the $| \phi \rangle$ is called "the ket ϕ". Together, they form a bra-ket, or bra(c)ket.

Essentially, kets like $|\phi\rangle$ give us a representation of some quantum state, while bras like $\langle \psi |$ can combine with kets to give us the probability amplitude of finding a particle in the state $|\phi\rangle$ to be in the state $|\psi\rangle$ upon performing a measurement [Tow00].

Ok, but the entire point of separating bras and kets was so that we could use them in contexts that *aren't* just forming bra(c)kets. Otherwise, we could have just stuck with the inner product. We have to ask ourselves, now that we have these free-floating bras and kets, what happens when we smash them together?

This, as it turns out, is fundamental to understanding the nature of entangled states. But to understand the details of this process, we need to introduce a little more math. First, we'll define the *outer product* as follows:

[1] This is a lie
[2] This is because of the different inner product notation used in math vs in physics.
[3] Note that in the case of $\langle \psi |$, because of the Riesz representation theorem, we can easily convert back to a vector representation for ψ if that is more convenient to work with.

DEFINITION 2.12 (Outer (Tensor) product of vectors). Let V be a finite-dimensional vector space, and let $u, v \in V$. Define the *outer product* of u and v, denoted $u \otimes v$, as the matrix M st

$$u \otimes v = M = u(v^T)^\star.$$

That is,

$$M_{i,j} = u_i(v_j^\star). \qquad \triangle$$

Note that the outer product also referred to by the *tensor product*, as we will see shortly. This is because the outer product is a special case of the tensor product, namely that where u, v are vectors.

EXAMPLE 2.12.1. Let

$$u = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Then

$$\begin{aligned} u \otimes v &= \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} v_0^\star & v_1^\star & v_2^\star & v_3^\star \end{bmatrix} \\ &= \begin{bmatrix} u_0v_0^\star & u_0v_1^\star & u_0v_2^\star & u_0v_3^\star \\ u_1v_0^\star & u_1v_1^\star & u_1v_2^\star & u_1v_3^\star \\ u_2v_0^\star & u_2v_1^\star & u_2v_2^\star & u_2v_3^\star \end{bmatrix} \end{aligned}$$

$\triangle$

Whereas the inner product takes two vectors (of equal length) and collapses them down to a scalar, the outer product combines two vectors (*not* necessarily of equal length) by inflating them into a matrix. Further, note that the two are related: the standard Euclidean inner product is simply the *trace* of the outer product.[4]

But the similarity runs slightly deeper than that. Observe that if the vectors are of equal length, the standard Euclidean inner product, (u, v), is simply $(u^T)^\star v$, whereas the outer product is $u(v^T)^\star$. From this, one might deduce that since $\langle\psi||\phi\rangle$ is the inner product (ϕ, ψ), that

$$|\phi\rangle\langle\psi| = \phi \otimes \psi.$$

i.e., reversing the order yields the outer product. Indeed, this is the case.[5] Note that in the special case where $\psi = \phi$ (i.e., in $|\phi\rangle\langle\phi|$), we call this a *projection operator*. To see why, consider the following: let $A = |\phi\rangle\langle\phi|$. Suppose we have some other state, $|\psi\rangle$. Then

$$\begin{aligned} A|\psi\rangle &= |\phi\rangle\langle\phi||\psi\rangle \\ &= \text{proj}_{|\phi\rangle}(|\psi\rangle) \end{aligned}$$

where the leap from the first line to the second is definitional.

The definition of the outer product in the infinite-dimensional case is modestly similar to the finite-dimensional one:

DEFINITION 2.13 (Outer product in Hilbert Spaces). Let $\mathcal{H}$ be a Hilbert space, and $\mathcal{H}^*$ its dual. Let $u, v \in H$, and let $\varphi_u \in \mathcal{H}^*$ be the element of $\mathcal{H}^*$ corresponding to u under the Riesz representation theorem. Then the tensor product $u \otimes v$ corresponds to an operator $M \in \mathcal{L}(V)$ st $\forall w \in V$,

$$M(w) = \varphi_u(w)v \qquad \triangle$$

DEFINITION 2.14 (Tensor Product of Hilbert Spaces). Let V, W be Hilbert spaces on a field $\mathbb{F}$, with bases $\mathcal{V} = \{e_1, e_2, \ldots, e_k\}$, and $\mathcal{W} = \{f_1, f_2, \ldots, f_\ell\}$. Then define the *tensor product* of V and W, denoted $V \otimes_{\mathbb{F}} W$ or $V \otimes W$, to be the space generated by the basis $\{v_i \otimes w_j \mid v_i \in \mathcal{V}, w_j \in \mathcal{W}\}$. $\triangle$

EXAMPLE 2.14.1. Here are a few examples of tensor products of $\mathbb{R}$:

(a) $\mathbb{R} \otimes \mathbb{R} \cong \mathbb{R}^2$.
(b) $\mathbb{R}^3 \otimes \mathbb{R}^3 \cong \mathbb{R}^9$.
(c) In general, $R^m \otimes R^n \cong R^{mn}$, and for arbitrary spaces U and V, $\dim(U \otimes V) = \dim(U) \times \dim(V)$ $\triangle$

Finally, we can get to the good stuff. Let's say we have two different quantum systems, with states corresponding to elements of the Hilbert spaces $\mathcal{H}_1$, $\mathcal{H}_2$. Suppose, for simplicity, that the systems are non-interacting. Then Hilbert space representing the combined system is

$$\mathcal{H}_1 \otimes \mathcal{H}_2.$$

If $|\psi\rangle \in \mathcal{H}_1$ represents the state of system 1, and $|\phi\rangle \in \mathcal{H}_2$ represents the state of system 2, then we represent the combined state by $|\psi\rangle_{\mathcal{H}_1} \otimes |\phi\rangle_{\mathcal{H}_2}$, or (more frequently) $|\psi\phi\rangle$. Although we introduced $\otimes$ in the context of outer products and matrix multiplications, matrices themselves form a vector space, so we can think of $|\psi\phi\rangle$ as a gigantic vector in an mn-dimensional space (unless $\mathcal{H}_1, \mathcal{H}_2$ are infinite, but the analogy still sorta works).

One last note before moving on: it turns out that not every state of $\mathcal{H}_1 \times \mathcal{H}_2$ is necessarily of the form $|\psi\rangle_{\mathcal{H}_1} \otimes |\phi\rangle_{\mathcal{H}_2}$! Let $\{e_i\}$ be a basis for $\mathcal{H}_1$, and $\{f_j\}$ be a basis for $\mathcal{H}_2$. Then the most general possible state of

[4]Technically, the outer product can be defined easily where u, v are not from spaces of the same dimension, but those aren't relevant for our purposes here.

[5]Fun fact: just as the Euclidean inner product can be generalized to an arbitrary inner product, so too can the outer product be generalized to the *kronecker product*.

$\mathcal{H}_1 \otimes \mathcal{H}_2$ is of the form

$$|\theta\rangle = \sum_{i,j} c_{i,j} |e_i\rangle_{\mathcal{H}_1} \otimes |f_j\rangle_{\mathcal{H}_2}$$

basically, there are situations in which we have valid states in the tensor product space that cannot be written as the tensor product of states from the constituent spaces. As it turns out, such "inseperable" states correspond to *entangled* states.

OK, enough math for now. Let's move on to something more tangible.

3. Basics of Quantum Information

First, we introduce kets outside of a math-heavy setting (because unless you really care about mathematical intuition and find it essential to your understanding of a topic, the math isn't super relevant), and then jump straight into some basics of Quantum Information, and finally to QEC.

3.1. (Bra[c])Kets: Non-Math Version

Kets are a component of Dirac notation, also referred to as *bra-ket* notation. Bras look like $\langle\psi|$, and kets look like $|\phi\rangle$. The portion inside is just a label to keep track of which bra is which, and similarly with kets. Kets are much more important to the material we'll be discussing today, so we'll highlight them first, and speak only briefly about bras.

Despite the new foreign notation, we've actually been using kets the entire semester. Let's say we have some arbitrary wavefunction Ψ representing a quantum system. Suppose that this system has an orthonormal basis $\psi_0, \psi_1, \psi_2, \ldots$. Then as we learned, we can express Ψ by

$$\Psi = \sum_n c_n \psi_n.$$

$|\Psi\rangle$ means almost exactly the same thing, just a little more succinctly. Recall that for many of the systems we examined, we referred to a collection of $\psi_0, \psi_1, \ldots, \psi_n$ as an *orthonormal basis*. This is not a coincidence. Indeed, as detailed in the math section above, these "basis" wavefunctions really are the basis of an honest-to-goodness vector space.[6] Ket's make that a little more tangible. By

$$|\Psi\rangle$$

we really mean the column vector

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

hence, $|\Psi\rangle$ is a *coordinate vector*, which tells us weights for each of the $\psi_0, \ldots, \psi_n$. Note that the magnitude of this vector is

$$\| |\Psi\rangle \| = \sum_n c_n^2$$

which, if Ψ has been *normalized*, is equal to 1. Hence, "normalized" in the context we've been using it means exactly the same thing as it did for the vectors we encountered in Math 65!

Since the $\psi_0, \psi_1, \ldots, \psi_n$ form a basis, we can then write $|\Psi\rangle$ as a linear combination of basis vectors:

$$|\Psi\rangle = c_0|\psi_0\rangle + c_1|\psi_1\rangle + \cdots + c_n|\psi_n\rangle$$

Recall that the labels inside the kets are arbitrary. Then, since "ψ sub whatever" is annoying to write, we might express this same quantity more succinctly by simply labeling the basis states with $0, 1, \ldots, n$ instead of $\psi_0, \psi_1, \ldots, \psi_n$:

$$|\Psi\rangle = c_0|0\rangle + c_1|1\rangle + \cdots + c_n|n\rangle.$$

Finally, all that's really left to introduce is what is meant by something like $|\psi\phi\rangle$ (alternatively denoted by $|\psi\rangle|\phi\rangle$, or $|\psi\rangle \otimes |\phi\rangle$). Unfortunately, this does *not* just represent the entangled state of ψ with ϕ, or something nice like the dot product of $|\psi\rangle$ and $|\phi\rangle$. Instead, it describes the *composite* state of a two systems that have been combined together.

Let's say we have some quantum system A whose behavior is described by the basis states $|\psi_0\rangle, |\psi_1\rangle, \ldots |\psi_m\rangle$. Suppose A is in a state $|\psi\rangle$, where $|\psi\rangle$ is a linear combination of the m basis states. Suppose now that we also have some system B that is not interacting with A, and that it has some basis $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_n\rangle$. Now let's suppose we bring these two systems together, and combine them into one, such that the states of A and B themselves do not change. Then, we have a new system AB (also denoted $A \otimes B$) of dimension mn, and we describe the state of the system by $|\psi\phi\rangle$. So really, this is just a notation for talking about the states of multiple things at once.

Now, a brief word on bras! We define the bra $\langle\psi|$ to be the *conjugate transpose* of $|\psi\rangle$. This gives bras the following properties:

(a) $\langle\psi||\psi\rangle = \|\psi\|$

(b) $\langle\psi||\phi\rangle$ gives the *probability amplitude* for finding a state $|\phi\rangle$ in the state $|\psi\rangle$ upon performing a measurement [Tow00].

(c) $|\phi\rangle\langle\phi|$ gives a *projection operator* onto the state $|\phi\rangle$. That is, if we let $A = |\phi\rangle\langle\phi|$, and consider some arbitrary $|\psi\rangle$, then $A|\psi\rangle = \mathrm{proj}_{|\phi\rangle}(|\psi\rangle)$. Here, it is invaluable to fall back on linear algebra intuition, and remember that kets are really vectors.

(d) Finally, if we have some operator P, then $\langle\psi|P|\psi\rangle$ is the expected value for P on the state $|\psi\rangle$. That is, $\langle\psi|P|\psi\rangle = \langle P\rangle_\psi$.

[6]Well, technically, a Hilbert space. But... meh. Let's live a little.

Now, we have everything we need to introduce quantum information and QEC. Onwards!

3.2. The Qubit

In classical computing, the fundamental unit of information is represented in the form of *bits*, which can attain binary values of 0 and 1. These are typically represented by two allowed states of a circuit, which are distinguished by a large potential difference. We want to find an analogous structure in quantum mechanics.

A natural choice would be the spin-state of a spin-$\frac{1}{2}$ particle [Bar15]. So, suppose we have some spin-$\frac{1}{2}$ particle, and let us refer to the two basis spin states by $|0\rangle$ and $|1\rangle$. Then the total spin-state of the system will be a superposition of the two basis states, given by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where (as usual) α, $\beta \in \mathbb{C}$. Really we could have chosen any two orthonormal basis states with which to represent our qubit, but this choice is convenient for our intuition, as it allows us to employ the Pauli spin matrices to describe the behavior of the system [Bar15]. Note that if we perform a measurement on the system, then just as in a classical computer, we can have *only one* of $|0\rangle$, $|1\rangle$. Hence, qubits act "like" bits when we perform a measurement; loosely speaking, it is only while they are in a superposition of states that we need to treat them differently. Until that point though, $|\psi\rangle$ can be any of a continuous range of linear combinations of the two basis states.

You might be thinking, *What's the big deal? If these just act like regular bits, how do we get any computational speedups? I was promised an end to RSA, fast solving of dihedral hidden subgroups, and some nonsense about discrete logarithms. I want my (grant) money back!* The key is that they only act like regular bits *after a measurement has been performed*, while prior to taking a measurement, they are a superposition of them.

Here's an example. In a classical computer, whether we take a measurement or not, a bit is only really in one "state." Hence, using n bits, we can represent 2^n states — but only one at a time, e.g.

$$\underbrace{01111001\ldots01000000}_{n \text{ bits}}$$

By contrast, a quantum computer with n qubits can be in a superposition of all 2^n states *simultaneously!*

$$\begin{aligned}|\phi\rangle = \quad & \alpha_0 \overbrace{\big|00000000\ldots00000000\big\rangle}^{n \text{ qubits}} \\ + \quad & \alpha_1 \big|00000000\ldots00000001\big\rangle \\ \vdots \\ + \quad & \alpha_k \big|01111001\ldots01000000\big\rangle \\ \vdots \\ + \quad & \alpha_{2^n} \big|11111111\ldots11111111\big\rangle\end{aligned}$$

Similarly to the case of a single qubit, measurement results for any of the states on the right hand side will occur with probability $|\alpha_i|^2$.

We will not go into detail here, but essentially, this is the property that allows quantum algorithms to perform more efficiently than classical ones, particularly in situations with massive parallelism. Because exponentials are incredibly powerful. If we could cobble together ≈ 300 qubits, then the number of states in our quantum computer would be about $50,000\times$ larger than the estimated number of particles in the known universe. As Carlton Caves once remarked, "Hilbert space is a big place" [NC11].

3.3. Circuitry in Quantum Computing

Now, let's begin examing how we might compute things by manipulating kets. In order to perform computations, we must have some notion of logical operations. Essentially, we want to develop the quantum analog of a circuit. We start with circuit elements. For simplicity's sake, we'll just deal for now with a single-qubit system.

In the realm of classical computing, we have all sorts of logical operations, whose behavior are defined by a "truth table." Let's start simple, with the NOT operator. Clasically, this is defined as follows:

$$0 \xrightarrow{\text{NOT}} 1$$
$$1 \xrightarrow{\text{NOT}} 0$$

Can we define an analogous NOT operator in our quantum computer? That is, can we find some operator X that performs the following map for an arbitrary input?

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{X} \alpha|1\rangle + \beta|0\rangle$$

The answer is yes, and it turns out the desired X has the following form:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

which one might recognize as σ_x, the Pauli spin matrix for projection onto the x axis. X is often referred to as the "bit flip operator", or the "Paui sigma x operator."

In the case of X, things worked out nicely. But before we go on to throw in fistfulls of other operators (also referred to as "gates"), we need to pause and think about what we're doing. What sorts of restrictions must we place on them to make sure they make physical sense? Remarkably, it turns out we only need one. If we let U be the matrix representation of some quantum gate, we only require that U is *unitary*. This is defined as follows ([NC11]):

Definition 3.1. Let U be a matrix representation of an operator, and let $U^\dagger$ denote its adjoint (for our purposes today, "adjoint" means the conjugate transpose).

Then U is said to be *unitary* iff

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

that is, $U^\dagger = U^{-1}$. △

Why does it make sense that we require our operators to be unitary? Because, (as discussed in the math section), unitary matrices have the handy property of preserving the *magnitude* of a vector, which (when applied to kets) means that probability for an arbitrary input state is conserved!

There are many other important gates fundamental to quantum circuitry, but we define just the two other most important ones below. First, we give the truth table for each, then show their matrix representations. We consider the Z gate, which applies a phase flip to $|1\rangle$:

$$|0\rangle \xrightarrow{Z} |0\rangle$$
$$|1\rangle \xrightarrow{Z} -|1\rangle$$

this has matrix representation

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

which is σ_z, the Pauli sigma z operator.[7] The third important quantum gate is known as the *Hadamard* gate H, which maps the basis kets to equal superpositions, with a possible phase-flip:

$$|0\rangle \overset{H}{\longmapsto} \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$|1\rangle \overset{H}{\longmapsto} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

It has matrix representation

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

All of the operators above can be shown to be unitary, and hence probability-conserving.[8] In quantum circuits, we denote these operators by boxes containing their symbol along some "wire." To see an example, let's examine the transformation each of the operators we've introduced performs on an arbitrary input state:

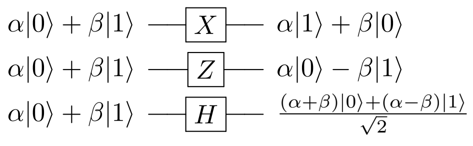
$\alpha|0\rangle + \beta|1\rangle$
X
$\alpha|1\rangle + \beta|0\rangle$
$\alpha|0\rangle + \beta|1\rangle$
Z
$\alpha|0\rangle - \beta|1\rangle$
$\alpha|0\rangle + \beta|1\rangle$
H
$\frac{(\alpha+\beta)|0\rangle + (\alpha-\beta)|1\rangle}{\sqrt{2}}$

Figure 1: Fundamental Quantum Gates

We now shift focus from single-qubit systems to to multi-qubit systems. In classical computing, it is known that any function performed on bits can be computed by composing NAND gates in various patterns [NC11]. NAND is hence said to be *universal*, and has the following truth table (where $\overline{a \wedge b}$ denotes "a NAND b")

Table 1: The NAND gate truth table

a	b	$\overline{a \wedge b}$
0	0	1
0	1	1
1	0	1
1	1	0

While we do not have a *single* universal multi-qubit quantum gate, we do have one that comes close. It is called the controlled-NOT gate, or CNOT. This is the quantum analog of the XOR operator. It's so important that I'll give it its own fancy definition thing:

DEFINITION 3.2 (CNOT). Let CNOT denote the *controlled-NOT* operator on two qubits. It has the following properties:

$$|00\rangle \xrightarrow{CNOT} |00\rangle$$
$$|01\rangle \xrightarrow{CNOT} |01\rangle$$
$$|10\rangle \xrightarrow{CNOT} |11\rangle$$
$$|11\rangle \xrightarrow{CNOT} |10\rangle$$
△

Note that here, the first qubit in our ket *controls* whether or not the second qubit gets inverted. We can represent this by the following equivalent circuit diagram:

DEFINITION 3.3 (CNOT, redux). Let $\oplus$ denote addition modulo 2. That is, $a \oplus b = 0$ if $a + b$ is even, and $a \oplus b = 1$ if $a + b$ is odd. Then the *controlled-NOT* circuit performs the following operation:

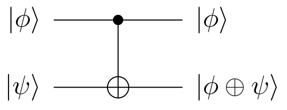

Figure 2: CNOT circuit

△

Here, the black dot represents the junction where $|\phi\rangle$ might act to flip $|\psi\rangle$, and $|\phi\rangle$ is called the *control* qubit. Meanwhile, the $\oplus$ on the diagram denotes where $|\psi\rangle$ may or may not be bit flipped (depending on the value

[7]But what about σ_y?? Well, it turns out to be an allowed quantum gate as well, but it will not be important for our discussion today.
[8]Note that there is actually a fourth "essential" operator we haven't mentioned (because it is boring), namely the identity operator I.

of $|\phi\rangle$), and $|\psi\rangle$ is referred to as the *target* qubit. CNOT can be represented by the matrix

$$U_{CN}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$$

over the space $\{|0\rangle,|1\rangle\}\otimes\{|0\rangle,|1\rangle\}$. This can be quickly verified to be unitary. What is exciting is that *any* computation on qubits can be performed by an architecture composed solely of CNOT, together with X, Z, and H.

3.4. The Effects of Decoherence

In the introduction, I promised that decoherence would turn out to be important in quantum computing. The following example, graciously provided to me by Prof. Lynn, illustrates why. Suppose we have the following quantum circuit:

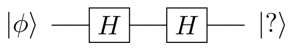

Figure 3: Double Hadamard

This performs the following maps:

$$|0\rangle \longrightarrow \frac{|0\rangle+|1\rangle}{\sqrt{2}} \longrightarrow |0\rangle$$
$$|1\rangle \longrightarrow \frac{|0\rangle-|1\rangle}{\sqrt{2}} \longrightarrow |1\rangle$$

and as we can see, this should should perform an identity transformation (alternatively, note that $H^2 = I$).

Recall that decoherence occurs when we have phase uncertainty between two states. In the example above, this could be catastrophic. Suppose that we input $|0\rangle$ into our circuit.[9] It passes through the first Hadamard gate, and the expected transformation is performed. But suppose now that our state decoheres to the point where we have a spontaneous phase flip:

$$\frac{|0\rangle+|1\rangle}{\sqrt{2}} \xrightarrow{\text{!!! oh no !!!}} \frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

then, upon passing through the second hadamard gate, it will output $|1\rangle$! Suddenly, we have $I|0\rangle = |1\rangle$, and similarly, we could get $I|1\rangle = |0\rangle$. This is terrible news! Our identity operator has become twisted up with the NOT operator X! And if it were the *first* qubit that were flipped, I (the identity, not me) would have the effect of ZX! Surely this would destroy any meaningful computations. We need to find a way of mitigating these effects.

4. Simple Error Correction

As we saw above, quantum computers, like classical computers, must be robust against random errors for computations to yield meaningful results. The landscapes here, though, are much different than those in classical computing. Whereas in the CPU of a typical modern desktop computer, errors occur only once in about 10^{17} computations (an stupendously low error rate), we saw in the introduction that quantum computers will experience decoherence errors on the order of microseconds, even when cooled substantially [NC11], [Unr94]. Clearly, we need highly capable error-correcting methods if we are to have any shot at computing anything.

4.1. Classical Error Correction

We examine classical error correction methods for inspiration. Although the error rate in an isolated, classical CPU is incredibly low (to the point where error correction might not be a concern), there are many technologies that require classical error correction techniques for even basic functionality. The CPU is isolated quite well from the outside world, whereas other digital systems (e.g., routers transmitting data from the internet) are not. Hence, there exists a robust system of classical error-correction procedures that we would like to examine. One of the simplest (and admittedly, crudest) methods is to use a *replication code.* As an example, suppose we had the following binary bit string:

$$101010.$$

we want to make this robust against random bitflip errors (i.e., where a bit changes from a 0 to a 1). To do so, we might perform the following procedure:

(a) Take our original string, 101010, and copy every bit 3 times, yielding the string

$$111000111000111000.$$

(b) Say a random bit flips, yielding the following string:

$$111001111000111000.$$

(c) If we're sliding along the bit string in with a window of length 3, on our second segment, we'll notice that not all the bits have the same value: 001.

(d) We want to know how likely it is that the final 1 should really be a 0. To calculate this, suppose that p is the probability of a bit flip. Then the probability that two or more of the bits are flipped is $3p^2(1-p)+p^3$ [NC11]. Thus, if we flip the third bit, the probability that we have properly corrected our code is $3p^2 - 2p^3$.

(e) Whenver $p < 1/2$, this correction code will make our transmission more reliable.

[9]Note that although we would never be inputting a known, definite $|0\rangle$ into our quantum computer, performing the analysis in this way allows us to exploit linearity of quantum gates to show that the ensuing error would have the same effect on our superposition.

However, we run into a problem when attempting to apply this to our quantum computers: inherently, by checking each of the bits to see if they match the surrounding bits, we are performing a measurement. Here, I gave the particular example of a repitition code, but it turns out that in general, almost all classical error-correcting schemes rely on measurement. But in a quantum computer, performing a measurement before the final stage of our calculation will destroy the superposition, and reduce our quantum computer to a hyper-expensive, low-powered classical computer stuffed in a super-cold fridge. And nobody wants that.

But wait!, you might be thinking. *I see a loophole. Why don't we just copy the qubits before taking our measurements, measure the copies, and correct whatever error we see?* Ah, a wise guy. Well, I have just the wet-blanket theorem for you!

Theorem 4.1 (No cloning; [NC11]). *Suppose we have a quantum machine with two "slots", an input (A) and a slot (B) that we will copy data from A into. Let $|\psi\rangle$ represent a state in A that we want to copy into B, and let $|e\rangle$ denote the state in B we want to overwrite (note that we can represent this combined system by $|\psi\rangle \otimes |e\rangle$). Then there exists no unitary operator U such that for all normalized $|\psi\rangle$,*

$$U(|\psi\rangle \otimes |e\rangle) = |\psi\rangle \otimes |\psi\rangle. \qquad \triangle$$

More tangibly, this means we cannot copy arbitrary, unknown quantum states.

Thus, we really can't weasel our way out of this situation. We need to develop fundamentally new error-correction techniques in order to utilize quantum computing

4.2. QEC Techniques

Although classical computing has failed us in implementation, we can be inspired by the concepts behind its methods to develop robust QEC codes. Here is a summary of the challenges we encouter, as enumerated by [NC11]:

1. We cannot clone arbitrary quantum superpositions. And even if we were able to, it would not be possible to compare the states we obtain to one another.
2. Errors are no longer discretized. In classical computing, we can have a bit flip, and that's pretty much it. In quantum computing, errors become continuous. We could have bitflip errors, phase flip errors, and anything in between!
3. Measurement destroys information! In classical error correction, we can observe an output from an operation, and decide whether the data has been corrupted, and if so, how to correct for it. Meanwhile, measurement of a quantum state makes recovery from error impossible, as it simply destroys data.

these sound like pretty insurmountable obstacles. But, as in most cases, it turns out that they are not.

4.2.1. Three qubit codes

Suppose we have a system of 3 qubits. Suppose too that between our initial state and our final measurement, there is a chance p for each qubit that we will see an output that has been bit flipped relative to the input, and hence by complement there is a $1-p$ chance the qubit will emerge having only undergone an identity transformation. The following *bit flip code*, inspired by the repitition code above, can help us correct for such errors. Using *ancillary* qubits, supplemental qubits containing information *only* about the state of the system, we can monitor whether a bit flip has occurred, and if so, how to fix it.

Let us denote the "logical" 0 state (in this case, $|000\rangle$) by $|0_L\rangle$. Similarly, let us denote the "logical" 1 state (in this case, $|111\rangle$) by $|1_L\rangle$. We consider the following circuit:

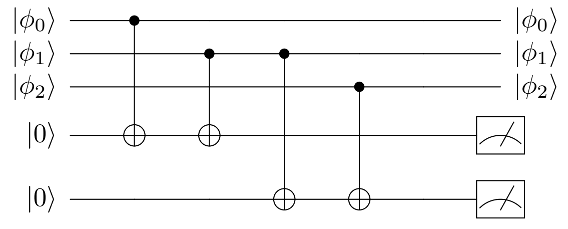

Figure 4: Three-qubit bit flip code.

The two $|0\rangle$ states are the *ancillary* qubits mentioned previously, and the semicircle things appended to the end of the circuit represent *measurement devices.*

So how does this work? Suppose our input has been encoded perfectly by the superposition $\alpha|000\rangle + \beta|111\rangle$. We feed the qubits in individually, with the i^{th} qubit into $|\phi_i\rangle$. But alas, a bit flip has occured! To fix it, we employ the followint procedure:

(a) First, we attempt to identify the error that has occurred. There are four possible errors in our system, and these correspond to eigenstates of the following projection operators ([NC11]):

$$\begin{aligned} P_0 &= |000\rangle\langle 000| + |111\rangle\langle 111| && \text{no error has occurred} \\ P_1 &= |100\rangle\langle 100| + |011\rangle\langle 011| && X \text{ on qubit 1} \\ P_2 &= |010\rangle\langle 010| + |101\rangle\langle 101| && X \text{ on qubit two} \\ P_3 &= |001\rangle\langle 001| + |110\rangle\langle 110| && X \text{ on qubit three} \end{aligned}$$

here, for each P_k, we are summing the projection operators representing the case that a bit flip occurs on $|0_L\rangle = |000\rangle$ (these correspond to the ones on the left hand side) and those representing the case that a bit flip occurs on a state that was input as $|1_L\rangle = |111\rangle$. The different cases above are referred to as the differet *error syndromes* of the system.

(b) Initially, before we take a measurement of the bottom two wires of our circuit, we have an *superposition* of the error syndromes, call it $|E\rangle$. For each of the P_i, the kets in the definition of P_i are eigenstates of P_i. Now, note that by virtue of being connected by CNOT gates, the $|0\rangle$ states at the bottom (together representing $|E\rangle$) have become entangled with the input $|\Phi\rangle$. At some point, we perform a measurement on $|E\rangle$ by checking the values of the ancillary qubits, and determine their states. Here, a $|1\rangle$ indicates that the two $|\phi_i\rangle$'s feeding the control bits must have been bit flipped relative to each other. Since the ancillary bits were entangled with the input $|\Phi\rangle$, we now know exactly which *error syndrome* the system has experienced, but do not know *any* information abot the coefficients α and β. Hence, we have preserved the information encoded in $|\Phi\rangle$!

(c) Supposing that the probability is very low that *more* than one qubit has flipped, we identify (based on our measurements of the syndrome) which bit has flipped. It is not difficult then to create quantum circuitry to apply the X gate to the flipped qubit.

(d) Hence, we have recovered the initial state, $\alpha|000\rangle + \beta|111\rangle$, having performed no measurements at all on the coefficients α, β.

This is a remarkable result. We have just successfully identified (and recovered from) a catastrophic error, all without measuring the actual state of the system itself!

There exist error-correcting codes for other forms of errors as well. As it turns out, the three qubit phase flip code is structurally similar to the circuit above, save for the addition of a few Hadamard gates [NC11]. Together, they can be composed to form the *Shor code*.

4.2.2. The Shor code, briefly

Admittedly, the Shor code is a little harder to understand than either the phase or bit flip codes. As such, I will not go into too much detail explaining it here, instead opting to move on quickly to the next section. But, the Shor code has the desirable property that it can correct for a bit flip, a phase flip, or both! What's more, the Shor code corrects *arbitrary* single-qubit errors, since it turns out that these two errors form a *basis* for the vector space of errors [Got97]. Hence, we can easily correct multi-qubit errors simply by creating many parallel branches of the Shor code, and inputting the qubits simultaneously.

In order to do so, we must redefine our logical "1" and "0" in some pretty funky ways:

$$|0_L\rangle = \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000 + |111\rangle\rangle)}{2\sqrt{2}}$$
$$|1_L\rangle = \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000 - |111\rangle\rangle)}{2\sqrt{2}}$$

and feed it into the circuit below [NC11].

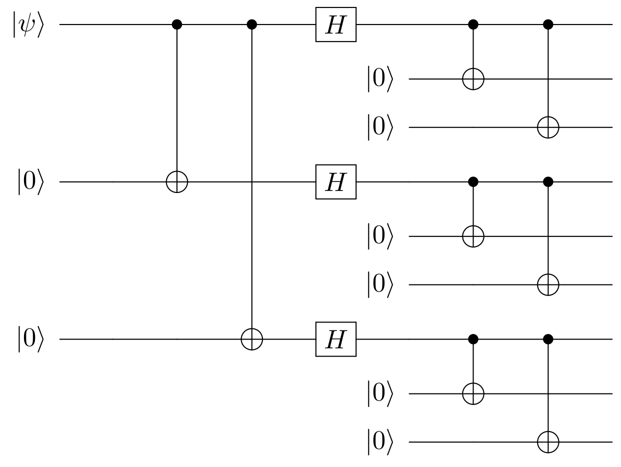

Figure 5: The Shor code, in all its glory.

5. A general theory

The Shor code introduces some nice ideas that we can employ to find strategies for generating *general* error-correction codes for some input of length n. Unfortunately, with generality comes math, and we'll need to think about some linear and abstract algebra in order to make headway here. This is is a big punchline though, so it'll be worth it.

5.1. When can we identify errors precisely?

In order to correct for errors, we first need a heuristic to determine whether or not we can identify them precisely. The key insight here comes in our discussion of three qubit codes. How did we identify error syndromes and distinguish them from one another? We used *projection operators*. Upon feeding them some input ket (representing a combined state of 3 basis kets), we were returned an eigenstate with eigenvalue 0 if the error they measure did not occur, and 1 if it did. This gives us an idea: the space of wavefunctions of our system forms a Hilbert space. By determining whether our state is an eigenfunction of some projection operator P, we identify whether it is *some element of the subspace* C that P projects onto [NC11]. Of course, we are not sure exactly *which* element, and that uncertainty is how we avoid collapsing our state and losing information. Once we have identified the error subspace, we can perform some *recovery* operator that maps the entire subspace C in such a way that any element of C would return to where it "should" have been originally. Again, since we haven't performed a measurement, we don't know the particulars of *where* the state goes upon recovery, only that it is in the right place.

With that conceptual overview, let's discuss exactly what C represents in a little more depth. C is a subspace of our hilbert space of states, such that single-qubit error operators (and sometimes multi-qubit error operators too) acting on an element of C will take it to a *distinguishably different subspace* [Got98]. This portion is key! An error E has to not only be detectable, but also *distinguishable* from other errors.

When could this fail? The answer is simple: whenever two of the error subspaces intersect nontrivially [NC11]. Or, equivalently stated, when the subspaces are not *orthogonal* to one another.[10] So really what we need is to develop a code that will correct for an *orthogonal basis* of errors.

5.2. ORTHOGONALITY AND ID'ING ERRORS

Suppose we are encoding k "logical" qubits in n physical qubits. Then any code that performs this task will correspond to a "coding subspace" of our Hilbert space with dimension 2^k (the dimension of the original Hilbert space is 2^n). Call this coding subspace T. If our code can correct a basis of error operators, then it can correct an arbitrary error.

It has been shown that one basis for the error space is the set of all tensor products of Pauli spin matrices [Got97]. These, together with the matrices we obtain by multiplying by a factor of -1 or $\pm i$, form what is known in abstract algebra as an *group* ([Got97]):

DEFINITION 5.1 (Group). Let G be a set of elements. G is called an *group* iff G is equipped with a function called a *binary relation*, denoted here as "+", such that $+ : G \times G \to G$, and

(a) $\forall a, b \in G$, we have $a + b \in G$.

(b) $\forall a, b, c \in G$, we have $(a + b) + c = a + (b + c)$

(c) $\exists e \in G$ s.t. $\forall a \in G$, $e + a = a = a + e$

(d) $\forall a \in G$, $\exists a^{-1} \in G$ s.t. $a + a^{-1} = e$. e is then referred to as the *identity element*.

if additionally, we have that $\forall a, b \in G$, $a+b = b+a$, then G is called *Abelian*, and $+$ is said to be *commutative*. We will also define subgroups at this point. △

DEFINITION 5.2 (Subgroup). Let $(G, +)$ be a group, and let $(H, +) \subseteq (G, +)$. Then $(H, +)$ is said to be a *subgroup* if $(H, +)$ is a group. When the group operation is understood, this is sometimes abbreviated by $H \leq G$, where we use $\leq$ instead of $\subseteq$ to highlight that H is closed under the group operation of G. △

I will use the same notation as in Gottesman's thesis, denoting the group of basis error tensors by $\mathcal{G}$, and $\mathcal{G}_n$ if it is specified that we are discussing the particular case of an n-qubit input. Here, since the group operation is standard matrix multiplication, we will not use a special symbol (such as "+") to denote it. I will define $\mathbb{G}$ to be the vector space of general error tensors generated by the elements of $\mathcal{G}$ together with the usual definitions of addition and scalar multiplication.[11]

We return to the issue of distinguishability. For our error code to correct multiple distinct errors, e.g. $E_a, E_b \in \mathbb{G}$ with $E_a \neq E_b$, then we need to satisfy two criteria [Got97]:

(a) Distinct errors acting on distinct basis states must yield different results. That is: let $|\psi_i\rangle$, $|\psi_j\rangle$ be basis states in T. As we stated earlier, the only way to guarantee distinguishability is for the subspaces to be orthogonal to one another, which necessitates $E_a|\psi_1\rangle$ being orthogonal to $E_b|\psi_2\rangle$. We can guarantee this by taking the projection operator of one basis state onto another after first composing the error operators. If they are indeed orthogonal, this should be identically 0. Hence, we must have
$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = 0$$
whenever $i \neq j$ [Got97].

(b) The second stipulation necessitates that we cannot learn anything about our state by performing a measurement of ancillary qubits. Hence, let $|\psi_i\rangle, |\psi_j\rangle$ be (again) a basis for our coding space. Then $\forall E_0, E_1 \in \mathbb{G}$, we must have
$$\langle\psi_i|E_a^\dagger E_b|\psi_i\rangle = \langle\psi_j|E_a^\dagger E_b|\psi_j\rangle.$$

We can combine these equations together into a single statement. We must have $\forall 0 \leq i, j \leq 2^k$, $\forall E_a, E_b \in \mathbb{G}$,
$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = C_{ab}\delta_{ij}$$
where C_{ab} is a Hermetian matrix, and δ_{ij} is the kronecker delta [Got97]. Because C_{ab} is Hermetian, we can

[10]Subspaces are said to be orthogonal to one another if, excluding the zero vector, every element of one is orthogonal to every element of the other.

[11]Yes I know it's not a field, but $\mathbb{G}$ was the easiest symbol to find.

employ the Spectral Theorem and diagonalize it. Finally, renormalizing as we need to, we can use this to obtain a new basis $\{F_a\}$ for the error space, with

$$\langle \psi_i | F_a^\dagger F_b | \psi_j \rangle = \delta_{ab}\delta_{ij}$$

Hence, we have our orthogonality condition (nonzero only if all the subscripts match), and thus we can always take a measurement of some sufficient arrangment of ancillary qubits to determine what error has occured, and use this as a heuristic to determine what correction operator to apply.

5.3. The Stabilizer formalism

In our analysis of quantum codes, it ends up being particularly rewarding to investigate those elements of $\mathbb{G}$ that leave the basis states of our code space unperturbed. That is, we want to examine those error operators E such that the basis states are eigenvectors of E with eigenvalue $+1$. This has a name in the field of group theory, the *stabilizer*. The algebra definition is pretty tricky and abstract, but we'll try to do it justice.

Definition 5.3 (Group action). Let G be a group with operation $\cdot$, and let Ω be some set. Let e be the identity element of G. Then an *action* of G on Ω is a function $\varphi : G \times \Omega \to \Omega$ such that

(a) $\forall x \in \Omega$, $\varphi(e, x) = x$.
(b) Let $g, h \in G$. Then $\forall x \in \Omega$, $\varphi(g, \varphi(h, x)) = \varphi(g \cdot h, x)$. △

Essentially, an action is a way of defining a sort of "permutation" on the set Ω, such that it preserves some form of the structure of the acting group G. Now, we can define stabilizers and normalizers.

Definition 5.4 (Stabilizers). Let G be a group acting on a set Ω by φ, and let $x \in \Omega$. Then the *stabilizer* of x in G, denoted $\mathrm{Stab}_G(x)$, is given by

$$\mathrm{Stab}_G(x) = \{g \in G \mid \varphi(g, x) = x\}.$$

importantly, the stabilizer of any element of Ω is a subgroup of G. △

Definition 5.5 (Normalizers). Let G be a group, and let H be a subgroup of G. $\forall g \in G$ let gH denote the set $\{gh \mid h \in H\}$, and similarly, let Hg denote $\{hg \mid h \in H\}$. Then the *normalizer* of H in G, denoted $\mathbf{N}_G(H)$, are those elements of g such that $gH = Hg$:

$$\mathbf{N}_G(H) = \{g \in G \mid gH = Hg\}. \quad △$$

There are rich connections to be made here. By reframing error correction codes (a relatively poorly-understood field) in the context of Abstract Algebra (a fairly mature field), we can fall back on results proven in Algebra to make finding error-correcting codes more tractable.

Let's consider the general case of correcting an error in a system of k logical qubits with coding space T encoded in n physical qubits. The set of all error operators that fix *all* of the basis states of T (i.e., the operators for which the basis states are eigenvectors of eigenvalue $+1$) are the elements of a stabilizer of T in $\mathcal{G}$ under some action. Call this subgroup $\mathcal{S}$. Then $\mathcal{S}$ is abelian, and we have the following theorem ([Got97]):

Theorem 5.1. $\forall E \in \mathcal{S} \cup (\mathcal{G} - \mathbf{N}_G(\mathcal{S}))$, *the quantum code corresponding to* $\mathcal{S}$ *will detect* E. *Further, the code will* correct *any set of errors* $\{E_i\} \subseteq \mathbb{G}$ *iff* $\forall E_a, E_b \in \{E_i\}$, *we have* $E_a E_b \in \mathcal{S} \cup (\mathcal{G} - \mathbf{N}_G(\mathcal{S}))$. *Note that this does not require* E_a *or* $E_b \in S \cup (\mathcal{G} - \mathbf{N}_G(\mathcal{S}))$, *only that their product is.* △

Hence, for some particular error basis group $\mathcal{G}$, we can use well-formulated algebraic results to find entire families of effective error-correcting codes.

6. Conclusion

Using this result, we can guarantee the existence of families of error-correcting codes for nearly all multi-qubit systems. What's more, these error-correcting codes are (often) fast enough to correct decoherence effects before they can damage the integrity of the computation. Thus, by drawing inspiration from classical coding theory (and results from linear and abstract algebra), researchers managed to find a away around the first major hurdle to making quantum computing a feasible reality.

That was almost 20 years ago. Since then, progress has exploded. Today, companies such as D-Wave have demonstrated functional quantum computers comprised of systems of just over 2000 qubits. Such devices are being used to perform various high-complexity optimization problems, such as those that occur in machine-learning research, as well as extremely high-fidelity materials simulations, with an example being protein annealing.

It will be exciting to see where the future of quantum computing leads. Whether or not the tech lives up to the hype, it is unquestionable that the field is replete with beautiful concepts in math, physics, and information theory. There remain many open problems to solve, and interesting questions yet to ask.

And if things do pan out computationally, we might finally be able to perform rapid, accurate simulations of reality on a quantum level. As Richard Feynman once said, "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical."

Well Mr. Feynman — we're getting there.

References

[Axl15] Sheldon Jay Axler. *Linear Algebra Done Right.* Undergraduate Texts in Mathematics. Springer International Publishing, third edition, 2015.

[Bar15] Stephen M. Barnett. Lecture notes on introduction to quantum information, September 2015.

[CRSS96] A. R. Calderbank, E. M Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction via codes over gf(4), 1996.

In this paper, the authors abstract the problem of finding quantum-error-correcting codes to finding "additive self-orthogonal subcodes" of a galois field. The rest of the paper is dedicated to studying the properties of such codes in this new framework.

[DNM09] Simon J. Devitt, Kae Nemoto, and William J. Munro. Quantum error correction for beginners. 2009.

In this paper, the authors give a large example-based review of the developments in QEC since the 1990s, as well as an introduction to the formal mathematics underlying the process. Describes how error-correction protocols can allow us to gain information about what error the system has experienced, without actually performing a measurement to determine the values of the coefficients on our basis states. Then goes very deep into the subject.

[Got97] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction.* PhD thesis, California Institute of Technology, May 1997.

[Got98] Daniel Gottesman. Notes on stabilizer codes, July 1998.

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[RP98] Eleanor G. Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. 1998.

In this paper, the authors give an introduction to the basics of quantum computing, focusing in particular on the differences between classical and quantum computers.

[Sch03] Maximilian Schlosshauer. Decoherence, the measurement problem, and interpretations of quantum mechanics. 2003.

In this paper, the author lays out a clear description of quantum decoherence and the measurement problem.

[Sho95a] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. 1995.

In this paper, the author lays out methods by which two classically "difficult" computational problems (factoring integers and finding discrete logarithms) can be computed in polynomial time by a quantum computer.

[Sho95b] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[TCCS13] Ming-Chung Tsai, Po-Chung Chen, Kuan-Peng Chen, and Zheng-Yao Su. Algebraic quantum error-correction codes, 2013.

In this paper, the authors give a formal mathematical treatment of quantum codes, and then describe how these might be generated systematically.

[Tow00] J.S. Townsend. *A Modern Approach to Quantum Mechanics.* International series in pure and applied physics. University Science Books, 2000.

In this amazing and wonderful and extremely well-written book, the illustrious author John Townsend, whom I have had the honor of meeting, lays out beautiful and concise explanations of the fundamentals of modern physics. A true masterpiece. (Please give me an A ;))

[Unr94] W. G. Unruh. Maintaining coherence in quantum computers. 1994.

In this paper, the author discusses the challenges of preventing the state of a quantum computer from decohering during the computational process. He establishes a rough limit on the time a computation can take, proportional to the termal time scale.