

Quantum Computing

Vaughn Sohn

December 12, 2024

Contents

1	Quantum Algorithms	2
1.1	Introduction	2
1.2	Elementary quantum algorithms using quantum parallelism	2
1.3	Hamiltonian simulations	5
1.4	Quantum Fourier transform	8
1.5	Phase estimation	11
1.6	Applications of phase estimation	15
1.7	Applications of the QFT	19
1.8	Quantum search algorithms	22
1.9	Amplitude estimation algorithm (Quantum counting)	26
1.10	HHL (Harrow–Hassidim–Lloyd) algorithm	28
1.11	Optimality of the quantum search algorithm	30
2	Introduction to Computational Complexity	33
2.1	Introduction	33
2.2	The class NP: Reducibility and completeness	33
2.3	Quantum complexity	36
3	Quantum Noise	38
3.1	Introduction	38
3.2	Axiomatic approach to quantum evolutions	38
3.3	Interpretation of quantum channels	40
3.4	Examples	41
4	Quantum Error Correction and Fault Tolerance	43
4.1	Introduction	43
4.2	Some distance measure	43
4.3	The basic code and Shor code	45
4.4	Theory of QEC	48
4.5	Constructing quantum codes	52
4.6	Stabilizer codes	55
4.7	Fault-tolerant quantum computation	63
A	Details of the proof	70
A.1	Optimality of the quantum search algorithm with M solutions	70
A.2	Period finding algorithms when r doesn't divide 2^t	70
A.3	Choi-Kraus theorem	70

Chapter 1

Quantum Algorithms

Lecture 9

1.1 Introduction

7 Oct. 10:30

이번 챕터에서 우리는 *Quantum Algorithm*에 대해 다루고자 한다. Quantum algorithm은 quantum circuit이나 quantum computer에서 구현되는 알고리즘을 지칭한다. Classical computer가 어려운 문제를 해결하기 위하여 만들어진 것처럼, quantum algorithm에 대해서 공부하고 새로운 방식을 고안하는 것은 quantum computer의 동작방식과 quantum computer의 한계를 분석하기 위한 중요한 과제이다. Quantum computer라는 개념이 등장하고 나서부터 지금까지 많은 종류의 quantum algorithm들이 고안되어 왔다. 이번 강의에서 다루고자 하는 quantum algorithm은 다음과 같다.

- Elementary quantum algorithms
- Hamiltonian simulations
- Quantum Fourier transform
- Phase estimation
- Quantum search algorithm (Grover search algorithm)
- Amplitude amplification / estimation algorithms
- HHL algorithm

1.2 Elementary quantum algorithms using quantum parallelism

Quantum mechanics만의 특징을 이용할 수 있는 quantum computer는 *Quantum parallelism*이라는 특성을 가진다. 이는 quantum computer가 특정 oracle; function $f(x)$ 에 대하여 동시에 여러개의 입력에 대한 결과를 병렬적으로 얻을 수 있다는 의미이다. 고전적인 개념인 $f(x)$ 를 quantum computer에서 실행하기 위해서는, quantum computer의 연산단위인 *unitary operator*로 함수를 표현해야하는 필요가 있다.

먼저, 간단한 one-bit boolean function $f : \{0, 1\} \rightarrow \{0, 1\}$ 를 생각해보자. 직관적으로, 이 함수에 대응되는 operator는 다음과 같이 설계할 수 있다. 이 operator는 $f(0) = 1$ 이라면, $U_f |0\rangle = |1\rangle$ 처럼 동작하도록 quantum gate들을 이용하여 구현된다.

$$|x\rangle \xrightarrow{U_f} |f(x)\rangle$$

그러나 이러한 방식으로 operator를 설계하게 되면, *non-invertible* 함수 $f(x)$ 에 대한 operator는 더이상 unitary 조건을 만족하지 못한다. 따라서 이 문제를 해결하기 위하여 control을 수행하는 추가적인 input qubit을 추가하여 unitary operator가 되도록 설계한다.

Definition 1.2.1 (Unitary oracle). 함수 f 에 대한 unitary operator U_f 는 다음과 같이 정의된다.

$$|x\rangle |y\rangle \xrightarrow{U_f} |x\rangle |y \oplus f(x)\rangle$$

$|x\rangle$ 는 f 의 입력으로 사용되는 **oracle qubit**이며, $|y\rangle$ 는 1일 때는 $f(x)$ 의 결과를 flip 시키고 0일 때는 $f(x)$ 의 결과를 반환하는 역할을 수행한다. 그럼 이렇게 설계한 unitary operator에 *superposition state*를 $|x\rangle$ 로 제공하면 결과적으로 우리는 다음과 같은 two-qubit state를 얻게 된다.

$$|+\rangle|y\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle)$$

즉, one-bit boolean function에서 가능한 모든 입력 0, 1에 대한 출력을 U_f 를 한 번 호출함으로써 얻게 된 것이다! Classical computer에서 병렬연산은 서로 다른 컴퓨팅 자원을 사용할 뿐, f 를 여러번 호출해야 하는 사실은 변하지 않지만, quantum computer에서의 병렬연산은 실제로 f 를 한 번 호출하여 모든 연산을 수행할 수 있다.

이렇게 어떤 함수 f 에 대응되는 unitary operator를 설계하는 것은 n-bit boolean function에 대해서 쉽게 일반화할 수 있다. n-bit boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 에 대응되는 unitary operator는 Definition 1.2.1을 이용하여 쉽게 설계할 수 있으며, 더 나아가 n-qubit에 대한 superposition state를 입력으로 제공하면 다음 결과를 얻게된다.

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

이 회로를 이용하면 한 번의 연산만으로 2^n 개의 입력에 대한 출력을 동시에 얻을 수 있다. 그러나 한 가지 주의해야 할 점은, 우리가 결과를 측정하는 순간 여러개의 superposition output은 사라지고 확률에 따라서 단 하나의 결과만을 얻을 수 있다는 사실이다. 따라서 이러한 quantum parallelism만의 독특한 특징을 잘 활용하여 효과적인 연산을 수행할 수 있도록 알고리즘을 설계하는 것이 중요하다.

이 강의에서는 parallelism의 장점을 활용하는 대표적인 알고리즘들(e.g., Deutsch's algorithm, Deutsch-Josza algorithm, Simon's algorithm, and Bernstein-Vazirani algorithm)에 대해 소개한다.

1.2.1 Deutsch's algorithm

Deutsch's algorithm은 주어진 one-bit boolean function f 이 **balance**인지 **constant**인지를 판단하는 문제를 해결한다.¹ Classical computer가 이 문제를 해결하기 위해서는 두 합수값 $f(0), f(1)$ 을 비교하기 위해서 반드시 2번의 함수 호출이 필요하다. 그러나, 지금부터 우리는 quantum computer는 단 한번의 gate call만으로 이 문제를 해결할 수 있음을 보이고자 한다.

1. input state : 우리는 다음과 같은 state를 input으로 제공한다.

$$|\psi\rangle = |0\rangle|1\rangle$$

2. apply Hadamard gates on both qubits : H gate를 가하면, 다음과 같은 상태로 변화한다.

$$|\psi\rangle = |+\rangle|- \rangle$$

3. apply U_f : operator를 통과한 state는 다음과 같다.²

$$\begin{aligned} |\psi\rangle &= U_f \left(\frac{1}{2} (|00\rangle + |10\rangle - |01\rangle - |11\rangle) \right) \\ &= \frac{1}{2} (|0, f(0)\rangle|1, f(1)\rangle - |0, 1 \oplus f(0)\rangle - |1, 1 \oplus f(1)\rangle) \\ &= \frac{1}{2} \left(|0\rangle(-1)^{f(0)}(|0\rangle - |1\rangle) + |1\rangle(-1)^{f(1)}(|0\rangle - |1\rangle) \right) \\ &= \frac{1}{2} \sum_{x \in \{0,1\}} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

따라서 각 case에 대해 state는 다음의 2가지 모습을 띠게 된다.

$$|\psi\rangle = \begin{cases} \pm |+\rangle|- \rangle & \text{if } f(0) = f(1) \\ \pm |- \rangle|- \rangle & \text{if } f(0) \neq f(1) \end{cases}$$

¹constant: $f(0) = f(1)$, balance: $f(0) \neq f(1)$

² $|f(x)\rangle - |1 \oplus f(x)\rangle$ 는 $f(x)$ 의 값에 따라서, $|0\rangle - |1\rangle$ ($f(x) = 0$) 또는 $|1\rangle - |0\rangle$ ($f(x) = 1$)이 된다.

4. apply again Hadamard gates on oracle qubit : 마지막으로 oracle qubit에 H gate를 가하면, f 의 종류에 따라서 다음과 같은 상태가 된다.

$$|\psi\rangle = \begin{cases} \pm |0\rangle |- \rangle & \text{if } f(0) = f(1) \\ \pm |1\rangle |- \rangle & \text{if } f(0) \neq f(1) \end{cases}$$

따라서, oracle qubit을 측정하면, 100%의 확률로 0 또는 1의 값을 얻게될 것이며, 그 값에 따라서 우리는 f 가 balance인지 constant인지를 다음 규칙에 따라서 쉽게 판단할 수 있다. \square

$$|\psi\rangle = \begin{cases} \text{constant} & \text{if } q_o = 0 \\ \text{balance} & \text{if } q_o = 1 \end{cases}$$

Deutsch's algorithm은 classical algorithm보다 quantum algorithm 알고리즘이 더 효과적임을 보인 첫 번째 알고리즘이다. 하지만, 그 효과는 단지 2번의 호출을 1번으로 줄일 뿐이다. 따라서 지금부터 더 효과적인 알고리즘들에 대해서 소개하고자 한다.

Lecture 10

1.2.2 Deutsch-Jozsa algorithm

14 Oct. 10:30

Deutsch-Jozsa algorithm은 간단히 말하자면, one-bit boolean function에 대한 문제인 Deutsch-Jozsa algorithm을 n -bit boolean function에 대한 문제로 일반화한 것이다. 즉, n -bit boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 이 **constant**인지 **balanced**인지를 판단하는 문제를 해결하는 알고리즘이다.³ 마찬가지로 classical computer가 이 문제를 해결하기 위해서는 최대 $2^{n-1} + 1$ 개의 function value를 비교해야하기 때문에 $2^{n-1} + 1$ 번의 함수 호출을 필요로 한다. 그러나, 지금부터 이런 문제를 해결하려고 할 때도, quantum computer는 단 한번의 gate call만으로 이 문제를 해결할 수 있음을 보일 것이다.

1. input state : 우리는 다음과 같은 state를 input으로 제공한다. $\{0, 1\}^n$ 의 가능한 모든 input을 나타내기 위하여, oracle qubit은 n 개의 qubit으로 구성된다.

$$|\psi\rangle = |0\rangle^{\otimes n} |1\rangle$$

2. apply Hadamard gates on both qubits: 이때, $|+\rangle^{\otimes n}$ 은 가능한 모든 2^n 개의 n -bit string들의 superposition이기 때문에, 다음과 같이 표현할 수 있다.

$$\begin{aligned} |\psi\rangle &= |+\rangle^{\otimes n} |- \rangle \\ &= \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \end{aligned}$$

3. apply U_f : operator를 통과한 state는 다음과 같다. (Deutsch's algorithm의 표현을 이용하자)

$$\begin{aligned} |\psi\rangle &= U_f \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x \in \{0,1\}^n} |x\rangle (|0\rangle - |1\rangle) \right) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

4. apply again Hadamard gates on oracle qubit : H gate를 임의의 n -qubit computational basis에 가한 결과는 다음과 같다. 이는 single qubit에 대한 동작을 n -qubit에 대해 독립적으로 적용한 결과이다.

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle \quad (1.1)$$

이를 이용하여 H gate를 적용한 state를 표현하면, 다음과 같다.

$$|\psi\rangle = \frac{1}{2^n} \sum_{x, z \in \{0,1\}^n} (-1)^{f(x) + x \cdot z} |z\rangle |- \rangle$$

³여기서 말하는 balanced는 2^n 개의 input중에서 2^{n-1} 개의 input에 대한 결과가 0이고, 나머지 2^{n-1} 개의 input에 대한 결과가 1인 경우를 의미한다.

만약 $f(x)$ 가 constant function이라면, $\forall x$ 에 대해서 $f(x)$ 의 값은 항상 동일하기 때문에, $(-1)^{f(x)}$ 의 값이 $+1$, 또는 -1 이라는 constant가 되어 다음과 같이 나타낼 수 있다.

$$|\psi\rangle = \pm \frac{1}{2^n} \sum_{x,z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle |-\rangle$$

$z = 0^n$ 인 경우를 가정해보자. 이는 x 가 어떤 값이 되던지간에 $x \cdot z$ ⁴의 값이 0이 되기 때문에, 다음과 같이 표현할 수 있게 된다. $|0\rangle^{\otimes n}$ 의 amplitude의 square norm이 1이기 때문에, 100% 확률로 0^n 을 측정할 수 있다.

$$\pm \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot 0^n} |0\rangle^{\otimes n} |-\rangle = \pm \frac{1}{2^n} 2^n |0\rangle^{\otimes n} |-\rangle = |\pm |0\rangle^{\otimes n} |-\rangle|$$

반면, $f(x)$ 가 balance function이라면, $f(x)$ 의 값이 0인 경우와 1인 경우가 정확히 $1/2$ 씩 나타나기 때문에, 각 항들이 소거되면서 $z = 0^n$ 에 대한 amplitude가 0이 된다.

$$\left(\frac{1}{2^n} 2^{n-1} |0\rangle^{\otimes n} |-\rangle \right) + \left(-\frac{1}{2^n} 2^{n-1} |0\rangle^{\otimes n} |-\rangle \right) = |0|0\rangle^{\otimes n} |-\rangle|$$

따라서, oracle qubit을 측정한 결과가 0^n 인지 확인하여, f 가 constant인지 balanced인지를 알 수 있다. \square

$$|\psi\rangle = \begin{cases} \text{constant} & \text{if } q_o = 0^n \\ \text{balance} & \text{if } q_o \neq 0^n \end{cases}$$

1.3 Hamiltonian simulations

Hamiltonian simulation은 quantum computer가 고안된 핵심적인 이유 중 하나이다. 리처드 파인만의 말을 인용하자면 자연, 그중에서도 특히 미시세계는 고전역학을 따르지 않기 때문에, quantum mechanical을 따르는 simulation이 필요하다.

Note. *Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.*

Hamiltonian simulation은 간단히 말해 quantum state의 time evolution을 구하는 것이다. Schrödinger equation에 의하면, initial state $|\psi(0)\rangle$ 의 시간 t 에 대한 time evolution은 다음과 같이 주어진다.

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$$

이 방정식을 곧바로 해결하여 문제를 풀 수도 있지만, Hamiltonian matrix의 크기가 exponential하게 증가하기 때문에 이를 classical algorithm으로 효과적으로 해결하기는 어렵다.

1.3.1 Solution of Hamiltonian simulations

우리는 k-local Hamiltonian을 이용하여 Hamiltonian simulation 문제를 해결하고자 한다.

Definition 1.3.1. k-local^a Hamiltonian은 주어진 H 가 n -qubit system에서 최대 k 개의 system에 대해서만 동작하는 H_i 들의 합으로 표현되는 Hamiltonian이다. (이때 L 은 n 에 대해 polynomial이다.)

$$H = \sum_{i=1}^L H_i$$

^a여기서 local은 geometrically local이 아니라 단순히 system의 '개수'를 나타내기 위해 사용된다.

Example. 예를 들어, 다음과 같이 정의되는 Hamiltonian은 2개의 system (2, 4)에 대해서만 동작한다.

$$H_i = I_1 \otimes Z_2 \otimes I_3 \otimes X_4 \otimes I_{\perp}$$

⁴ $x \cdot z = x_1 z_1 + x_2 z_2 + \cdots + x_n z_n \mod 2$

Hamiltonian이 k-local Hamiltonian이라고 가정할 때, 이를 이용하여 어떻게 문제를 해결할 수 있을까? 그 아이디어는 단순하다. e^{-iHt} 를 계산하는 것은 어렵지만, e^{-iH_it} 는 최대 $k << n$ subsystem에만 작용하기에 더 단순하며, 그 단순성 덕분에 quantum circuit을 사용하여 시뮬레이션 하기 쉽다.

만약, $[H_j, H_k] = 0$ ⁵이 성립한다면 $e^{-iHt} = e^{-i\sum H_j t} = \prod e^{-iH_j t}$ ⁶가 성립하기 때문에, 각각의 H_i 에 대한 time evolution을 독립적으로 계산한 결과를 곱하여 전체 time evolution을 쉽게 구할 수 있다. 그러나 일반적으로 $[H_j, H_k] = 0$ 은 성립하지 않기 때문에, 우리는 Trotter formula를 이용한다.

Theorem 1.3.1 (Trotter formula). Hermitian operator H_j, H_k 에 대하여, 어떤 t 에 대해서도 다음이 성립한다.^a

$$e^{i(H_j+H_k)t} = \lim_{m \rightarrow \infty} \left(e^{iH_j t/m} e^{iH_k t/m} \right)^m$$

이는 H_j, H_k 가 not-commute라 하더라도 항상 성립한다.

^a직관적으로, 주어진 time t 대신에, 매우 짧은 시간간격인 t/m 에 대한 time evolution을 m 번 반복하여 t 에 대한 time evolution을 근사할 수 있다는 것을 의미한다.

따라서 $L = 2$ 인 k-local Hamiltonian $H = H_1 + H_2$ 에 대해, Hamiltonian simulation을 수행하는 propagator $\tilde{U}(t) = e^{-iH_1 t} e^{-iH_2 t}$ 를 시간에 대해서 미분하면, 다음과 같이 전개할 수 있다. (with initial condition $\tilde{U}(0) = I$)

- Eq. (1.3): H_1, H_2 는 matrix라서 순서를 바꿀 수 없기에, 다른 항을 더하여 항을 정리하고 다시 뺀다.
- Eq. (1.4): commutator를 사용하여 항을 정리한다.

$$i \frac{d\tilde{U}(t)}{dt} = H_1 e^{-iH_1 t} e^{-iH_2 t} + e^{-iH_1 t} H_2 e^{-iH_2 t} \quad (1.2)$$

$$= (H_1 + H_2) e^{-iH_1 t} e^{-iH_2 t} + e^{-iH_1 t} H_2 e^{-iH_2 t} - H_2 e^{-iH_1 t} e^{-iH_2 t} \quad (1.3)$$

$$= H e^{-iH_1 t} e^{-iH_2 t} + [e^{-iH_1 t}, H_2] e^{-iH_2 t} \quad (1.4)$$

$$= H \tilde{U}(t) + [e^{-iH_1 t}, H_2] e^{-iH_2 t} \quad (1.5)$$

미분 방정식에 대한 Duhamel's formula⁷를 이용하면, $\tilde{U}(t)$ 를 다음과 같이 구할 수 있다.

$$\tilde{U}(t) = U(t) - i \int_0^t e^{-iH(t-s)} [e^{-iH_1 s}, H_2] e^{-iH_2 s} ds \quad (1.6)$$

1.3.2 Performance

그렇다면, 이제 실제 Hamiltonian에 대한 time evolution $U(t) = e^{-iHt}$ 와 Trotter formula를 이용하여 근사한 $\tilde{U}(t)$ 의 error가 어떻게 bound되는지 분석해보자. Eq. (1.6)의 항을 이항시키면, 두 operator의 차이에 대한 norm의 upper bound를 구할 수 있다.⁸

$$\|\tilde{U}(t) - U(t)\| = \left\| -i \int_0^t e^{-iH(t-s)} [e^{-iH_1 s}, H_2] e^{-iH_2 s} ds \right\| \leq \int_0^t \| [e^{-iH_1 s}, H_2] \| ds \quad (1.7)$$

이때, norm은 다음과 같이 정의된다.

Definition 1.3.2.

$$\|A\| \triangleq \max_{|\psi\rangle} \|A|\psi\rangle\|_2 = \max_{|v\rangle \neq 0} \frac{\|A|v\rangle\|_2}{\||v\rangle\|_2}$$

이제 이 norm이 적절한 error rate ϵ 에 의하여 bound됨을 보임으로써, 이 Hamiltonian simulation을 실제로 활용할 수 있을지 분석할 수 있다.

⁵ $[H_j, H_k] = H_j H_k - H_k H_j$

⁶ 지수함수의 성질이 성립하기 위해서 필요한 조건. 실수들은 항상 commutative하기 때문에 이 성질이 자명하게 정립되었던 것

⁷ See https://en.wikipedia.org/wiki/Duhamel%27s_principle

⁸ $e^{-iH_i t}$ 는 unitary operator이기 때문에 norm을 변화시키지 않기 때문에 무시해도 된다.

이를 위하여 다음과 같은 새로운 연산자를 가정하자.

$$G(t) \triangleq [e^{-iH_1 t}, H_2] e^{iH_1 t} = e^{-iH_1 t} H_2 e^{iH_1 t} - H_2, \text{ with } G(0) = 0 \quad (1.8)$$

이 연산자에 대하여 $\tilde{U}(t)$ 처럼 시간에 대한 도함수를 구하면 다음과 같고,

$$i \frac{d}{dt} G(t) = e^{-iH_1 t} [H_1, H_2] e^{iH_1 t}$$

양변을 t 에 대해서 적분하면 다음과 같다.

$$G(t) = G(0) - i \int_0^t e^{-iH_1 s} [H_1, H_2] e^{iH_1 s} ds$$

norm을 취하고 triangle inequality를 이용하면, $\|G(t)\|$ 에 대한 upper bound를 얻는다.

$$\|G(t)\| = \left\| -i \int_0^t e^{-iH_1 s} [H_1, H_2] e^{iH_1 s} ds \right\| \leq \int_0^t \| [H_1, H_2] \| ds = t \| [H_1, H_2] \| \quad (1.9)$$

이때, $G(t)$ 의 norm은 자기자신의 정의 (1.8)에 의하여 다음 관계가 성립하게 된다.

$$\|[e^{-iH_1 t}, H_2]\| = \|G(t)\| \leq t \| [H_1, H_2] \|$$

따라서 이를 Eq. (1.7)에 대입하면, 다음을 얻는다.

$$\|\tilde{U}(t) - U(t)\| \leq \int_0^t \underbrace{\|[e^{-iH_1 s}, H_2]\|}_{s \| [H_1, H_2] \|} ds \leq \int_0^t s \| [H_1, H_2] \| ds$$

$\|[H_1, H_2]\|$ 는 s 에 관계없는 상수이기 때문에, 다음과 같이 정리할 수 있으며,

$$\|\tilde{U}(t) - U(t)\| \leq \frac{t^2}{2} \| [H_1, H_2] \|$$

commutator를 전개한 뒤, triangle inequality와 norm의 성질⁹을 이용하면 다음과 같이 전개할 수 있다.

$$\begin{aligned} \|\tilde{U}(t) - U(t)\| &\leq \frac{t^2}{2} \| [H_1, H_2] \| = \frac{t^2}{2} \| H_1 H_2 + (-H_2 H_1) \| \leq \frac{t^2}{2} (\| H_1 H_2 \| + \| H_2 H_1 \|) \\ &\leq \frac{t^2}{2} 2 \| H_1 \| \| H_2 \| \leq t^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \end{aligned}$$

정리하면, 다음과 같다.

$$\|\tilde{U}(t) - U(t)\| = \|U(t) - \tilde{U}(t)\| \leq t^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \quad (1.10)$$

따라서 시간간격 Δt 에 대한 error는 다음과 같이 bound된다.

$$\| e^{-iH\Delta t} - (e^{-iH_1 \Delta t} e^{-iH_2 \Delta t}) \| \leq (\Delta t)^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \quad (1.11)$$

Trotter formula에 의하여 시간간격 $\Delta t = t/m$ 에 대하여 m 단계 근사의 오차는 Δt 에 대한 단일 단계 오차의 누적으로 생각할 수 있기 때문에, 다음을 얻는다.

$$\left\| e^{-iHt} - (e^{-iH_1 \Delta t} e^{-iH_2 \Delta t})^m \right\| \leq m \frac{t^2}{m^2} \max \{ \| H_1 \|, \| H_2 \| \}^2 = \frac{t^2}{m} \max \{ \| H_1 \|, \| H_2 \| \}^2$$

이 bound를 사용하면, 우리가 원하는 target error rate ϵ 을 달성하기 위해서 필요한 m 의 값을 $m = O(t^2 \epsilon^{-1})$ 으로 결정할 수 있다.¹⁰

이 과정을 더 많은 term을 가지는 k-local Hamiltonian에 대해서 일반화할 수 있다.

$$\left\| e^{-i \sum_{i=1}^L H_i \Delta t} - \prod_{j=1}^L e^{-i H_j \Delta t} \right\| = O \left(\frac{t^2}{L} \sum_{i < j} \| [H_i, H_j] \| \right).$$

⁹ $\|AB\| \leq \|A\| \|B\|$

¹⁰ $\max \{ \| H_1 \|, \| H_2 \| \}^2$ (i.e., $\| H_i \|$)의 값은 상수라서 무시하였다.

$\Delta t = t/m$ 로 가정하면 다음을 얻는다.

$$\left\| e^{-i \sum_{i=1}^L H_i t} - \left(\prod_{j=1}^L e^{-i H_j \Delta t} \right)^m \right\| = O \left(\frac{m \Delta t^2}{L} \sum_{i < j} \| [H_i, H_j] \| \right) = O \left(\frac{t^2}{m L} \sum_{i < j} \| [H_i, H_j] \| \right)$$

이때, $\|H_i\| = O(1)$ 이므로 $\sum \| [H_i, H_j] \| = L^2$ 이 되어 다음과 같이 bound된다.

$$\left\| e^{-i \sum_{i=1}^L H_i t} - \left(\prod_{j=1}^L e^{-i H_j \Delta t} \right)^m \right\| = O \left(\frac{Lt^2}{m} \right).$$

$m = O(Lt^2\epsilon^{-1})$ 로 설정하게되면 우리는 항상 target error rate ϵ 을 upper bound로 가지는 근사 operator $\tilde{U}(t)$ 를 구성할 수 있고 이를 이용하여 simulation 할 수 있다. 따라서, 이 solution은 주어진 k -local Hamiltonian에 대하여, time t 에 대한 quadratic overhead (i.e., error)를 가지고 simulation을 할 수 있음을 보여준다. 즉, simulation time이 증가하더라도 error rate은 polynomial하게 증가하게 된다. \square

반면, 2nd-Trotter formula를 이용하여 표현할 수도 있다.

$$e^{i(A+B)\Delta t} = e^{iA\Delta t/2} e^{iB\Delta t} e^{iA\Delta t/2} + O((\Delta t)^3)$$

이를 이용하면, time t 에 대하여 error는 다음과 같이 표현된다.

$$\left\| e^{-i(A+B)t} - \prod_{i=1}^m e^{-iA\Delta t/2} e^{-iB\Delta t} e^{-iA\Delta t/2} \right\| = O(m(\Delta t)^3) = O(t^3/m^2)$$

즉, $m = O(t^{3/2}\epsilon^{1/2})$ 으로 설정하면, target error rate ϵ 을 달성할 수 있으며 이는 1st-Trotter formula를 사용한 simulation보다 더 효율적이다.¹¹ 2nd-Trotter 방법을 L 개의 term을 갖는 Hamiltonian에 대하여 일반화하면 m 은 다음과 같다.

$$m = O \left(\frac{\left(\sum_{j=1}^L \|H_j\| t \right)^{3/2}}{\epsilon^{1/2}} \right)$$

더 나아가 p th order Trotter formula를 사용하면, m 은 다음과 같다.

$$m = O \left(\frac{\left(\sum_{j=1}^L \|H_j\| t \right)^{1+1/p}}{\epsilon^{1/p}} \right)$$

Hamiltonian simulation을 위하여 다양한 연구들이 현재까지도 진행되고 있으며, 대표적인 알고리즘들은 다음을 참고하라.

- Higher-order product formula [Chi+21]
- Linear combination of unitary (LCU) [BCK15]
- Quantum signal processing (*optimal*) [Haa19]

Lecture 11

1.4 Quantum Fourier transform

16 Oct. 10:30

1.4.1 Quantum Fourier transform

Quantum Fourier transform은 Shor algorithm이나 HHL algorithm과 같은 다양한 quantum algorithm에서 사용되는 중요한 알고리즘이다. QFT에 대해서 소개하기에 앞서, 먼저 classical Fourier transform에 대해서 간단히 알아보자.

¹¹time t 에 대한 $3/2$ overhead

Definition 1.4.1 (Discrete Fourier transform). Discrete Fourier transform은 vector $\mathbf{x} \in \mathbb{C}^N$ 을 입력으로 받아서, 다음과 같이 정의되는 연산을 수행하여 output vector $\mathbf{y} \in \mathbb{C}^N$ 으로 변환하는 과정이다.^a

$$y_k \triangleq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$$

^a $\mathbf{x} = x_0 \cdots x_{N-1}$, $\mathbf{y} = y_0 \cdots y_{N-1}$

Definition 1.4.2 (Quantum Fourier transform). Quantum Fourier transform은 DFT와 유사하게, quantum state vector $|x\rangle = |x_0 \cdots x_{N-1}\rangle$ 을 입력으로 받아서, output quantum state vector $|y\rangle = |y_0 \cdots y_{N-1}\rangle$ 으로 변환하는 과정이다. 단, DFT와는 다르게 computational basis $\{|0\rangle, \dots, |N-1\rangle\}$ 에 대한 변환만이 정의되어 있다.

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

따라서 임의의 state vector $|x\rangle$ 의 transform은 변환된 basis vector $|k\rangle$ 들의 linear combination으로 표현하게 된다.

$$\sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{j=0}^{N-1} x_j \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle,$$

이때, y_k 는 다음과 같다.^a

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}.$$

^aDFT와 똑같다.

즉, QFT가 하는 일은 basis $\{|k\rangle\}$ 에 대한 linear combination으로 표현된 벡터 $|x\rangle$ 를 다른 basis $\{|j\rangle\}$ 에 대한 linear combination으로 나타내는 basis transform이다.¹²

이제 QFT가 어떻게 구현되는지 알아보자. $|j\rangle$ 에 대한 변환을 수행하는 QFT의 circuit은 다음과 같다.

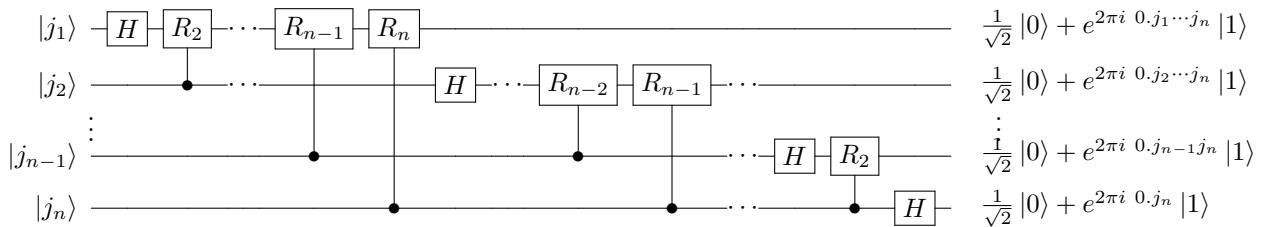


Figure 1.1: QFT circuit

QFT가 n -qubit system에 대해서 computational basis를 변환시킬 때, 우리는 computational basis를 나타내기 위하여 binary representation을 도입하고자 한다. 어떤 $|j\rangle \in \{|0\rangle, \dots, |2^n - 1\rangle\}$ 에 대한 binary representation은 다음과 같다. ($N = 2^n$)

$$j = j_1 j_2 \cdots j_n = j_1 2^{n-1} + \cdots + j_n 2^0 = \sum_{k=1}^n j_k 2^{n-k}.$$

또한, 소수도 다음과 같은 binary representation으로 표현할 수 있다.

$$0.j_l j_{l+1} \cdots j_m = j_l/2 + j_{l+1}/2^2 + \cdots + j_m/2^{m-l+1} = \sum_{k=l}^m j_k / 2^{k-l+1}$$

¹² $\{|k\rangle\}$ basis에서의 amplitude는 x_i 이며, $\{|j\rangle\}$ basis에서의 amplitude는 y_i 이다.

Binary representation을 이용하면 QFT의 연산을 다음과 같이 분석할 수 있다.

- Eq. (1.12): Definition 1.4.2에 따라, $|j\rangle$ 에 대한 QFT는 다음과 같이 표현된다.
- Eq. (1.13): $k = \sum k_l 2^{n-l}$ 으로 $k/2^n = \sum k_l 2^{-l}$ 이다.
- Eq. (1.14): $e^{a+b} = e^a e^b$, 그리고 $|k\rangle$ 가 n -qubit에 대해 tensor product로 표현됨을 이용한다.
- Eq. (1.15): 표현 단순화. ($\sum_{k_1, k_2, \dots, k_n \in \{0,1\}} \equiv \sum_{k_l \in \{0,1\}}$ 로 표현)
- Eq. (1.16): (1) $k_l = 0$, then $e^{2\pi i j k_l 2^{-l}} = e^0$. (2) $k_l = 1$, then $e^{2\pi i j k_l 2^{-l}} = e^{2\pi i j 2^{-l}}$
- Eq. (1.17): 모든 tensor product들을 전개한뒤 fraction을 binary representation으로 표현한다.

$$|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle \quad (1.12)$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1, \dots, k_n\rangle \quad (1.13)$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 \bigotimes_{l=1}^n e^{2\pi i j k_l 2^{-l}} |k_l\rangle \quad (1.14)$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \sum_{k_l=0}^1 e^{2\pi i j k_l 2^{-l}} |k_l\rangle \quad (1.15)$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n [|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle] \quad (1.16)$$

$$= \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1j_2\dots j_n} |1\rangle)}{2^{n/2}} \quad (1.17)$$

즉, $|j\rangle$ 의 각 qubit $|j_i\rangle$ 에 대해 독립적으로 특정 gate U 를 적용하여 $|0\rangle + e^{2\pi i 0.j_i\dots j_n}$ 가 되도록 quantum circuit을 설계하면, 그 결과가 QFT에 해당한다는 사실을 알아냈다.

$$U |j_i\rangle \rightarrow |0\rangle + e^{2\pi i 0.j_i\dots j_n} |1\rangle$$

본격적으로 quantum circuit을 만들기 위해서 rotation operator R_k 를 다음과 같이 정의하자. 이렇게 정의한 operator는 $|0\rangle$ 에 대해서는 아무것도 수행하지 않지만, $|1\rangle$ 에 대해서는 phase $e^{2\pi i / 2^k}$ 를 적용한다.

$$R_k \triangleq \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{pmatrix}$$

Fig. 1.1의 연산을 단계별로 따라가보자.

1. input state : 다음과 같은 input state로 시작한다. 이는 computational basis state 중 하나이다.

$$|\psi\rangle = |j_1, \dots, j_n\rangle$$

2. apply Hadamard gate on the first qubit : 첫 번째 qubit; $|j_1\rangle$ 에 H 를 적용하면, 다음을 얻는다. ¹³

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^{j_1} |1\rangle) |j_2, \dots, j_n\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle$$

3. apply controlled- R_2 gate with the first qubit as the target and the second qubit as control.

$$\begin{aligned} |\psi\rangle &= \begin{cases} \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle & \text{if } j_2 = 0, \\ \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.01} e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle & \text{if } j_2 = 1 \end{cases} \\ &= \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle) |j_2, \dots, j_n\rangle. \end{aligned}$$

¹³ 때, $e^{i\pi} = e^{2i\pi \frac{1}{2}} = -1$ 그리고 $e^0 = e^{2i\pi \frac{0}{2}} = 1$ 라는 사실을 이용한다.

4. apply controlled- R_k gate consequently : 3번의 과정을 control qubit을 하나씩 증가시키면서 반복한다. 이때, control qubit의 순서가 i 번째라면, R_i gate를 적용해야한다.

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) |j_2, \dots, j_n\rangle$$

5. apply controlled- R_k gate consequently with other *target state* : 3-4번 과정을 다른 control qubit에 대해서 반복한다.

예를 들어, second qubit을 target qubit으로서 가정하면, 다음과 같은 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^2}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_2 j_3 \cdots j_n} |1\rangle) |j_3, \dots, j_n\rangle$$

일반화하면, j 번째 qubit에 대해, $k(j < k)$ 번째 qubit을 control qubit으로 controlled- R_{k-j} gate를 차례대로 적용하는 과정을 반복한다.

6. n 번째 qubit까지 이 과정을 수행하면 최종적으로 다음 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_2 j_3 \cdots j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_n} |1\rangle)$$

7. 따라서 SWAP gate를 적용하여 qubit의 순서를 반대로 뒤집으면, 우리가 구하고자하는 QFT의 결과 (see Eq. (1.17))를 얻을 수 있다. \square

따라서 우리는 1번의 QFT circuit을 수행하는 것으로 n 개의 data에 대한 QFT결과를 동시에 얻을 수 있다. (단, 관측하면 확률에 따라 하나의 결과만 얻게 된다.)

1.4.2 Performance

그렇다면, QFT가 가지는 gate complexity가 얼마인지 분석해보자. 알고리즘에 따라 첫 번째 qubit에 대해서 1개의 H -gate, 그리고 $(n-1)$ 개의 controlled rotation gate를 필요로한다. 두 번째 qubit에 대해서는 1개의 H -gate, 그리고 $(n-2)$ 개의 controlled rotation gate를 필요로한다. 따라서 n 개의 qubit에 대해 모두 필요한 gate의 개수는 다음과 같다.¹⁴

$$n + n - 1 + \cdots + 1 = \frac{n(n+1)}{2} = \Theta(n^2)$$

Some Remarks

- QFT는 exponential speed-up을 달성하는 것처럼 보이지만, 실제로는 값을 관측하게 되면 하나의 테이터에 대한 결과만 얻을 수 있다.
- 또한, input state $|j\rangle$ 를 준비하는 과정도 효과적이지 못하다.
- 따라서 QFT를 활용하여 알고리즘을 설계하는 것은 쉽지 않다.

1.5 Phase estimation

1.5.1 Phase estimation

QFT를 활용하는 중요한 알고리즘 중 하나가 바로 *phase estimation*이다. phase estimation이 해결하고자 하는 문제에 대해서 먼저 소개한다. 다음의 관계를 만족하는 unitary operator U 에 대해, 우리가 알고 있는 것은 오직 eigenvector인 $|u\rangle$ 이고 eigenvalue는 알지 못한다고 하자.

$$U|u\rangle = e^{2\pi i \varphi}|u\rangle$$

Phase estimation의 목적은 eigenvalue에 대한 phase $\varphi \in [0, 1]$ 를 구하는 것이다.

Eigenvector를 알고있을 때, eigenvalue를 구하는 문제에는 다양한 활용이 존재한다. 예를 들어, 주어진 Hamiltonian에 대해 ground state의 energy를 추정하는 상황을 가정하자. k-local Hamiltonian \hat{H} 와 ground state $|\psi_0\rangle$ 에 대해 다음이 성립한다.

$$\hat{H}|\psi_0\rangle = E_0|\psi_0\rangle$$

따라서, 우리가 E_0 를 estimate하기를 원한다는 것은 \hat{H} 에 대한 eigenvector $|\psi_0\rangle$ 이 알려져있을 때 eigenvalue를 구하는 것을 의미하며, 이는 phase estimation으로 해결할 수 있는 문제이다.¹⁵

¹⁴qubit의 순서를 바꾸기 위해 필요한 SWAP gate에 대한 gate complexity는 $O(n)$ 이므로 무시할 수 있다.

¹⁵ \hat{H} 를 unitary operator $\hat{U} = e^{-i\hat{H}t}$ 로 구성하면, \hat{U} 에서 eigenvalue는 $e^{-iE_0 t}$ 가 된다. (with matrix exponential)

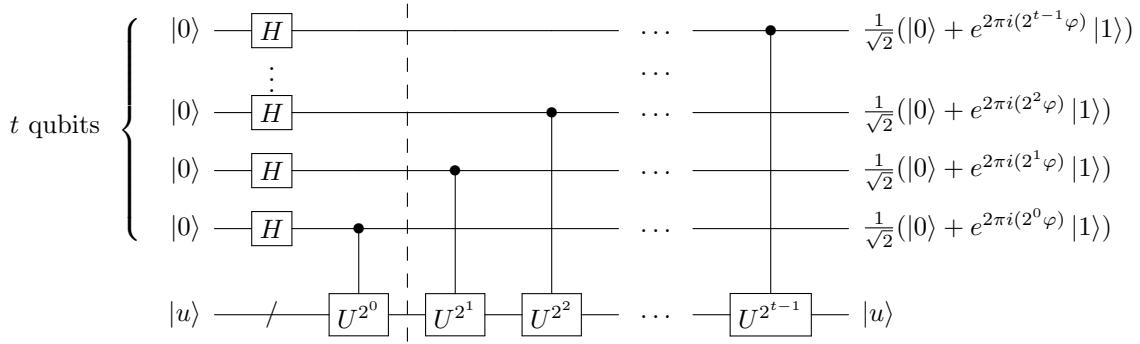


Figure 1.2: QPE circuit (first step)

Phase estimation을 설계하기 위해, 우리는 먼저 다음과 같은 oracle들을 가지고 있다고 가정한다.

- eigenvector $|u\rangle$ 를 준비시킬 수 있는 oracle
- controlled- U^{2^j} operation을 수행할 수 있는 oracle

Phase estimation algorithm은 크게 2가지 step으로 구성되는데, 첫 번째 step은 Fig. 1.2와 같이 구성되어 controlled-U operation을 차례대로 수행하는 과정이다. 두 번째 step은 첫 번째 register에 대해 inverse QFT를 수행하는 것이다. 각 step별로 어떻게 연산이 이루어지는지 따라가보자.

먼저 Fig. 1.2의 연산은 다음 단계를 따른다.

1. input state : phase estimation이 사용하는 2개의 register들을 각각 다음과 같이 초기화한다.¹⁶

$$|\psi\rangle = |0^{\otimes t}\rangle |u\rangle$$

2. apply H -gate : 첫 번째 register에 대해 H -gate를 적용하면 다음과 같은 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t} |u\rangle$$

3. apply controlled-U operation : 두 번째 register를 target으로 하고, 첫 번째 register의 각 qubit을 control로 하여 controlled-U operation을 가하는 과정을 control qubit을 바꿔가면서 수행한다.

예를 들어, t 번째 qubit을 control qubit으로 하여 controlled- U^{2^0} 를 가하면, 다음의 상태를 얻는다.¹⁷

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t-1}(|0\rangle |u\rangle + |1\rangle e^{2\pi i(2^0\varphi)} |u\rangle) \\ &= \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t-1}(|0\rangle + e^{2\pi i(2^0\varphi)} |1\rangle) |u\rangle \end{aligned}$$

일반화하여, j 번째 qubit을 target qubit으로 하여 두 번째 레지스터에 controlled- U^{t-j} gate를 가하는 과정을 t 개의 qubit에 대해서 모두 수행하면 얻게되는 최종 state는 다음과 같다.

$$\frac{1}{2^{t/2}} \left(|0\rangle + e^{2\pi i 2^{t-1}\varphi} |1\rangle \right) \left(|0\rangle + e^{2\pi i 2^{t-2}\varphi} |1\rangle \right) \dots \left(|0\rangle + e^{2\pi i 2^0\varphi} |1\rangle \right) = \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle. \quad (1.18)$$

이렇게 얻은 state는 QFT를 $|N\psi\rangle$ 에 대해서 적용한 result state와 동일하다. (put $j = N\varphi$, $N = 2^t$) 따라서, φ 의 값을 얻기 위하여 Eq. (1.18)에 inverse QFT를 수행하여 $N\varphi$ 를 구할 수 있다.

앞에서 우리는 $\varphi \in [0, 1]$ 이라고 가정하였으므로, $\varphi = 0.\varphi_1\varphi_2 \dots \varphi_t = \sum_{k=1}^t \varphi_k / 2^k$ 이라는 binary representation으로 표현할 수 있다. 따라서 inverse QFT 결과로 얻는 $N\varphi$ 는 다음과 같다.

$$N\varphi = 2^t \sum_{k=1}^t \varphi_k / 2^k = \sum_{k=1}^t \varphi_k 2^{t-k} = \varphi_1 \varphi_2 \dots \varphi_t$$

¹⁶ 첫 번째 register의 qubit 개수 t 는 QPE의 accuracy, success probability를 결정한다.

¹⁷ target qubit은 $|u\rangle$ 이지만, U^{2^0} s는 phase만 변화시키기 때문에, control qubit에 대해 phase term을 옮겨서 표현할 수 있다.

즉, 최종적으로 QPE를 진행하면 φ 의 binary representation에서 소수점 아래 자릿수를 값으로 갖는 t 개의 qubit을 얻게되며, 각 qubit의 값은 0 또는 1이므로 computational basis에서 측정하게되면 φ 의 근사값을 얻을 수 있다. \square

Lecture 12

1.5.2 Performance

28 Oct. 10:30

앞에서 우리는 phase φ 가 t 개의 소수점 아랫수로 표현될 수 있는 값이라는 가정을 했었다. 그러나, $\varphi = 0.\varphi_1\varphi_2 \dots \varphi_t\varphi_{t+1} \dots$ 처럼 $t+1$ 개 이상의 소수점 아랫자리수를 가지게 되면, QPE를 사용하여 얻은 결과 $\tilde{\varphi}$ 는 실제값의 근사치가 된다. 즉, error가 발생하게 된다.

error를 분석하기 위해서, t 개의 qubit로 나타낼 수 있는 이진수 표현중에서 가장 φ 와 가까운 φ 보다 크지 않은 수를 만드는 정수를 b 라고 하자. ($b/2^t = 0.b_1 \dots b_2$)

$$b = \arg \min_{0 \leq b \leq 2^t - 1} \varphi - b/2^t, \quad (b/2^t \leq \varphi) \quad (1.19)$$

우리는 이제 QPE의 측정 결과가 b 와 가까우며, 따라서 φ 를 높은 확률로 추정할 수 있음을 보이고자한다. Eq. (1.18)에 IQFT를 적용한 결과는 다음과 같다.

$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle \xrightarrow{\text{IQFT}} \frac{1}{2^t} \sum_{k,l=0}^{2^t-1} e^{2\pi i \varphi k} \underbrace{e^{-2\pi i l k / 2^t}}_{|k\rangle} |l\rangle = \frac{1}{2^t} \sum_{k,l=0}^{2^t-1} \left(e^{2\pi i (\varphi - l/2^t)} \right)^k |l\rangle.$$

만약 $\varphi = l/2^t$ 라면, $e^{2\pi i (\varphi - l/2^t)} = 1$ 이 2^t 개의 항에 대해서 더해지기 때문에, $|l\rangle$ 의 amplitude가 1이되어서 100% 확률로 $|l\rangle = |N\varphi\rangle$ 를 관측한다. 그러나 φ 가 $l/2^t$ 와 유사하다면, 다른 항들도 관측될 확률을 가질 수 있고 이것이 바로 error를 발생시키는 원인이 된다.

Definition 1.5.1 (Inverse Quantum Fourier transform). Inverse QFT는 fourier basis $|k\rangle$ 에 대하여, 다음 변환을 수행한다.^a

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

^aphase를 소거하기 위해 phase가 음수이다.

다른 항이 관측될 확률을 분석하기 위해서, α_l 을 $|(b+l) \bmod 2^t\rangle$ state에 대한 amplitude라고 하자. 등비급수 공식을 사용하여 나타내면 다음과 같다. ($\delta = \varphi - b/2^t$) (See Eq. (1.19))

$$\alpha_l \triangleq \frac{1}{2^t} \sum_{k=0}^{2^t-1} \left(e^{2\pi i (\varphi - (b+l)/2^t)} \right)^k = \frac{1}{2^t} \left(\frac{1 - e^{2\pi i (2^t \varphi - (b+l))}}{1 - e^{2\pi i (\varphi - (b+l)/2^t)}} \right) = \frac{1}{2^t} \left(\frac{1 - e^{2\pi i (2^t \delta - l)}}{1 - e^{2\pi i (\delta - l/2^t)}} \right)$$

이때, a_l 은 $b + l \bmod 2$ 연산에 의해 결정되기 때문에, 서로 다른 l 에 대해서 동일한 α_l 을 정의할 수 없도록 l 은 $-2^{t-1} < l \leq 2^{t-1}$ 범위로 제한된다.

이제 QPE의 측정 결과가 m 이라고 가정하자. 우리는 m 과 b 의 차이 $|m - b|$ 가 β 보다 클 확률을 계산하고자한다. 이는 우리가 허용할 수 있는 error β 보다 더 큰 error가 발생할 failure probability를 의미한다.

$$\Pr(|m - b| > \beta) = \sum_{\beta \leq |l|} |\alpha_l|^2 = \sum_{-2^{t-1} < l \leq -(\beta+1)} |\alpha_l|^2 + \sum_{\beta+1 \leq l \leq 2^{t-1}} |\alpha_l|^2 \quad (1.20)$$

이때, $|1 - e^{i\theta}|$ 에 대한 다음의 2가지 bound를 적용하여 확률을 계산할 수 있다.

- $|1 - e^{i\theta}| \geq \frac{2|\theta|}{\pi}$, for $(-\pi \leq \theta \leq \pi)$
- $|1 - e^{i\theta}| \leq 2$, for any $\theta \in \mathbb{R}$.

l 의 범위에 대한 제한 덕분에, $-\pi \leq 2\pi(\delta - l/2^t) \leq \pi$ 조건 만족하여 $|\alpha_l|$ 의 upper bound를 구할 수 있다.

$$|\alpha_l| = \left| \frac{1}{2^t} \left(\frac{1 - e^{2\pi i (2^t \delta - l)}}{1 - e^{2\pi i (\delta - l/2^t)}} \right) \right| \leq \frac{2}{2^t |1 - e^{2\pi i (\delta - l/2^t)}|} \leq \frac{1}{2^{t+1} (\delta - l/2^t)} = \frac{1}{2} \left(\frac{1}{2^t \delta - l} \right) \quad (1.21)$$

따라서 Eq. (1.20)에 Eq. (1.21)을 대입하면, 다음의 upper bound를 얻는다.

$$\Pr(|m - b| > \beta) \leq \frac{1}{4} \left[\sum_{-2^{t-1} < l \leq -(\beta+1)} \frac{1}{(l - 2^t \delta)^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l - 2^t \delta)^2} \right] \quad (1.22)$$

$0 \leq 2^t \delta \leq 1$ 을 이용하여 간략히 표현하면, 다음을 얻는다.

- Eq. (1.23) : upper bound를 구하기 위해서 각 항의 값이 최대가 되도록 하는 $2^t \delta$ 를 대입한다.¹⁸
- Eq. (1.24) : 첫 번째 항을 보면 l^2 에 대해서 연산하고 있기 때문에 l 의 범위를 음수에서 양수로 바꾸어도 된다.
- Eq. (1.25) : 두 항의 하나로 나타낸 뒤¹⁹, 합을 적분으로 근사한다.

따라서 정리하자면, QPE의 failure probability는 β 에 대해 $\frac{1}{2(\beta-1)}$ 로 bound 된다.

$$\Pr(|m - b| > \beta) \leq \frac{1}{4} \left[\sum_{-2^{t-1} < l \leq -(\beta+1)} \frac{1}{l^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l-1)^2} \right] \quad (1.23)$$

$$= \frac{1}{4} \left[\sum_{\beta+1 \leq l < 2^{t-1}} \frac{1}{l^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l-1)^2} \right] \quad (1.24)$$

$$\leq \frac{1}{2} \sum_{l=\beta}^{2^{t-1}-1} \frac{1}{l^2} \leq \frac{1}{2} \int_{\beta-1}^{2^{t-1}-1} \frac{1}{l^2} dl \leq \frac{1}{2(\beta-1)}. \quad (1.25)$$

만약 우리가 φ 를 정확도 2^{-n} 으로 추정하기를 원한다면,²⁰ $\beta = 2^{t-n} - 1$ 으로 선택한다. 이때, t 는 QPE에 사용하는 qubit의 개수이다. ($p = t - n$) 그럼 이 β 에 대해, failure probability는 $1/2(2^{t-n} - 2) = 1/2(2^p - 2)$ 로 bound되기 때문에, QPE의 정확도는 적어도 $1 - 1/(2(2^p - 1))$ 이상이다.

따라서, n bits로 표현된 φ 를 success probability가 $1 - \delta$ 이상이 되도록 QPE를 수행하기 위해서는, 첫 번째 레지스터의 개수 t 를 다음과 같이 결정해야한다.

$$t = n + \left\lceil \log \left(2 + \frac{1}{2\delta} \right) \right\rceil$$

또한, 앞에서 우리는 eigenvector의 state $|u\rangle$ 로 준비할 수 있는 oracle을 가정했다. 그러나, 실제 상황에서 는 이것이 불가능할 수도 있다. 만약 우리가 다른 state $|\psi\rangle$ 를 $|u\rangle$ 대신에 사용했다고 하자. $|\psi\rangle$ 는 eigenbasis에 대해 다음과 같은 linear combination으로 표현된다.

$$|\psi\rangle = \sum_u c_u |u\rangle$$

각 eigenstate의 eigenvalue가 $e^{2\pi i \varphi_u}$ 라면, $|\psi\rangle$ 에 대해 phase estimation을 수행한 결과는 다음과 같이 각각의 eigenstate에 대해 phase estimation을 수행한 결과들의 superposition state가 될 것이다.

$$|0^{\otimes n}\rangle |\psi\rangle \xrightarrow{\text{QPE}} \sum_u c_u |\varphi_u\rangle |u\rangle,$$

따라서 이러한 output state를 측정하게 되면, probability $|c_u|^2$ 에 따라서 랜덤하게 어떤 eigenstate $|u\rangle$ 에 대한 phase $|\varphi_u\rangle$ 를 얻게된다. 즉, 우리가 실제로 얻고자하는 eigenstate $|u\rangle$ 에 대한 amplitude c_u 가 충분히 크다면, QPE를 여러번 반복하여 우리가 원하는 eigenvalue를 얻을 수 있다.

마지막으로, QPE의 resource requirement를 분석해보자.

- (space complexity) accuracy ϵ 을 달성할 probability가 $1 - \delta$ 가 되도록 필요한 qubit의 개수 :

$$t = \log(1/\epsilon) + \left\lceil \log \left(2 + \frac{1}{2\delta} \right) \right\rceil = O \left(\log \frac{1}{\delta\epsilon} \right).$$

¹⁸ 첫 번째 항은 $|l\rangle$ 음수이기 때문에 $2^t \delta$ 가 0이 되도록하고, 두 번째 항은 $|l\rangle$ 양수이기 때문에 $2^t \delta$ 가 1이 되도록한다.

¹⁹ using $\frac{1}{l^2} + \frac{1}{(l-1)^2} \leq \frac{2}{l^2}$.

²⁰ n-bit 이진수에서 n번째 자릿수 이하에서 발생하는 차이값

- (time complexity)

- (step 1) QPE를 실행하기 위해서는 controlled unitary를 2^t 번 가해야하므로, gate complexity를 qubit의 개수 t 에 대해서 표현하면 $O(1/(\delta\epsilon))$ 이 된다.
- (step 2) t 개의 qubit에 대해 IQFT를 가하기 때문에 IQFT의 complexity는 $\Theta(t^2)$ 이 된다.

따라서 QPE의 total time complexity는 다음과 같다.

$$O\left(\frac{1}{\delta\epsilon}\right) + \Theta\left(\left(\log \frac{1}{\delta\epsilon}\right)^2\right)$$

Lecture 13

1.6 Applications of phase estimation

30 Oct. 10:30

1.6.1 Ground state energy estimation

앞에서 phase estimation의 대표적인 예시로 사용했던 ground state energy estimation에 대해 조금 더 자세히 살펴보자. H 가 다음의 eigendecomposition을 가지는 Hamiltonian operator라고 하자.²¹

$$H |\psi_j\rangle = \lambda_j |\psi_j\rangle, \quad (\text{suppose } 0 < \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1} < \frac{1}{2})$$

우리가 ground state $|\psi_0\rangle$ 의 정확한 state를 알고 있다면, QPE를 적용해서 높은 확률로 ground state energy를 추정할 수 있다. 반면, 다음과 같은 eigenstate의 approximation state를 이용한다고 하자.

$$|\psi\rangle = \sum_{k \in \{0, \dots, N-1\}} c_k |\psi_k\rangle$$

이 state를 관측했을 때, ground state를 얻을 확률은 $p_0 = |\langle\psi|\psi_0\rangle|^2$ 이고, p_0 이 충분히 크다면 QPE를 여러번 반복하여 ground state energy를 추정할 수 있다. (Poly)

Ground state energy estimation은 ϵ 의 정확도로 ground state energy λ_0 를 추정하고자 한다. ($\epsilon < \lambda_0$) 이 문제는 quantum many-body physics, quantum chemistry, optimization 등 많은 분야에서 중요한 문제이다. 일반적으로, QPE를 활용하기 위해서 Hamiltonian에 대한 unitary operation을 이용할 수 있다고 가정한다.

$$U = e^{2\pi i H}$$

이 unitary에 대해 ground state energy는 phase에 위치하게 되며, 따라서 QPE를 적용할 수 있다.

$$U |\psi_0\rangle = e^{2\pi i \lambda_0} |\psi_0\rangle.$$

1.6.2 Order-finding algorithm

Prerequisite: Uncomputation

먼저, order-finding algorithm을 설명하기 전에 uncomputation technique에 대해 먼저 이야기해보자. Classical circuit f 에 대응되는 quantum circuit O_f 를 구성하는 과정을 *uncomputation*이라고 한다.²²

$$O_f |x, b\rangle = |x, b \oplus f(x)\rangle$$

구체적으로 uncomputation 과정을 어떻게 수행하는지 단계별로 살펴보자.

1. classical circuit을 *reversible*하도록 n 개의 ancilla qubit을 추가한다. ($g(x)$ 는 임의의 garbage 값)

$$\tilde{O}_f |x\rangle |0^n\rangle |b\rangle \rightarrow |g(x)\rangle |f(x)\rangle |b\rangle$$

2. $|b\rangle$ 와 ancilla qubit에 대해 controlled-X gate를 적용한다.

$$|g(x)\rangle C(X) |f(x)\rangle |b\rangle \rightarrow |g(x)\rangle |f(x)\rangle |b \oplus f(x)\rangle$$

3. O_f^\dagger 를 적용하고 ancilla qubit을 무시하면, $O_f = \tilde{O}_f^\dagger C(X) \tilde{O}_f$ 는 우리가 원하는 연산을 수행한다.

$$\tilde{O}_f^{-\dagger} |g(x)\rangle |f(x)\rangle |b \oplus f(x)\rangle \rightarrow |x\rangle \underbrace{|0^n\rangle}_{\text{ancilla}} |b \oplus f(x)\rangle$$

²¹다음 가정은 non-degenerate case; 중복된 고윳값이 없는 상태를 다룬다는 것을 의미한다.

²²Deutsch's algorithm에서 사용했던 oracle과 동일한 개념이다.

Order-finding

이제 order-finding problem이 무엇인지 소개하려고 한다. order-finding problem은 다음과 같이 표현된다. ²³

- input: integer (x, N) , $(x < N, \gcd(x, N) = 1)$
- output: the **order** of $x \bmod N$, r .

Definition 1.6.1 (Order). 다음을 만족하는 가장 작은 양의 정수 r 를 *order of x modulo N* 이라고 한다.

$$x^r \equiv 1 \pmod{N}$$

Classical algorithm으로 이 문제를 input의 크기 L 에 대하여 polynomial time에 해결할 수 있는 방법은 현재까지 없다고 알려져 있다. 이때 L 은 N 과 N 보다 작은 수 x 를 표현하는데 필요한 bit의 개수를 의미하기 때문에 $L \triangleq \lceil \log N \rceil$ 이다.

Quantum algorithm으로 이 문제를 해결하기 위해서 우리는 *phase estimation*을 활용한다. Phase estimation을 적용하기 위해서는 unitary operator U 와 그에 대응되는 eigenvector $|u\rangle$, eigenvalue φ 의 형태로 문제를 정의해야 한다. 따라서 이를 위해 computational basis $|y\rangle \in \{|0\rangle, \dots, |2^L - 1\rangle\}$ 에 대하여 다음과 같이 동작하는 unitary를 정의하자.

$$U|y\rangle \triangleq \begin{cases} |xy \bmod N\rangle & 0 \leq y < N, \\ |y\rangle & N \leq y < 2^L \end{cases}$$

이 unitary는 N 보다 큰 basis들의 subspace에 대해서는 identity matrix처럼 행동하지만, 그보다 작은 subspace에 대해서는 주어진 input x 에 대해 $xy \bmod N$ 연산 결과가 되도록 만든다. 이렇게 정의한 unitary가 정말로 valid한지를 확인하기 위해 subspace에 대해 작용하는 unitary $|U\rangle$ 가 valid한지를 확인해보자.

$$\langle y|\tilde{U}^\dagger \tilde{U}|y'\rangle = \langle xy \bmod N|xy' \bmod N\rangle = \langle y|y'\rangle$$

연산 전후 norm을 보존하기 때문에 우리가 정의한 unitary는 valid하다. ²³ □

이제 다음과 같이 정의한 state가 U 의 eigenvector임을 보인다면, phase estimation을 통해 어떤 임의의 eigenvector에 대해서도 항상 eigenvalue r 을 얻을 수 있다는 사실을 입증할 수 있다.

$$|u_s\rangle \triangleq \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \bmod N\rangle$$

어떤 정수 $0 \leq s \leq r-1$ 에 대해서도 $|u_s\rangle$ 가 eigenvector임을 보이기 위해 definition을 이용하자.

- Eq. (1.26) : $x^k \bmod N$ 은 modulo 연산 때문에 항상 그 결과가 $\{0, \dots, N-1\}$ 범위에 존재하여 unitary operation이 $|x^{k+1} \bmod N\rangle$ 으로 state를 변화시킨다.
- Eq. (1.27) : $k = k+1$ 로 재정의하면, sigma의 적용 범위가 $[0, r-1]$ 에서 $[1, r]$ 로 변화하지만, modulo N 연산 때문에 연산 범위를 동일하게 생각할 수 있다.
- Eq. (1.29) : 따라서 U 를 $|u_s\rangle$ 에 적용한 결과는 단순히 $|u_s\rangle$ 에 phase $e^{2\pi i s / r}$ 을 가한 결과와 동일하므로 $|u_s\rangle$ 는 U 의 eigenvector이다. □

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^{k+1} \bmod N\rangle \quad (1.26)$$

$$= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s(k-1)}{r}\right) |x^k \bmod N\rangle \quad (1.27)$$

$$= \frac{1}{\sqrt{r}} \exp\left(\frac{2\pi i s}{r}\right) \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \bmod N\rangle \quad (1.28)$$

$$= \exp\left(\frac{2\pi i s}{r}\right) |u_s\rangle. \quad (1.29)$$

따라서, 어떤 s 에 대해서도 phase estimation을 적용하면 s/r 을 얻을 수 있다.

²³ 또는 $\gcd(x, N) = 1$ 이라면, modulo N 에서 x 의 역원은 항상 존재한다는 성질을 활용해서 증명할 수도 있다.

단, phase estimation을 수행하기 위해서는 아직 2가지의 요소가 더 필요하다.

- 다음 연산을 구현하는 방법 (t 개의 qubit를 이용할 때)

$$|z\rangle U^{z_t 2^{t-1}} \cdots U^{z_1 2^0} |y\rangle = |z\rangle |x^z y \mod N\rangle.$$

- U 의 eigenvector $|u_s\rangle$ 를 ²⁴ 없이 준비하는 방법.

(1) 의 구현은 앞에서 설명한 *uncomputation* 기법을 활용하여 생각할 수 있다. 다음 gate는 $f(z) = x^z \mod N$ 을 수행하는 classical function을 quantum circuit으로 구현한 것이다.²⁵

$$|z\rangle |y\rangle |0\rangle \xrightarrow{\tilde{O}_f} |z\rangle |y\rangle |x^z \mod N\rangle \quad (1.30)$$

만약 $f(z) = x^z \mod N$ 을 계산하는 classical circuit이 있다면, uncomputation을 사용하여 Eq. (1.30)과 같은 연산을 quantum computer에서도 수행할 수 있다. *modular exponentiation*²⁶에 의하여, $f(z) = x^z \mod N$ 은 다음과 같이 계산할 수 있다. ($z = z_t z_{t-1} \cdots z_1$)

$$x^z \mod N = (x^{z_t 2^{t-1}} \mod N)(x^{z_{t-1} 2^{t-2}} \mod N) \cdots (x^{z_1 2^0} \mod N)$$

따라서 이 아이디어를 이용하여 uncomputation 과정을 적용하면 다음의 circuit을 만들 수 있다.

1. *modular exponentiation*을 사용하면, Eq. (1.30)을 다음과 같이 계산할 수 있다.

$$|z\rangle |y\rangle |0\rangle \xrightarrow{\tilde{O}_f} |z\rangle |y\rangle |(x^{z_t 2^{t-1}} \mod N)(x^{z_{t-1} 2^{t-2}} \mod N) \cdots (x^{z_1 2^0} \mod N)\rangle$$

2. $|q_3\rangle$ 과 ancilla qubit $|q_2\rangle$ 을 곱한다. ($|q_1 q_2 q_3\rangle$)

$$|z\rangle |y\rangle |x^z \mod N\rangle \rightarrow |z\rangle |x^z y \mod N\rangle |x^z \mod N\rangle$$

3. inverse operation \tilde{O}_f^\dagger 를 취한다.

$$|z\rangle |x^z y \mod N\rangle |x^z \mod N\rangle \xrightarrow{\tilde{O}_f^\dagger} |z\rangle |x^z y \mod N\rangle |0\rangle$$

$|q_3\rangle$ 을 무시하면, $O_f = \tilde{O}_f^\dagger U \tilde{O}_f$ 는 우리가 원하는 연산을 수행한다. 만약 $|x^z y \mod N\rangle$ 을 나타내기 위해서 사용하는 qubit의 개수가 L 개라면, 이 연산의 time complexity는 $O(L^3)$ 이다.

- QPE에 사용하는 qubit의 개수 t 는 다음과 같이 설정된다.²⁷

$$t = 2L + 1 + \lceil \log(2 + 1/(2\delta)) \rceil = O(L)$$

- $0 \leq j < t$ 범위의 모든 j 에 대해 이를 계산해야하므로, $O(t)$ 번 연산을 수행한다.

- 매 단계마다, $x^z y \mod N$ 값을 갱신하기 위해서 이전 단계까지의 누적 계산 결과 $x^{z_1 2^0} \cdots x^{z_{t-1} 2^{t-2}}$ 에 이번 단계에 얻은 결과인 $x^{z_t 2^0}$ 를 곱한 뒤 $\mod N$ 을 취한다.

– L bit 수의 곱셈 연산의 complexity: $O(L^2)$

– $\mod N$ 연산의 complexity: $O(L^2)$

반면, (2) 를 구현하는 것은 조금 더 까다롭다. 이 문제를 해결하기 위해서 $|u_s\rangle$ 들의 superposition state를 대신 입력으로 제공하는 방법을 사용할 수 있다. 만약 r 개의 qubit를 이용하여 $|u_s\rangle$ 를 준비한다고 가정하면, superposition state는 다음과 같이 정리할 수 있다.²⁸

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = \frac{1}{r} \sum_{s,k=0}^{r-1} \exp\left[-\frac{2\pi i s k}{r}\right] |x^k \mod N\rangle = \sum_{k=0}^{r-1} \delta_{k0} |x^k \mod N\rangle = |1\rangle.$$

따라서 $|1\rangle$ 을 입력으로 사용하면, 확률에 따라서 임의의 s 에 대해 $\varphi \approx s/r$ 를 적어도 $(1 - \delta)/r$ 확률로 $2L + 1$ bit 정확도를 달성할 수 있다.

²⁴eigenvector의 정의에 이미 우리가 얻고자하는 값인 r 이 들어있다

²⁵ $0 \oplus f(z)$

²⁶ $ab \mod N = (a \mod N)(b \mod N)$

²⁷우리가 조절할 수 있는 parameter인 L 에 의해, QPE의 accuracy가 조절된다.

²⁸complex exponential의 orthogonality에 의하여, $\sum_{s=0}^{r-1} e^{(2\pi i s/r)(k-0)} = \begin{cases} 0, & \text{for } k \neq 0 \\ r, & \text{for } k = 0 \end{cases}$

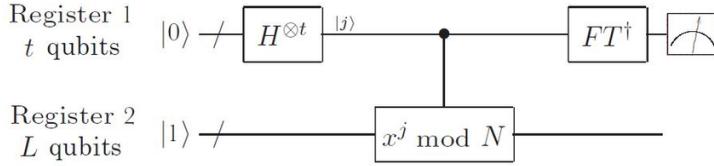


Figure 1.3: Order-finding algorithm [Dha15]

The continued fraction expansion

우리가 앞의 알고리즘을 통해서 얻은 결과는 s/r 이다. 따라서 이 결과로부터 r 의 값을 얻어내는 추가적인 후처리가 필요하다. 우리가 얻은 $\varphi \approx s/r$ 은 임의의 $2L+1$ bit 소수이므로 임의의 실수이지만, 실제로 우리가 얻고자 하는 값은 s/r 은 분명히 유리수이다. 이 특징을 활용하기 위해 continued fraction이라는 개념을 도입한다. continued fraction은 주어진 실수를 연분수 형태로 표현한 것이다.²⁹

$$[a_0, \dots, a_M] \triangleq a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}$$

주어진 convergent를 연분수 형태로 변환하는 알고리즘을 *continued fraction algorithm*이라고 한다.

Theorem 1.6.1. s/r 이 유리수이고, φ 가 QPE로 얻은 근사값일 때 두 값의 차이가 다음과 같은 upper bound를 만족한다면, continued fraction algorithm을 적용하여 얻은 결과가 바로 s/r 이다.

$$\left| \frac{s}{r} - \varphi \right| \leq \frac{1}{2r^2}$$

Order-finding problem에서는 $(1-\delta)/r$ 의 높은 확률로 $2L+1$ bit 정확도를 달성하기에, 다음을 만족한다.

$$\left| \frac{s}{r} - \varphi \right| \leq \frac{1}{2^{2L+1}} \leq \frac{1}{2r^2}$$

따라서 continued fraction algorithm을 적용할 수 있으므로, $O(L^3)$ 의 time complexity로 s/r 을 구할 수 있고 $\gcd(s, r) = 1$ ³⁰이라면 r 을 구할 수 있다.

Performance

Order-finding algorithm의 정확도를 분석하기 위해서, 이 알고리즘이 실패하는 경우를 분석해보자.

- QPE가 실패하는 경우 : 아주 작은 확률 ϵ 로 실패할 수 있다. 이 경우 qubit의 개수를 늘려서 QPE의 정확도를 높이거나 QPE를 여러번 반복하여 각 outcome의 확률을 비교하여 해결할 수 있다.
- $\gcd(s, r) \neq 1$ 인 s 를 얻는 경우 : $0 \leq s < r - 1$ 범위에서 두 수가 서로소일 확률은 매우 높기 때문에, 알고리즘을 $2 \log N$ 번 반복하면 이 문제를 해결할 수 있다.

마지막으로 Order-finding algorithm의 circuit이 필요로하는 자원을 분석해보자.

- $O(L)$: Register 1에 H 변환을 가하는데 필요한 circuit의 개수
- $O(L^2)$: IQFT를 적용하는데 필요한 circuit의 개수
- $O(L^3)$: Controlled unitary; modular exponentiation에 필요한 circuit의 개수
- $O(L^3)$: continued fraction algorithm이 사용하는 time complexity
- $O(\log N)$: 오류가 발생했을 때, 다시 알고리즘을 실행하는 횟수

²⁹이때 $[a_0, \dots, a_M]$ 을 M th convergent라고 한다.

³⁰만약 두 수 사이에 공약수가 존재하면, continued fraction algorithm이 반환하는 값이 통분한 값이 된다

1.6.3 Shor's algorithm: factoring

이제 마지막으로 그 유명한 Shor's algorithm에 소개하고자 한다. Classical computer에서 polynomial time에 해결하는 효과적인 알고리즘이 아직 존재하지 않는 *factoring*을 quantum computer를 사용하여 효율적으로 수행할 수 있다는 사실을 보여주는 중요한 알고리즘이다.

Prime factorization problem은 다음과 같이 정의된다.

- **input:** natural number N
- **output:** list of prime numbers $\{p_1, p_2, \dots, p_k\}$ such that $N = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_k^{e_k}$

이를 해결하는 Shor's algorithm은 다음과 같다.

1. **if** $N \mid 2$, **return** 2
2. **if** $N \mid p$ 에 대해서 $N = p^k$ 라면, **return** p ³¹
3. $1 \leq x \leq N - 1$ 범위에서 랜덤하게 x 를 선택한다.

- (a) **if** $\text{gcd}(N, x) > 1$ ³², **return** $\text{gcd}(N, x)$

- (b) **else**, **order-finding algorithm**을 사용하여 order r 을 찾는다.

4. **if** $r \mid 1$, **goto** step 3.

5. **else**,

- (a) 다음을 계산한다.

$$x^r - 1 \equiv (x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \pmod{N}$$

- (b) 계산 결과로부터 얻은 2개의 항에 대해, 다음의 조건들을 확인한다.

- i. **if** $N \mid x^{r/2} - 1$, **error**³³

- ii. **elif** $N \mid x^{r/2} + 1$, **goto** step 3.

- iii. **neither**, $N \mid (x^{r/2} - 1)(x^{r/2} + 1)$ 을 동시에 나눌 수 없으므로 각각의 항이 N 의 인수를 포함하고 있다는 의미이다.

return $\boxed{\text{gcd}(N, x^{r/2} + 1), \text{gcd}(N, x^{r/2} - 1)}$

Shor's algorithm이 성공하기 위해서는 order-finding algorithm으로 얻은 r 이 짹수여야하며, **neither** case에 포함되어야 한다. 정수론의 개념들을 도입하면, 성공확률은 $1/2$ 이상임을 증명할 수 있으며, 여기에서 증명을 생략한다.

Lecture 14

1.7 Applications of the QFT

4 Nov. 10:30

이번 section에서는 QFT를 활용하는 문제들에 대해서 소개하고자 한다. QFT를 활용하여 효과적으로 해결할 수 있는 문제들은 모두 *Hidden subgroup problem*으로 일반화할 수 있다.

1.7.1 Period-finding

Hidden subgroup problem을 소개하기 전에, 먼저 hidden subgroup problem의 특수한 경우들에 대해 이야기하고자 한다. 첫 번째로 소개하는 문제는 *period-finding problem*이다.

- **input:** period function f such that $f(x) = f(x + r)$ for some unknown integer r , ($0 < r < 2^L$)
- **output:** least integer $r > 0$ such that $f(x) = f(x + r)$

주어진 period function은 **uncomputation**을 사용하여 다음과 같은 unitary operator로 표현할 수 있다.

$$U_f |x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$$

³¹ 이를 판단하는 효과적인 classical algorithm이 존재

³² using Euclidean algorithm

³³ order $r \mid 1$ 이를 만족하는 가장 작은 수이기 때문에 이런 경우는 일어나지 않는다.

이 U_f 가 주어지면, 다음의 과정을 따라서 period r 을 찾을 수 있다.

1. input state : $|0\rangle|0\rangle$ 으로 state를 초기화한다.³⁴

$$|\psi\rangle = |0\rangle|0\rangle$$

2. apply H gate on 1st register

$$|\psi\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle$$

3. apply U_f : $|y\rangle = |0\rangle$ 으로 2nd register에 $f(x)$ 가 저장된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle$$

4. apply QFT on 2nd register

QFT를 사용하여 함수 $f(x)$ 를 표현하면, 다음과 같다. (See Definition. 1.4.2)

$$f(x) \rightarrow \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{2\pi i l x / r} |\hat{f}(l)\rangle, \quad |\hat{f}(l)\rangle \triangleq \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} e^{-2\pi i l x / r} |f(x)\rangle$$

따라서 이를 이용하면, state를 다음과 같이 표현할 수 있다.

$$|\psi\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x / r} |x\rangle|\hat{f}(l)\rangle$$

5. apply IQFT on 1st register³⁵

$$|\psi\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x,y=0}^{2^t-1} e^{2\pi i l x / r} e^{-2\pi i x y / 2^t} |y\rangle|\hat{f}(l)\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x,y=0}^{2^t-1} e^{(2\pi i x)(l/r - y/2^t)} |y\rangle|\hat{f}(l)\rangle \quad (1.31)$$

final state의 coefficient를 분석하기 위해 r 이 2^t 를 나누는가에 대한 cases를 분석할 수 있다.

- $r \nmid 2^t$: 일반적으로는 r 이 2^t 를 나누지 않는 경우가 더 많다.³⁶ 그러나 이 경우에도 최종 outcome이 높은 확률로 nl 이 된다.
- $r \mid 2^t$: $nr = 2^t$ 가 되게 만드는 positive integer가 존재한다. 따라서 이를 이용하면 state가 다음과 같이 단순화 된다. (Eq. (1.31)에 $r = 2^t/n$ 대입)³⁷

$$\frac{1}{2^t} \sum_{x,y=0}^{2^t-1} e^{(2\pi i x)(nl-y)} |y\rangle = \sum_{y=0}^{2^t-1} \delta_{y,nl} |y\rangle = |nl\rangle = \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |nl\rangle|\hat{f}(l)\rangle$$

따라서 1st register를 측정한 결과를 2^t 로 나누면 $nl/2^t = l/r$ 으로 r 을 구할 수 있다. \square

1.7.2 Discrete logarithm

이번에 소개하려는 문제는 *Discrete logarithm problem*; 이산 로그 문제이다. 이 문제는 다음과 같이 정의된다.

- **input:** $x \in G$, 이때 $G = \langle g \rangle$ 는 g 로부터 만들어지는 cyclic group³⁸³⁹이다.
- **output:** minimum α such that $g^\alpha = x$ (or equivalently, $\alpha = \log_g x$)

³⁴(1st register) 합수값을 저장하기 위해 $t = O(L + \log(1/\epsilon))$ 개의 qubit, (2nd register) ancilla qubit

³⁵2nd register가 x, l 간의 QFT를 이용했다면, 1st register는 l 이 아닌 다른 변수 y 에 대해 IQFT를 수행하게 된다.

³⁶ 2^t 의 약수; 2의 거듭제곱이 아닌 이상 나눌 수 없음

³⁷ $nl - y$ 에 대해, complex exponential의 orthogonality를 사용하여 정리한다.

³⁸See https://en.wikipedia.org/wiki/Cyclic_group

³⁹대표적인 cyclic group으로 modular multiplication이 있다.

다음은 다양한 cyclic group들의 discrete log에 대한 예시이다.

- (Property of logarithm) For any $G = \langle g \rangle$, $\log_g 1 = 0$,
- For $G = \mathbb{Z}_7^\times$, $\log_3 2 = 2$,
- For $G = \mathbb{Z}_{541}^\times$, $\log_{126} 282 = 101$.

이 문제를 해결하기 위해, Discrete logarithm problem을 *period finding* problem의 형태로 변환할 수 있다. 먼저 다음 integer function f 를 가정하자. 여기서 $\gcd(a, N) = 1$ 이며, modulo N 에서 a order가 r 이라고 하자.

$$f(x_1, x_2) = a^{sx_1+x_2} \bmod N$$

이렇게 정의한 함수는 어떤 l 에 대해 $(l, -ls)$ 라는 period를 가진다.

$$f(x_1 + l, x_2 - ls) = a^{sx_1+sl+x_2-ls} \bmod N \equiv a^{sx_1+x_2} \bmod N = f(x_1, x_2)$$

Uncomputation을 사용하면 이 f 에 대응되는 unitary를 구성할 수 있다.

$$U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|y \oplus f(x)\rangle$$

우리는 discrete logarithm problem을 다시 다음과 같이 표현할 수 있다.⁴⁰

- input: $b \in \mathbb{Z}_N^\times$ such that $b \equiv a^s \pmod{N}$, $a \in \mathbb{Z}$
- output: integer s

따라서 period function f 를 이용하면 문제를 해결할 수 있다.

1. input state : 3개의 register를 사용한다.

$$|\psi\rangle = |0\rangle|0\rangle|0\rangle$$

2. apply H -gate on 1st and 2nd register

$$|\psi\rangle = \frac{1}{2^t} \sum_{x_1, x_2=0}^{2^t-1} |x_1\rangle|x_2\rangle|0\rangle$$

3. apply U_f : $f(x_1, x_2)$ 를 3rd register에 저장한다.

$$|\psi\rangle = \frac{1}{2^t} \sum_{x_1, x_2=0}^{2^t-1} |x_1\rangle|x_2\rangle|f(x_1, x_2)\rangle$$

4. apply QFT on 3rd register

QFT를 사용하여 함수 $f(x_1, x_2)$ 를 표현하면 다음과 같다. (See Definition. 1.4.2)

$$|f(x_1, x_2)\rangle \rightarrow \frac{1}{r} \sum_{l_1, l_2=0}^{r-1} e^{2\pi i(l_1 x_1 + l_2 x_2)/r} |\hat{f}(l_1, l_2)\rangle$$

$$|\hat{f}(l_1, l_2)\rangle \triangleq \frac{1}{r} \sum_{x_1, x_2=0}^{r-1} e^{-2\pi i(l_1 x_1 + l_2 x_2)/r} |f(x_1, x_2)\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i l_2 j / r} |f(0, j)\rangle$$

따라서 이를 이용하면, state를 다음과 같이 표현할 수 있다.⁴¹

$$\begin{aligned} |\psi\rangle &= \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \sum_{x_1, x_2=0}^{2^t-1} e^{2\pi i(sl_2 x_1 + l_2 x_2)/r} |x_1\rangle|x_2\rangle|\hat{f}(sl_2, l_2)\rangle \\ &= \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \left[\sum_{x_1=0}^{2^t-1} e^{2\pi i(sl_2 x_1)/r} |x_1\rangle \right] \left[\sum_{x_2=0}^{2^t-1} e^{2\pi i(l_2 x_2)/r} |x_2\rangle \right] |\hat{f}(sl_2, l_2)\rangle \end{aligned}$$

⁴⁰ $a \nmid g$, $s \nmid \alpha$ 에 대응된다.

⁴¹ put $l_1 = l_2, l_2 = sl_2$

5. apply IQFT on 1st and 2nd register

$$|\psi\rangle = \frac{1}{\sqrt{r}} \sum_{l_2=0}^{r-1} |sl_2/r\rangle |l_2/r\rangle |\hat{f}(sl_2, l_2)\rangle$$

따라서 1st, 2nd register를 측정하면 각각 sl/r , 그리고 l/r 이라는 값을 얻게 되는데, $sl/r \times r/l = s$ 이므로 s 를 구할 수 있다. \square

1.7.3 Hidden subgroup problem

이제 Hidden subgroup problem에 대해 이야기할 시간이 찾아왔다.⁴²

Definition 1.7.1 (Hidden subgroup problem). Let f be a function from finitely generated group G to some finite set X . s.t. f is constant on the cosets of a subgroup K , and distinct on each coset.^{a b}

$$f : G \rightarrow X \quad \longrightarrow \quad U_f |g\rangle |h\rangle = |g\rangle |h \oplus f(g)\rangle \quad (\text{for } g \in G, h \in X)$$

Find a generating set for K .

^a K 가 주어졌을 때, coset은 G 의 원소에 대해서 $gK = \{gk : k \in K\}$ 로 정의된다.

^b같은 coset이라면 f 의 값은 매우 동일하지만, 서로 다른 coset에 속한다면 f 의 값은 서로 다르다.

앞에서 우리가 다루었던 order-finding problem, period-finding, discrete logarithm과 같은 문제들이 Hidden subgroup problem의 특수한 예시이다. 예를 들어, order-finding problem이 왜 Hidden subgroup problem으로 표현될 수 있는지 알아보자. Hidden subgroup problem에서 각각을 다음과 같이 정의하자.

- $G = \mathbb{Z}$
- $K = r\mathbb{Z}$
- (coset) $K_g = \{rg\}$ for $g \in \mathbb{Z}$ (e.g., $K_1 = \{\dots, -2, -1, 0, \dots, 1, 2\}$, $K_3 = \{\dots, -6, -3, 0, \dots, 3, 6\}$)
- $f(k) = a^k \pmod N$

이때, (1) $f(k) = f(k+r)$ 의 주기를 가지고 (2) $0 \leq k \leq r-1$ 범위에서 각각의 $f(k)$ 의 값을 모두 구분된다. 이 두 가지 성질은 f 가 동일한 coset에 속하는 원소들에 대해서 constant하고, 서로 다른 coset에 속한 원소들에 대해서는 distinct한 값을 가지는 것을 의미한다. 따라서 order-finding problem은 Hidden subgroup problem의 특수한 경우로 생각할 수 있다.

G 가 finite abelian group⁴³이라면, QFT를 적용하여 Hidden subgroup problem을 해결할 수 있다. 그러나, non-abelian group에 대해서는 아직 이 문제를 효과적으로 푸는 방법이 알려지지 않았다. (non-abelian group에 대한 대표적인 문제가 바로 *Graph isomorphism*)

1.8 Quantum search algorithms

Quantum search algorithm에 대해서 알아보자. 정렬되어있거나 특정한 규칙이 있는 데이터에 대한 search algorithm은 classical computer에서도 이미 충분히 효과적으로 수행할 수 있다. 그러나, 여기에서 다룰 search problem은 unstructured한 데이터를 다룬다. 이를 위해 먼저 하나의 oracle을 가정하자.

N 개의 항목이 있는 unstructured 데이터에 대해서, $f(x)$ 는 x 번째 항목이 우리가 찾고자 하는 항목인지 를 판단하는 oracle이다. 따라서 $f(x) = 1$ 이면, x 가 solution이 되고 $f(x) = 0$ 이면, x 가 solution이 아니다. 편의를 위해서 $N = 2^n$ 이며, 찾고자 하는 항목의 개수가 M ($1 \leq M \leq N$)이라고 가정하자.

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

이 문제를 어떻게 해결할 수 있을까? 단순하게 생각하면 랜덤한 x 를 선택하여 $f(x)$ 의 값을 확인한 뒤 x 가 solution인지 아닌지를 판단할 수 있으며, 선택한 x 가 solution일 확률은 M/N 이다. 만약 N 이 M 에 비해 exponential하게 크다면, success probability는 exponential하게 줄어들기에 이는 효율적이지 않다.

Quantum computer에서 f 에 대한 unitary는 다음과 같이 정의된다.

$$|x\rangle |q\rangle \xrightarrow{O} |x\rangle |q \oplus f(x)\rangle$$

⁴²See https://en.wikipedia.org/wiki/Hidden_subgroup_problem

⁴³교환법칙이 성립하는 군

x 는 N 개의 항목에 대한 인덱스를 나타내기 위해 n 개의 qubit으로 이루어지며, query qubit은 1개의 qubit으로 이루어진다. 즉, 만약 $f(x) = 1$; x 가 solution이라면 query qubit의 값이 bit-flip된다.

만약 $|q\rangle = |-\rangle$ 으로 설정하면, 다음과 같은 일이 벌어진다.

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{2} \right) \xrightarrow{O} \frac{|x\rangle |f(x)\rangle - |x\rangle |1 \oplus f(x)\rangle}{2} = (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{2} \right)$$

x 가 solution이라면 bit-flip이 일어나서 phase가 바뀌게 되고, solution이 아닌 경우에는 phase가 바뀌지 않는다. 따라서 이 oracle O 를 다음과 같이 단순하게 쓸 수 있다.

$$|x\rangle \xrightarrow{O} (-1)^{f(x)} |x\rangle.$$

따라서 이 oracle을 활용하면, classical computer와 동일한 randomized algorithm을 적용하여도 $O(\sqrt{N/M})$ 의 complexity를 가진다.

1.8.1 Grover operator

Unstructured search problem을 $\Theta(\sqrt{N})$ 복잡도로 해결할 수 있는 Grover search algorithm을 설명하기 위해 먼저 Grover operator G 를 정의한다.⁴⁴

$$G \triangleq (2|\Psi\rangle\langle\Psi| - I)O$$

이때 $|\Psi\rangle$ 는 다음과 같이 정의되며⁴⁵, O 는 oracle이다.

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle,$$

우리는 이제 G 가 superposition of solutions state $|\beta\rangle$, superposition of non-solutions ketalpha⁴⁶ span하는 two-dimensional space에서 회전 연산을 수행한다는 것을 보일 것이다.

$$|\alpha\rangle \triangleq \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle, \quad |\beta\rangle \triangleq \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle. \quad (1.32)$$

$|\Psi\rangle$ 를 다음과 같이 $|\alpha\rangle, |\beta\rangle$ 의 linear combination으로 표현할 수 있기 때문에, $|\Psi\rangle$ 는 $\text{span}\{|\alpha\rangle, |\beta\rangle\}$ vector space에 속한다.

$$|\Psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle$$

G 가 수행하는 연산을 이해하기 위해서, 각 단계마다 gate가 수행하는 역할을 분석해보자.

- Oracle O 가 $\text{span}\{|\alpha\rangle, |\beta\rangle\}$ 에 속하는 임의의 vector에 대해 다음과 같이 동작한다.

$$O(a|\alpha\rangle + b|\beta\rangle) = a|\alpha\rangle - b|\beta\rangle$$

$\Rightarrow |\alpha\rangle$ 축에 대한 reflection

- ($2|\Psi\rangle\langle\Psi| - I$) 연산은 eigenvalue ± 1 에 대해 spectral decomposition을 할 수 있다.

$$2|\Psi\rangle\langle\Psi| - I = |\Psi\rangle\langle\Psi| - (I - |\Psi\rangle\langle\Psi|)$$

$|\Psi\rangle\langle\Psi|$ 는 $|\Psi\rangle$ 에 대한 projector이고 반대로 $I - |\Psi\rangle\langle\Psi|$ 는 $|\Psi\rangle$ 의 orthogonal subspace에 대한 projector이다. 따라서 이 연산은 $\{|\Psi\rangle, |\Psi\rangle^\perp\}$ vector space에서 $|\Psi\rangle$ 축에 대한 reflection을 수행하게 된다.

$\Rightarrow |\Psi\rangle$ 축에 대한 reflection

따라서 2개의 reflection operator들의 곱으로 정의되는 Grover operator는 rotation operator가 된다. 이는 $G^k|\Psi\rangle$ 를 수행하면 $\text{span}\{|\alpha\rangle, |\beta\rangle\}$ vector space에 속하는 모든 vector를 만들어 낼 수 있다는 의미이다!

좀 더 자세한 설명을 위해 $|\Psi\rangle$ 를 θ 를 사용하여 표현하자.⁴⁶

$$|\Psi\rangle = \cos \frac{\theta}{2} |\alpha\rangle + \sin \frac{\theta}{2} |\beta\rangle$$

⁴⁴ $G = (H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n})O$

⁴⁵ N 까지의 모든 가능한 입력들의 superposition

⁴⁶ $(\sqrt{(N-M)/N})^2 + (\sqrt{M/nN})^2 = 1$ 을 이용하면 cos, sin function을 이용할 수 있다.

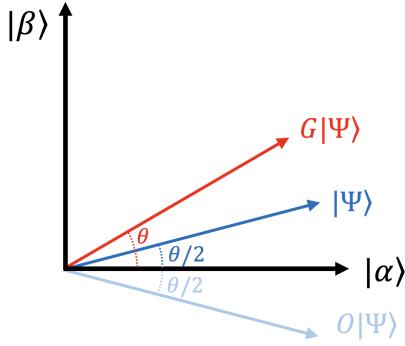


Figure 1.4: Grover operator

그럼, G 가 $|\Psi\rangle$ 에 대해 가해지면, state는 다음과 같이 변한다.

\Rightarrow 즉, G 는 $|\Psi\rangle$ 를 θ 만큼 rotation 시킨다.

$$G|\Psi\rangle = \cos \frac{3\theta}{2} |\alpha\rangle + \sin \frac{3\theta}{2} |\beta\rangle \quad (1.33)$$

따라서 G gate는 $\{|\alpha\rangle, |\beta\rangle\}$ basis에 대해 다음과 같은 matrix notation으로 표현할 수 있다.

$$G = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

G 를 k 번 적용하면, 다음 상태를 얻게된다.

$$G^k|\Psi\rangle = \cos \left(\frac{2k+1}{2}\theta \right) |\alpha\rangle + \sin \left(\frac{2k+1}{2}\theta \right) |\beta\rangle$$

만약 $(2k+1/2)\theta$ 가 $\pi/2$ 에 충분히 가깝다면, $\cos((2k+1/2)\theta)$ 가 0에 가까워지기 때문에 $G^k|\Psi\rangle$ 를 측정하면, $|\beta\rangle$ 를 얻을 확률이 높아진다. 이 아이디어를 기반으로 Grover search algorithm이 제안되었다.

1.8.2 Grover search algorithm

Grover search algorithm은 앞에서 도입한 Grover operator를 사용하여 unstructured search problem을 해결하는 알고리즘으로, 다음과 같은 과정을 따른다.

1. input state

$$|\psi\rangle = |0\rangle^{\otimes n}$$

2. apply Hadamard gate : $|\Psi\rangle$ 를 만들기 위해 Hadamard gate를 적용한다.

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

3. (Grover iteration; $G = (H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n})O$) 다음을 k 번 반복한다.

- (a) apply the oracle O
- (b) apply the Hadamard gate $H^{\otimes n}$
- (c) perform a conditional phase shift ($0\circ$ 아닌 모든 basis는 phase shift가 된다.)

$$|x\rangle \rightarrow -(-1)^{\delta_{x0}} |x\rangle$$

- (d) apply the Hadamard gate $H^{\otimes n}$

4. measure the state : 우리가 찾고자하는 M 개의 항목중에서 하나의 index를 확률에 따라 얻을 수 있다.
(See Eq. (1.32))

$$|\psi\rangle = \cos \left(\frac{2k+1}{2}\theta \right) |\alpha\rangle + \sin \left(\frac{2k+1}{2}\theta \right) |\beta\rangle \longrightarrow |\beta\rangle$$

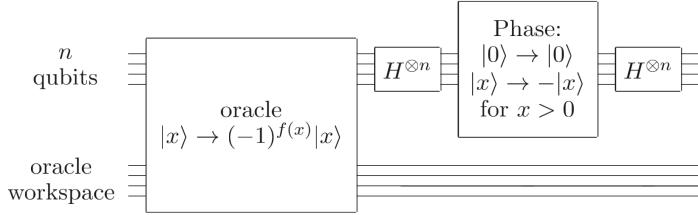


Figure 1.5: Grover operator circuit

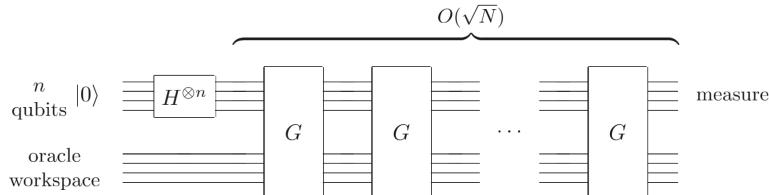


Figure 1.6: Grover search algorithm circuit

1.8.3 Performance

그렇다면, Grover algorithm을 사용하여 solution state $|\beta\rangle$ 에 다가가기 위해서는 총 몇 번의 iteration이 필요할까? Iteration을 k 번 했을 때, state는 다음과 같다.

$$G^k |\Psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right) |\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right) |\beta\rangle$$

k 를 다음과 같이 설정하면 $|\alpha\rangle$ 의 amplitude가 0이 되므로, state가 $|\beta\rangle$ 가 된다.

$$\frac{2k+1}{2}\theta = \frac{\pi}{2}$$

위 조건에 가장 가까운 k 를 선택했을 때, 얻을 수 있는 최종 state $|\psi\rangle$ 의 각도는 $\theta/2 \leq \pi/4$ 로 bound 된다. 따라서 최종 state가 β 가 될 확률은 적어도 절반이 넘는다.⁴⁷

좀 더 구체적으로 나타내면 다음과 같다.⁴⁸

$$k = \frac{\pi}{2\theta} - \frac{1}{2} = \frac{\pi}{4 \arccos \sqrt{(N-M)/N}} - \frac{1}{2} = \frac{\pi}{4} \sqrt{\frac{N}{M}} - \frac{1}{2} - O\left(\sqrt{\frac{M}{N}}\right)$$

따라서 Grover search algorithm의 complexity; k 는 $O(\sqrt{N/M})$ 이다. 이는 랜덤하게 M 을 선택하여 문제를 해결하는 방법보다 훨씬 효율적이다. ($O(N/M)$)⁴⁹

한 가지 남아있는 문제점은, k 를 선택하기 위해서는 M 의 값을 알아야한다는 것이다. 만약 문제에서 우리가 찾으려는 항목이 있는 곳의 개수 M 을 제공하지 않는다면, k 를 결정할 수 없다. 이를 해결하는 2가지 방법이 존재한다.

- k 를 랜덤하게 선택하여 탐색한다. (이 경우에도 비슷한 수준의 성능을 낼 수 있음이 알려져 있다.)
- M 을 추정하는 알고리즘을 사용한다. (by [amplitude estimation algorithm](#))

Lecture 15

1.8.4 Example: Classical circuit-SAT problem

6 Nov. 10:30

이번에는 Grover search algorithm을 이용하여 classical circuit-SAT problem을 해결하는 예시를 살펴보자.

Problem Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 을 계산하는 Poly(n) 크기의 classical circuit이 존재한다고 가정하자. 이때, $f(x) = 1$ 이 되도록 만드는 x 가 존재하는가? 존재한다면 x 는 몇 개인가?

⁴⁷ $\sin(\pi/4) = \cos(\pi/4) = 1/\sqrt{2}$

⁴⁸ $\arccos \sqrt{1-x} \approx \sqrt{x} + O(x^{3/2})$ 임을 이용한다.

⁴⁹ 성공확률이 M/N 이므로, complexity는 그 역수

랜덤하게 x 를 결정하여 $f(x)$ 의 값을 확인하는 방식은 $O(2^n)$ 이라는 complexity를 가진다. 그러나 Grover search algorithm을 사용하면, $O(2^{2/n})$ 이라는 complexity만으로도 이 문제를 해결할 수 있다.

먼저 uncomputation을 사용하여 주어진 f 에 대한 O_f 를 설계하자.

$$O_f|x, b\rangle = |x, b \oplus f(x)\rangle$$

이 oracle은 N 개의 항목중에서 M 개의 항목을 찾기 위해 각 항목의 위치가 우리가 찾기를 바라던 값인지 를 확인하는 search algorithm의 oracle과 동일하다. 따라서 Grover search algorithm을 동일한 방식으로 적용하면 주어진 문제를 쉽게 해결할 수 있다.

1.8.5 Example: Amplitude amplification

Grover search algorithm은 *amplitude amplification*을 수행하기 위해 사용할 수 있다. Amplitude amplification은 bad state $|\alpha\rangle$ 에 대한 amplitude는 감소시키고, good state $|\beta\rangle$ 에 대한 amplitude는 증가시키는 것을 목적으로 한다. 먼저 다음과 같은 unitary를 가정하자.⁵⁰

$$|\psi_0\rangle \triangleq U|0^n\rangle = \sqrt{p_0}|\beta\rangle + \sqrt{1-p_0}|\alpha\rangle = \sin \frac{\theta}{2}|\beta\rangle + \cos \frac{\theta}{2}|\alpha\rangle, \quad s.t. \langle\alpha|\beta\rangle = 0, p_0 \ll 1$$

이 문제에서는 unstructured search problem에서 요구하는 $|\alpha\rangle, |\beta\rangle$ 가 각각 특정 computational basis의 superposition state여야한다는 조건이 없기 때문에, $|\alpha\rangle$ 에 대한 reflector를 설계해야한다.

U, U^\dagger 그리고 bad state $|\alpha\rangle$ 의 orthogonal subspace에 대한 projector $I - |\alpha\rangle\langle\alpha|$ 가 주어졌을 때, 다음과 같이 initial state $|\psi_0\rangle$ 의 orthogonal subspace에 대한 projector $I - |\psi_0\rangle\langle\psi_0|$ 를 만들어낼 수 있다.

$$2|\psi_0\rangle\langle\psi_0| - I = U(2|0^n\rangle\langle 0^n| - I)U^\dagger.$$

따라서 다음 연산을 수행하면, G operator와 유사하게, $|\alpha\rangle$ 에 대한 reflection, $|\psi_0\rangle$ 에 대한 reflection 일어나서 $|b\rangle$ 에 가깝게 회전하게 된다.

$$G \triangleq (2|\psi_0\rangle\langle\psi_0| - I)(2|\alpha\rangle\langle\alpha| - I) = U(2|0^n\rangle\langle 0^n| - I)U^\dagger(2|\alpha\rangle\langle\alpha| - I)$$

1.9 Amplitude estimation algorithm (Quantum counting)

Amplitude estimation algorithm; Quantum counting algorithm은 Grover search problem에서 solution의 개수 M 을 추정하는 알고리즘이다. 이 알고리즘을 먼저 사용하게 되면, M 이 얼마인지 추정할 수 있을뿐더러 만약 주어진 데이터셋에 solution이 존재하지 않는 경우, $M = 0$ 임을 확인해줄 수 있다.

이를 이용하면, search problem의 해가 있는지 없는지를 판단하는 문제의 형태로 변환할 수 있는 NP-complete 문제를 푸는데 도움이 될 수 있다.

Grover operator G 를 $\{|\beta\rangle, |\alpha\rangle\}$ 에 대해서 matrix 표현으로 나타내면, 다음과 같다.

$$\begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$$

G 를 spectral decomposition하면 다음과 같이 나타낼 수 있고

$$G = \cos \theta |\beta\rangle\langle\beta| + \cos \theta |\alpha\rangle\langle\alpha| + \sin \theta |\beta\rangle\langle\alpha| - \sin \theta |\alpha\rangle\langle\beta|.$$

따라서 G 의 eigenvalue, eigenvector는 각각 다음과 같다.

- eigenvalue : $e^{\pm i\theta}$
- eigenvector : $|\psi_\pm\rangle = \frac{1}{\sqrt{2}}(|\beta\rangle \pm i|\alpha\rangle)$

만약 우리가 $e^{\pm i\theta}$ 에서 θ 의 값을 알아낸다면 $\sin(\theta/2) = \sqrt{M/N}$ 이므로 M 이 얼마인지 추정할 수 있다. θ 의 값을 알아내기 위해 phase estimation을 도입할 수 있다. 그런데 G 의 eigenvector를 준비하려면, $|\alpha\rangle, |\beta\rangle$ 의 상태를 알아야한다는 문제가 존재한다.

⁵⁰amplitude amplification은 사실 unstructured search problem에 대한 일반화된 형태이다.

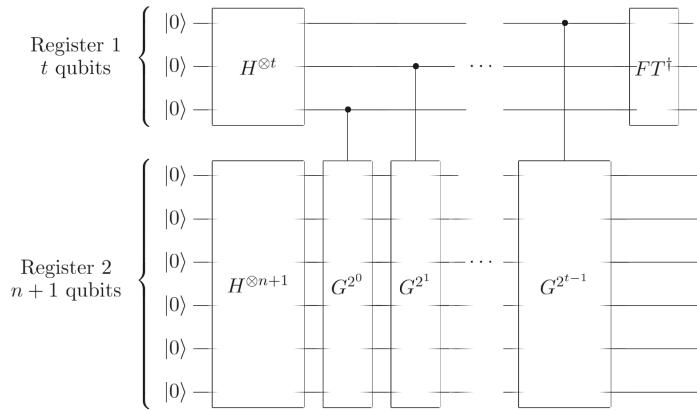


Figure 1.7: Quantum counting circuit

이를 해결하기 위해 initial state와 G 의 eigenvector가 다음의 관계를 가짐을 이용하자.

$$|\langle \psi_0 | \psi_+ \rangle|^2 = \frac{1}{2} \left| \sin \frac{\theta}{2} + i \cos \frac{\theta}{2} \right|^2 = \frac{1}{2} = |\langle \psi_0 | \psi_- \rangle|^2.$$

즉, eigenvector 대신에 initial state를 이용하면, 각각 $1/2$ 의 확률로 eigenstate 둘 중 하나가 되기 때문에 이를 이용하여 θ 를 추정할 수 있다. Fig. 1.7에서 Register 2를 $|\psi_0\rangle$ 으로 초기화하기 위해 H -gate를 사용하고 있는 것을 확인할 수 있다.

Phase estimation의 성능은 측정에 사용하는 register 1의 qubit 개수 t 에 의존한다. $t = m + \lceil \log(2 + 1/2\delta) \rceil$ 으로 선택하면, 적어도 $1 - \delta$ 의 확률로 2^{-m} accuracy를 달성할 수 있다.

$$|\Delta\theta| = |\theta - \tilde{\theta}| \leq 2^{-m}$$

그렇다면, $\tilde{\theta} \approx \theta$ 로부터 추정한 \tilde{M} 의 error $|\Delta M| = |\tilde{M} - M|$ 의 upper bound는 얼마인가? 다음의 inequality⁵¹을 Eq. (1.35)에 적용하면,

$$\left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \right| \leq \frac{|\Delta\theta|}{2}, \quad \left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) \right| \leq \sin\left(\frac{\theta}{2}\right) + \frac{|\Delta\theta|}{2}$$

$|\Delta M|/N$ 에 대해서 다음과 같이 정리할 수 있으며

$$\frac{|\Delta M|}{N} = \left| \sin^2\left(\frac{\theta + \Delta\theta}{2}\right) - \sin^2\left(\frac{\theta}{2}\right) \right| \quad (1.34)$$

$$= \left(\sin\left(\frac{\theta + \Delta\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \right) \left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \right| \quad (1.35)$$

$$\leq \left(2 \sin\left(\frac{\theta}{2}\right) + \frac{|\Delta\theta|}{2} \right) \frac{|\Delta\theta|}{2} \quad (1.36)$$

따라서 Eq. (1.36)를 $|\Delta M|$ 에 대해 정리하면, 다음의 upper bound를 얻을 수 있다.

$$|\Delta M| \leq \underbrace{\left(2N \sin\left(\frac{\theta}{2}\right) + \frac{N|\Delta\theta|}{2} \right)}_{\sqrt{M/N}} \frac{|\Delta\theta|}{2} = \left(2\sqrt{MN} + \frac{N}{2^{m+1}} \right) \frac{1}{2^{m+1}}$$

즉, $N \approx 2^{2m}$ 을 만족하도록 m 을 선택하면, $|\Delta M|$ 의 upper bound는 $O(\sqrt{M})$ 이 된다.

마지막으로 Quantum counting의 complexity를 계산해보자. Quantum counting algorithm은 t 개의 qubit에 모두 controlled- G^{2^i} gate를 가하기 때문에 $O(2^t)$ 의 complexity를 가진다. 이때, $t = m + \lceil \log(2 + 1/2\delta) \rceil$ 로 선택했기 때문에, $O(2^m)$ 이 되며, 이는 $O(\sqrt{N})$ 과 동일하다.

⁵¹ $\sin A - \sin B = 2 \cos\left(\frac{A+B}{2}\right) \sin\left(\frac{A-B}{2}\right)$

1.10 HHL (Harrow–Hassidim–Lloyd) algorithm

Problem Setup

마지막으로 다룰 알고리즘은 HHL algorithm[HHL09] 이다. HHL algorithm은 다음 linear equation의 solution x 를 구하는 quantum algorithm이다.

$$Ax = b$$

이때, A 는 nonsingular matrix이고 $A \in \mathbb{C}^{N \times N}$, $b \in \mathbb{C}^N$ 이다.⁵²

Linear algebra에 따르면 다음 관계를 만족하기 때문에, A 의 역행렬을 구해서 solution vector x 를 구할 수 있다. 그러나 문제는, A 의 역행렬을 계산하는데 exponential complexity가 요구된다는 것이다.

$$x = A^{-1}b$$

따라서 quantum computer를 사용하여 linear equation을 더 효과적으로 해결하기 위해 HHL algorithm이 고안되었다. 이를 위해서 먼저 A 가 Hermitian; $A = A^\dagger$ 이라고 가정하자. 만약, 주어진 A 가 Hermitian이 아니라면 다음과 같이 새로운 Hermitian 행렬 \tilde{A} 로 재구성하여

$$\tilde{A} = \begin{pmatrix} 0 & A^\dagger \\ A & 0 \end{pmatrix} = |1\rangle\langle 0| \otimes A + |0\rangle\langle 1| \otimes A^\dagger$$

다음과 같이 변환한 문제를 풀어서 b 를 구할 수 있기 때문에 이 가정은 일관성을 잃지 않는다.

$$\tilde{A}|0, x\rangle = |1, b\rangle$$

다음으로 b 를 quantum state vector로 표현할 수 있도록 $\|b\| = 1$ 라고 가정하자. 더 염밀하게 표현하자면, quantum state를 b 로 초기화하는 unitary U_f 가 존재한다고 하자.

$$|b\rangle = U_b |0^n\rangle$$

마지막으로, x 를 quantum state vector로 손실 없이 표현할 수 있으려면 $\|x\| = 1$ 을 만족해야한다. 그러나, b 가 unit vector라고 해서 $x = A^{-1}b$ 의 결과로 얻어지는 x 또한 unit vector일 것이라는 보장은 존재하지 않는다. 따라서 quantum version의 linear equation을 해결하는 알고리즘의 목표는, $|\hat{x}\rangle$ ⁵³와 차이가 매우 작은 state $|\tilde{x}\rangle$ 를 얻는 것이다.⁵⁴

$$\|\hat{x}\rangle - |\tilde{x}\rangle\| \leq \epsilon, \quad |\hat{x}\rangle = \frac{\hat{x}}{\|\hat{x}\|} = \frac{A^{-1}b}{\|A^{-1}b\|}.$$

알고리즘으로 얻은 output은 normalize된 solution이므로, 우리가 실제로 얻고 싶은 solution x 를 구하기 위해서는 $\|\tilde{x}\|$ 의 값까지 알고리즘이 output으로 반환해야 한다.

지금까지 설명한 내용을 정리하면, HHL algorithm의 problem은 다음과 같이 정의된다.

- **input:** Hermitian matrix $A \in \mathbb{C}^{N \times N}$, $b \in \mathbb{C}^N$
- **output:** $\tilde{x} \in \mathbb{C}^N$ s.t. $|\hat{x} - \tilde{x}| \leq \epsilon$ where $Ax = b$ and norm of solution $\|\tilde{x}\|$

Algorithm description

먼저 A 가 다음과 같은 eigenvalue, eigenvector를 가진다고 하자. ($0 \leq j < N$)

$$A|v_j\rangle = \lambda_j|v_j\rangle$$

이때, A 의 eigenvalue들이 다음을 따르며, d -bit representation으로 표현할 수 있다고 하자. (즉, A 는 positive matrix이다.)⁵⁵

$$0 < \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1} < 1$$

A 는 Hermitian이면서 동시에 positive matrix이므로 Hamiltonian simulation을 이용하면 다음과 같은 unitary를 만들 수 있다.⁵⁶

$$U = e^{i2\pi A}$$

⁵²without generality

⁵³정답 x 를 normalize한 state

⁵⁴solution이 $\|x\| \neq 1$ 이라면, quantum state vector로 표현 불가능하기에 우리가 QC를 통해 얻고자하는 결과는 \hat{x} 이다. 따라서 알고리즘의 연산 결과로 얻은 state가 normalize된 x 와 최대한 가깝게 만들고자 한다.

⁵⁵eigenvalue가 negative value를 가지는 경우에도 HHL algorithm을 적용할 수 있지만, 추후 사용할 QPE결과를 그에 맞춰서 수정해야 한다.

⁵⁶unitary로 설계되었기 때문에, 이제 quantum computer에서 사용할 수 있다!

이를 이용하면, HHL algorithm은 다음과 같이 동작한다.

1. input state : H 에 대한 eigenvector들의 superposition state인 $|b\rangle$ 로 2nd register를 초기화 한다.

$$|\psi\rangle = |0^d\rangle \sum_j \beta_j |v_j\rangle = |0^d\rangle |b\rangle$$

2. perform QPE : QPE를 수행하여, U 의 eigenvalue λ_j 들의 superposition state가 1st register에 저장되게 한다.

$$|\psi\rangle = \sum_j \beta_j |\lambda_j\rangle |v_j\rangle$$

Note (Idea). 이때, solution x 를 다음과 같이 표현할 수 있음을 이용하자.

$$A^{-1}|b\rangle = \left(\sum_j \lambda_j^{-1} |v_j\rangle \langle v_j| \right) |b\rangle = \sum_j \frac{\beta_j}{\lambda_j} |v_j\rangle$$

따라서 앞으로 우리가 할 일은 1st register에 대한 controlled rotation을 수행하여 각 β_j 에 λ_j^{-1} 을 곱하여 solution x 와 유사한 상태로 만드는 것이다.

3. apply controlled rotation unitary : U_{CR} 은 다음과 같이 정의된다. 즉, $|\lambda_j\rangle$ 의 값에 따라서 첫 번째 qubit이 $|1\rangle$ 방향으로 회전하게 된다.

$$U_{CR}|0\rangle |\lambda_j\rangle = \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) |\lambda_j\rangle$$

따라서 이렇게 정의된 gate를 $|\psi\rangle$ 에 가하면, 다음과 같은 상태가 된다.

$$|\psi\rangle = |0\rangle \sum_j \beta_j |\lambda_j\rangle |v_j\rangle \xrightarrow{U_{CR}} \sum_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \beta_j |\lambda_j\rangle |v_j\rangle$$

4. perform QPE^\dagger : QPE를 uncomputation하여 1st register⁵⁷를 초기상태 $|0\rangle^d$ 로 되돌린다. 따라서, ancilla bits들의 표현을 생략하면 최종적으로 얻게되는 state는 다음과 같다.

$$|\psi\rangle = \sum_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \beta_j |v_j\rangle.$$

5. measuring the single qubit

첫 번째 qubit의 값을 측정했을 때, outcome이 1이라면 post-state는 다음과 같으며,

$$\tilde{x} = \sum_j \frac{C\beta_j}{\tilde{\lambda}_j} |v_j\rangle$$

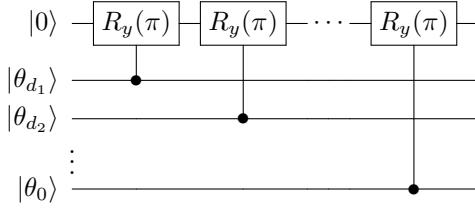
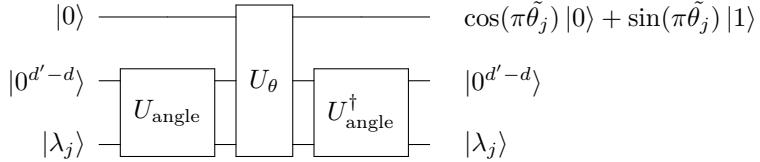
이를 normalize한 값; 즉 우리가 원하는 결과가 qubit에 저장되게 된다.

$$|\tilde{x}\rangle = \frac{\tilde{x}}{\|\tilde{x}\|} \approx |x\rangle$$

6. estimate $p(1)$ ⁵⁸

unnormalize solution x 를 얻기 위해서 필요한 값, $\|\tilde{x}\|$ 은 outcome이 1일 확률과 동일하다.

$$p(1) = \left\| \sum_j \frac{C\beta_j}{\tilde{\lambda}_j} |v_j\rangle \right\|^2 = \|\tilde{x}\|^2 \approx C^2 \|A^{-1}|b\rangle\|^2$$


 Figure 1.8: U_{angle} circuit

 Figure 1.9: U_{CR} circuit

$p(1)$ 은 C 가 커질수록 증가하기 때문에, C 값의 upper bound인 λ_0 이 가능한 커야 HHL algorithm⁵⁷을 효과적으로 동작할 수 있다. 또는 HHL을 적용하기 전에 **amplitude amplification**을 수행하여 최대한 $p(1)$ 이 커지도록 만들 수 있다.

마지막으로, 어떻게 U_{CR} 을 설계할 수 있는지에 대해 이야기 해보자. U_{CR} 은 d-bit representation으로 표현되는 $0 \leq \lambda_j < 1$ 값에 따라 computational basis에서 rotation을 수행한다. 이를 $0 \leq \theta < 1$, $\theta = 0.\theta_{d-1} \dots \theta_0$ 에 대해 표현하면, 다음과 같이 나타낼 수 있다.

$$U_\theta |0\rangle |\theta\rangle \rightarrow (\cos \pi\theta |0\rangle + \sin \pi\theta |1\rangle) |\theta\rangle$$

이 연산은 Y 축에 대한 rotation gate와 유사하게 동작한다.

$$e^{-i\tau \hat{Y}} = \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} \equiv R_y(2\tau)$$

따라서 $\tau = \pi(0.j_{d-1} \dots j_0)$ 으로 선택하여 $R_Y(2\tau)$ 를 이용하면, U_θ 를 구현할 수 있다. Fig. 1.8를 보면 우리가 QPE에서 사용했던 controlled operation과 다르게, λ_j 의 각 자릿수의 값에 따라 single qubit이 회전하는 구조로 이루어져 있음을 알 수 있다.

$$U_\theta = \sum_{j \in [2^d]} \exp(-i\pi(0 \cdot j_{d-1} \dots j_0) \hat{Y}) \otimes |j\rangle \langle j|$$

θ_j 를 다음과 같이 설정하면 $|\lambda_i\rangle$ 에 대한 controlled-rotation gate를 만들 수 있다.

$$\theta_j = \frac{1}{\pi} \sin^{-1}(C/\lambda_j)$$

θ 상태로 준비시키는 연산은 classical computer가 쉽게 할 수 있으므로 quantum computer도 uncomputation과 같은 기법을 사용하면, initial state를 $|\theta\rangle$ 로 전환하는 다음의 unitary를 만들 수 있다.

$$U_{\text{angle}} |0^{d'-d}\rangle |\lambda_j\rangle = |\theta_j\rangle$$

Lecture 16

1.11 Optimality of the quantum search algorithm

8 Nov. 17:00

이번 챕터에서는, quantum search algorithm의 optimality에 대해 다루어보고자 한다. Unstructured search problem⁵⁸의 single solution x 를 가진다고 하자. 그럼 이 문제를 해결하기 위해서 initial state $|\psi\rangle$ 에 대해 oracle O_x 를 k 번 적용하여 solution state에 가까워지도록 만들 수 있다. $O_x = I - 2|x\rangle \langle x|$ 는 $|x\rangle$ 에 대해서 phase -1 을 가하고 다른 state는 변화시키지 않는다.

⁵⁷ancilla

⁵⁸ $p(1)$ 을 empirical distribution으로부터 구할 수도 있고, 또는 amplitude estimation을 이용할 수도 있다. 만약 추정한 $p(1)$ 이 너무 낮다면 HHL을 사용하지 않는 결정을 내릴 수도 있다.

k 번 oracle을 적용하는 동안, unitary operation $U_1 \cdots U_k$ 를 각 연산의 사이에 적용한다고 하면, 다음과 같이 state evolution을 표현할 수 있다.

$$\begin{aligned} |\psi_k^x\rangle &\triangleq U_k O_x U_{k-1} O_x \cdots U_1 O_x |\psi\rangle \\ |\psi_k\rangle &\triangleq U_k U_{k-1} \cdots U_1 |\psi\rangle \end{aligned}$$

우리의 목표는 다음의 값에 대한 bound를 찾는 것이다.

$$D_k \triangleq \sum_x \| |\psi_k^x\rangle - |\psi_k\rangle \|^2$$

직관적으로, D_k 는 oracle을 적용하지 않았을 때의 state evolution $|\psi_k\rangle$ 를 기준으로, k 번 oracle을 적용했을 때 발생하는 편차를 나타낸다. 만약 D_k 의 값이 작다면, 어떠한 x 에 대해서도 그 state가 유사하다는 것이므로 높은 확률로 solution을 찾기 힘들 것이다.

증명은, 다음 과정을 따라 진행된다. (1) D_k 가 $O(k^2)$ 의 upper bound를 가진다는 것을 보인다. (2) N 개의 item을 구분하기 위해서는 D_k 는 $\Omega(N)$ 의 lower bound를 가져야 한다는 것을 보인다.
먼저 $D_k \leq 4k^2$ 를 보이자.⁵⁹ $k = 0$ 일 때는 자명하게 성립한다. k 일 때 $D_k \leq 4k^2$ 가 성립한다고 가정하면, $k + 1$ 일 때는 다음과 같이 쓸 수 있다.

$$D_{k+1} = \sum_x \|U_{k+1} O_x |\psi_k^x\rangle - U_{k+1} |\psi_k\rangle\|^2 = \sum_x \|O_x |\psi_k^x\rangle - |\psi_k\rangle\|^2 = \sum_x \|O_x (|\psi_k^x\rangle - |\psi_k\rangle) + (O_x - I) |\psi_k\rangle\|^2.$$

Norm에 대한 성질 $\|b + c\|^2 \leq \|b\|^2 + 2\|b\|\|c\| + \|c\|^2$ 을 적용하면, 다음과 같은 upper bound를 얻는다.

- Eq. (1.37) : Property of norm ($b \triangleq O_x (|\psi_k^x\rangle - |\psi_k\rangle)$ and $c \triangleq (O_x - I) |\psi_k\rangle = -2 \langle x | \psi_k \rangle |x\rangle$)
- Eq. (1.38) : Cauchy-Schwarz inequality
- Eq. (1.40) : Inductive hypothesis $D_k \leq 4k^2$

$$D_{k+1} \leq \sum_x \left(\| |\psi_k^x\rangle - |\psi_k\rangle \|^2 + 4 \| |\psi_k^x\rangle - |\psi_k\rangle \| \| \langle x | \psi_k \rangle | + 4 | \langle \psi_k | x \rangle |^2 \right) \quad (1.37)$$

$$\leq D_k + 4 \left(\sum_x \| |\psi_k^x\rangle - |\psi_k\rangle \|^2 \right)^{1/2} \left(\sum_{x'} | \langle \psi_k | x' \rangle |^2 \right)^{1/2} + 4 \quad (1.38)$$

$$= D_k + 4 \sqrt{D_k} + 4 \quad (1.39)$$

$$\leq 4k^2 + 8k + 4 = 4(k + 1)^2. \quad (1.40)$$

이제, N 개의 item을 구분하기 위해 $D_k = \Omega(N)$ 이 necessary condition임을 보이자.

증명을 위해서 먼저, $|\langle x | \psi_k^x \rangle|^2 \geq 1/2$ 가 모든 x 에 대해서 성립한다고 가정하자. 즉, 이는 어떤 solution x 가 주어지더라도 $1/2$ 이상의 확률로 solution을 구할 수 있다는 의미이다. (i.e., N 개의 item을 구분할 수 있다) 가정으로부터, 다음을 얻을 수 있다.

$$\| |\psi_k^x\rangle - |x\rangle \|^2 = 2 - 2 | \langle x | \psi_k^x \rangle | \leq 2 - \sqrt{2}$$

따라서 solution $|x\rangle$ 와 k 번 oracle을 적용하여 얻은 state $|\psi_k^x\rangle$ 의 차이를 error E_k 로 정의하면, 다음의 upper bound를 얻는다.⁶⁰

$$E_k \triangleq \sum_x \| |\psi_k^x\rangle - |x\rangle \|^2 \leq (2 - \sqrt{2})N. \quad (1.41)$$

Solution $|x\rangle$ 와 oracle을 적용하지 않았을 때의 state evolution $|\psi_k\rangle$ 의 차이를 F_k 로 정의하자.

$$F_k \triangleq \sum_x \| |x\rangle - |\psi_k\rangle \|^2$$

그렇다면, D_k 와 F_k 에 대해서 다음과 같은 관계를 유도할 수 있다.

⁵⁹by mathematical induction

⁶⁰ N 개의 item이 있으므로 가능한 x 의 개수는 N 개이다.

- Eq. (1.42) : By definition of D_k
- Eq. (1.43) : Property of norm ($b \triangleq |\psi_k^x\rangle - |x\rangle$ and $c \triangleq |x\rangle - |\psi_k\rangle$)
- Eq. (1.44) : By definition of D_k , F_k
- Eq. (1.45) : Property of norm

$$D_k = \sum_x \| |\psi_k^x\rangle - |x\rangle + |x\rangle - |\psi_k\rangle \|^2 \quad (1.42)$$

$$\geq \sum_x \| |\psi_k^x\rangle - |x\rangle \|^2 - 2 \sum_x \| |\psi_k^x\rangle - |x\rangle \| \| |x\rangle - |\psi_k\rangle \| + \sum_x \| |x\rangle - |\psi_k\rangle \|^2 \quad (1.43)$$

$$= E_k + F_k - 2 \sum_x \| |\psi_k^x\rangle - |x\rangle \| \| |x\rangle - |\psi_k\rangle \| \quad (1.44)$$

$$\geq E_k + F_k - 2\sqrt{E_k F_k} \quad (1.45)$$

$$= \left(\sqrt{E_k} - \sqrt{F_k} \right)^2 \quad (1.46)$$

Cauchy-Schwarz inequality를 이용하면, 다음이 성립함을 보일 수 있고

$$F_k \geq 2N - 2\sqrt{N} \quad (1.47)$$

$E_k \leq (2 - \sqrt{2})N$ 임을 이용하면 (see Eq. (1.41)), 충분히 큰 N 에 대해서 $c \gtrsim (\sqrt{2} - \sqrt{2 - \sqrt{2}})^2 \approx 0.42$ 보다 작을 때 다음이 성립함을 보일 수 있다.

$$D_k \geq cN$$

앞에서 우리가 보인 증명에 의하면, $D_k \leq 4k^2$ 이므로 다음을 얻는다.

$$k \geq \sqrt{cN/4}$$

따라서 정리하면, N 개의 item을 구분할 때, success probability가 적어도 $1/2$ 이상인 search algorithm의 optimal complexity는 $\Omega(\sqrt{N})$ 이다. \square

Chapter 2

Introduction to Computational Complexity

Lecture 16

2.1 Introduction

8 Nov. 17:00

Computational complexity는 computational problem을 풀기 위해서 필요로하는 자원의 종류인 time, space의 요구량을 분석하는 연구이다. Computational complexity의 주요 연구는 어떤 문제를 풀기 위한 best algorithm이 요구하는 자원의 양이 input size에 대해 어떤 lower bound를 가지는 것을 증명하는 것이다. 이는 심지어 그 문제를 해결하는 알고리즘이 구체적으로 무엇인지 알지 못하더라도 논의될 수 있다.

*Strong Church-Turing thesis*는 모든 computational model¹ probabilistic Turing machine에서 Polynomial time으로 시뮬레이션 할 수 있다고 주장한다. 양자 컴퓨터가 주목받고 있는 이유 중 하나로는 양자 컴퓨터가 classical computer가 polynomial time에 효과적으로 해결할 수 없으리라고 여겨졌던 문제들을 효과적으로 해결하는 경우가 발생하고 있기 때문이다. 이러한 결과들은 Strong Church-Turing thesis에 대해 의문을 제기하게 만든다.

Computational complexity를 다룰 때 있어서 주의해야 할 사항은 어떤 문제를 해결하는데 exponential resource가 요구된다는 것을 "엄밀하게" 증명하는 것은 매우 까다롭다는 사실이다. (e.g., NP problem) 이는 quantum computer에 대해서도 중요한 문제 중 하나인데 만약 양자컴퓨터가 polynomial time에 해결할 수 있는 NP problem이 사실은 P class에 속한다는 것이 증명된다면 quantum computer로서 얻을 수 있는 계산적 이점이 사라지기 때문이다.

2.2 The class NP: Reducibility and completeness

2.2.1 P and NP problems

우리는 지금부터 computational complexity를 다루기 위해 "YES / NO"라는 답변을 하는 *decision problem*에 대해서만 집중하고자 한다. Output이 2가지라는 측면에서, 이러한 problem은 output¹ one-bit은 boolean function $f : \{0, 1\}^* \rightarrow \{0, 1\}$ 로 나타낼 수 있다.

Turing machine¹ 어떤 language $L \subset \{0, 1\}^*$ 에 속하는 모든 word를 decide할 수 있을 때, 그 machine이 함수 $f_L : \{0, 1\}^* \rightarrow \{0, 1\}$, $f_L(x) = 1 \Leftrightarrow x \in L$ 를 계산한다고 한다.

Definition 2.2.1. 모든 $x \in \{0, 1\}^*$ 에 대해 polynomial time에 f_L 을 계산할 수 있는 turing machine M 이 존재한다면, language $L \subset \{0, 1\}^*$ 이 P class에 속한다고 한다.

$$x \in L \Leftrightarrow M(x) = 1, \quad x \notin L \Leftrightarrow M(x) = 0.$$

P는 직접적으로 주어진 문제를 해결하는 것과 관련 있다. 그러나 어떤 경우에는, 직접 문제를 푸는 것보다 적절한 hint가 주어졌을 때 해답을 검증하는 것이 더 효율적일 수 있다.¹

"효율적으로 검증 가능한 문제"를 엄밀하게 다루기 위해서 먼저 NP class를 정의하자.

¹efficiently verifiable solutions

Definition 2.2.2. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time TM M 과 polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language $L \subset \{0, 1\}^*$ 이 NP class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1 \\ x \notin L &\iff \forall u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 0 \end{aligned}$$

$x \in L$ 에 대해 $M(x, u) = 1$ 가 되게 하는 $u \in \{0, 1\}^{p(|x|)}$ 를 *witness*(or *witness*)라고 부른다.

a.i.e., M 이 polynomial time에 x 가 solution인지 아닌지를 확인할 수 있게 만드는 hint u

여기서 한 가지 주목할 점은 NP를 정의할 때, 그 문제를 풀기 위해서 요구되는 자원의 양이 얼마나 많이 필요한지는 정의하지 않았다는 점이다. NP는 단지 효율적으로 검증할 수 있는 문제들의 집합이다.

다음은 NP의 정의를 만족하는 decision problem의 예시들이다.

- Independent set problem: 그래프 G 와 k 가 주어졌을 때, k -size의 independent set이 존재하는지 확인하는 문제. → (*witness*) independent set이 존재한다면, 그 set의 원소들을 제공한다.
- Traveling salesman problem: 그래프 G 와 각 edge의 가중치가 주어졌을 때, 가중치의 합이 k 를 넘지 않으며 모든 노드를 정확히 한번씩만 방문하는 closed path가 존재하는지 확인하는 문제. → (*witness*) 최단 경로가 존재한다면, 그 경로를 제공한다.
- Subset sum problem: n 개의 정수 A_1, \dots, A_n 과 정수 T 가 주어졌을 때, 원소들의 합이 T 가 되는 subset이 존재하는지 확인하는 문제. → (*witness*) subset이 존재한다면, 그 subset의 원소들을 제공한다.

이렇게 P, NP를 정의하면 각 complexity class간의 관계를 생각해볼 수 있다.

- $P \subset NP^2$: 만약 $L \in P$ 가 TM N 을 사용하여 poly-time에 decide될 때, TM N 을 zero-polynomial witness $p(x)$ (i.e., empty)와 함께 사용하여 x 를 poly-time에 decide할 수 있으므로 $L \in NP$ 이다.
- $NP \subset EXP$: 만약 $L \in NP$ 가 TM M , witness $p()$ 를 사용하여 poly-time에 decide될 때, TM M 을 사용하여 모든 가능한 witness를 대입해보면서 x 가 solution인지 아닌지 decide할 수 있으므로 $L \in EXP$ 이다. ($2^{O(p(n))} \approx O(2^{n^c})$ time³)

Nondeterministic turing machine을 이용하면, NP를 다르게 정의할 수도 있다. NDTM은 매 step마다 2가지 서로 다른 transition function δ_0, δ_1 중에서 하나를 선택할 수 있다. 모든 x 에 대해 accept state에 도달하게 만드는 선택의 조합이 존재할 때, $M(x) = 1$ 이며 그렇지 않을 경우 $M(x) = 0$ 이다. NDTM에서의 witness는 이러한 선택의 조합을 제공하는 것이다.

Definition 2.2.3. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time NDTM M 과 polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language $L \subset \{0, 1\}^*$ 이 NP class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff M(x) = 1 \\ x \notin L &\iff M(x) = 0. \end{aligned}$$

2.2.2 Reducibility and NP-completeness

특정 class에 속하는 문제들중에서 가장 어려운 문제는 어떻게 정의할 수 있을까? Computational complexity에서는 이를 위해 reduction을 사용한다.

Definition 2.2.4. 다음을 만족하는 polynomial time computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ 이 존재한다면, language $L \subset \{0, 1\}^*$ 이 $L' \subset \{0, 1\}^*$ 에 polynomial time reducible이라고 한다.

$$\forall x \in \{0, 1\}^*, \quad x \in L \quad \text{if and only if} \quad f(x) \in L'.$$

직관적으로, reduction을 이용한다면 language L 에 속하는 word x 를 decide하는 대신, poly-time에 $f(x)$ 를 계산하여 language L' 에 속하는 $f(x)$ 를 decide하는 문제로 바꿔서 생각할 수 있다. 따라서 L 의 complexity는 L' 의 complexity보다 복잡할 수 없다.⁴

² $P \subset NP$ 인지는 아직 밝혀지지 않았다.

³ Since $O(p(n)) = O(n^c)$ for some constant c

⁴ L 을 푸는 문제를 L' 을 푸는 문제로 바꾸는데 걸리는 복잡도가 polynomial이므로

Definition 2.2.5. NP에 속하는 어떤 문제도 L' 로 polynomial time reduction될 때, L' 을 *NP-hard*라고 하며, NP-hard이면서 동시에 NP에 속하는 문제를 *NP-complete*이라고 한다.^a

^aNP-complete = NP-hard \cap NP

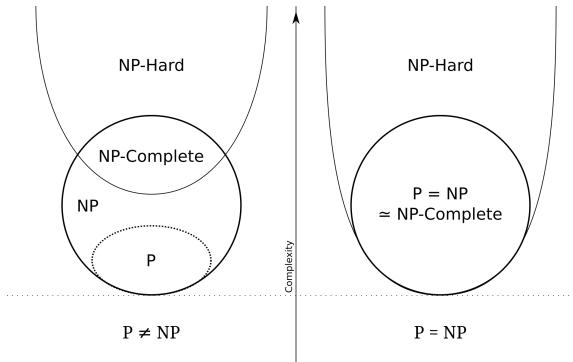


Figure 2.1: P and NP class [Np]

Lecture 17

2.2.3 Boolean formula and Cook-Levin theorem

11 Nov. 10:30

앞에서 우리는 NP에 속하는 여러 문제들의 예시를 살펴보았다. 이제 NP-complete에 속하는 문제 3-SAT problem를 소개하려고 한다. 이를 위해서 먼저 boolean formula에 대해 알아보자.

Boolean formula는 boolean variable u_1, \dots, u_n 과 logical operator NOT(\neg), OR(\vee), AND(\wedge)로 구성되는 수식이다. 어떤 boolean formula를 나타내는 변수를 φ 라고 정의할 때, boolean formula를 이루는 각각의 변수에 어떤 값을 대입하였는지를 $z \in \{0, 1\}^n$ 으로 표현하자. 이때, $\varphi(z) = 1$ 이 되도록 하는 z 가 존재한다면 φ 는 *satisfiable*이다. (e.g., $\varphi = u_1 \vee \neg u_2$ is satisfiable for $z = (1, 0)$)

특히 어떤 boolean formula가 *conjunctive normal form*이라는 의미는 변수들의 OR의 AND로 이루어진 수식이라는 의미이다. 예를 들어,

$$(u_1 \vee \bar{u}_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4) \wedge (\bar{u}_1 \vee u_3 \vee \bar{u}_4),$$

또는 일반화하여 나타내면 다음과 같다.

$$\bigwedge_i \left(\bigvee_j v_{i,j} \right)$$

이제 우리는 language SAT을 다음과 같이 정의할 수 있다.

Definition 2.2.6. Language **SAT**은 모든 가능한 satisfiable CNF formula들의 집합이다.^a 즉, SAT problem은 주어진 boolean formula x 가 satisfiable한지 결정하는 문제이다.

^a변수가 3개인 satisfiable CNF formula들의 집합을 3-SAT이라고 한다.

Theorem 2.2.1 (Cook-Levin theorem). 1. SAT is NP-complete, and 2. 3-SAT is NP-complete.

만약 주어진 boolean formula가 satisfiable하다면, $\varphi(z) = 1$ 이 되도록 하는 z 를 witness로 제공하면 문제를 풀 수 있기 때문에 SAT이 NP에 속한다는 것은 쉽게 증명할 수 있다. NP-hard에 속한다는 것을 증명하기 위해 NP에 속하는 모든 문제가 SAT로 reduction될 수 있음을 증명하면 된다.

이 수업에서는 SAT로의 reduction을 이용하는 대표적인 문제인 *Classical Hamiltonian problem*에 대해서 살펴보자 한다. 다음과 같은 3-local Hamiltonian을 가정하자.

$$H = \sum_{c=1}^m H_c(x_{c_1}, x_{c_2}, x_{c_3})$$

Hamiltonian problem은 ground state의 energy를 계산하는 문제이므로 다음과 같이 표현할 수 있다.

$$H_c(x_{c_1}, x_{c_2}, x_{c_3}) = 0, \quad x \in \{-1, 1\}$$

즉, boolean function의 값이 1이 되도록하는 3-SAT problem과 유사한 형태를 보이는 것을 알 수 있다.

$$f_c(u_{c_1}, u_{c_2}, u_{c_3}) = u_{c_1} \vee \bar{u}_{c_2} \vee u_{c_3} = 1, \quad u \in \{0, 1\}$$

이를 이용하면, SAT problem의 결과로부터 Hamiltonian problem의 결과를 쉽게 계산할 수 있다. 다음과 같이 Hamiltonian을 계산하면, $f_c = 1$ 이 되도록하는 조합에 대해, Hamiltonian의 energy가 0이 되는 것을 알 수 있다. (e.g., (1, 0, 0) for $f_c = 1$, then $H_c = 0$.)

$$\frac{1 - u_{c_1}}{2} \frac{1 + u_{c_2}}{2} \frac{1 - u_{c_3}}{2}$$

2.3 Quantum complexity

2.3.1 Probabilistic algorithms

Quantum computer의 complexity를 정의하기 전에 먼저, 우리에게 익숙한 deterministic algorithm에서 벗어나 probabilistic algorithm에 대해 살펴보자. Probabilistic algorithm을 수행하는 probabilistic Turing machine은 NDTM처럼 2개의 서로 다른 transition function δ_0, δ_1 을 가지며, 각각의 step마다 두 함수 중 하나를 1/2의 확률로 선택하여 수행한다. 확률에 따라서 동작하기 때문에 $x \in L$ 이 input으로 주어지더라도 TM이 종료되었을 때 reject state에 도달하게 되면 reject이라고 답한다.

Definition 2.3.1. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time PTM M 이 존재한다면, language $L \subset \{0, 1\}^*$ 이 BPP^a class에 속한다고 한다.^b

$$\Pr[M(x) = L(x)] \geq \frac{2}{3}$$

^abounded-error probabilistic polynomial

^b $M(x)$ 의 결과가 틀리지 않을 확률이 2/3이상

Definition 2.3.2. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time TM M 과 polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language $L \subset \{0, 1\}^*$ 이 BPP class에 속한다고 한다.^a

$$\Pr_{r \in \{0, 1\}^{p(|x|)}}[M(x, r) = L(x)] \geq \frac{2}{3}$$

^a이때 r 은 TM M 에 random하게 추가되는 random bits.

PTM에 대한 NP problem은 다음과 같이 정의된다.

Definition 2.3.3. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time TM M 과 polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language $L \subset \{0, 1\}^*$ 이 MA^a class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } \Pr_r[M(x, u, r) = 1] \geq \frac{2}{3} \\ x \notin L &\iff \forall u \in \{0, 1\}^{p(|x|)} \text{ s.t. } \Pr_r[M(x, u, r) = 0] \geq \frac{2}{3}. \end{aligned}$$

^aMerlin-Arthur

2.3.2 Quantum algorithms

이제 PTM에 대한 complexity class를 확장하여 Quantum computer에 대한 class를 정의해보자.

Definition 2.3.4. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 uniform family of polynomial time quantum circuits $\{Q_n : n \in \mathbb{N}\}$ 가 존재한다면, language $L \subset \{0, 1\}^*$ 이 BQP^a class에 속한다고 한다.

$$\Pr[Q_{|x|}(x) = L(x)] \geq \frac{2}{3}$$

^abounded-error quantum polynomial

BQP class는 Quantum circuit으로 문제를 풀었을 때, 오류가 발생할 확률이 $1/3$ 이하인 문제들의 집합이다. BQP class에 속하면서 P class에 속하지 않을 것이라고 믿는 문제들이 바로 quantum algorithm으로 해결하기 효과적인 문제들이다. (e.g., prime factorization) $Q(x)$ 를 실행한 뒤 첫 번째 qubit을 측정하여 그 값이 0인지 1인지를 기준으로 decidable problem을 해결할 수 있다.

Definition 2.3.5. 모든 $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 uniform family of polynomial time quantum circuits $\{Q_n : n \in \mathbb{N}\}$ 과 quantum state $|\psi\rangle$ 이 존재한다면, language $L \subset \{0, 1\}^*$ 이 QMA class에 속한다고 한다.

$$\Pr [Q_{|x|}(x, |\psi\rangle) = L(x)] \geq \frac{2}{3}$$

^aclassical algorithm의 hint 역할

QMA에 속하는 대표적인 예시로 k -local Hamiltonian problem^o 있다. 다음과 같이 주어진 k -local Hamiltonian^o 있을 때,

$$H = \sum_{i=1}^L H_i, \quad (\|H_i\|_1 \leq 1)$$

Hamiltonian의 ground state energy가 특정한 threshold b 이상인지, 또는 a 이하인지를 결정하는 문제이다. ($b - a = \Omega(n^{-\alpha})$) k -local Hamiltonian problem^o QMA에 속하는 이유는, phase estimation algorithm을 이용하여 eigen-value를 추정하고 그 값이 $b \geq, a \leq$ 인지를 확인하는 것으로 문제를 해결할 수 있기 때문이다.

더 나아가, k -local Hamiltonian problem은 QMA-complete이기 때문에 SAT problem처럼 QMA에 속하는 모든 문제들은 k -local Hamiltonian problem으로 reduction 될 수 있다.

2.3.3 BQP vs PSPACE

마지막으로, BQP class와 PSPACE class의 관계를 살펴보면서 complexity class간의 관계를 정리해보자.

Definition 2.3.6. TM^o input size^o 대해 polynomial space와 arbitrary time을 사용하여 문제를 해결할 수 있을 때, language $L \subset \{0, 1\}^*$ PSPACE class에 속한다고 한다.

$BQP \neq PSPACE$ 대해서는 아직 모르지만, $BQP \subset PSPACE$ 관계에 대해서는 증명할 수 있다.

$U = U_T \cdots U_1$ 인 unitary gate를 가정하자. ($T = \text{poly}(n)$)⁵ 이때 각 gate는 $\{C(X), \text{single-qubit gate}\}$ 중 하나이며, 따라서 각 gate는 최대 2개의 qubit에만 작용할 수 있다. Input qubit n 개에 추가로 중간 계산을 위해 ancilla qubit n 개를 사용한다고 가정하자. 그러면 U 를 적용한 뒤, 첫 번째 qubit을 관측했을 때 그 결과가 0이 될 확률은 amplitude를 계산해서 얻을 수 있다. ($|\mathbf{y}\rangle = |0\rangle |\mathbf{y}'\rangle$)

$$\langle \mathbf{y}|U_T \cdots U_1 |x^n, 0^n\rangle = \sum_{x_1, \dots, x_{T-1} \in \{0, 1\}^n} \langle \mathbf{y}|U_T |x_{T-1}\rangle \cdots \langle x_2|U_2|x_1\rangle \langle x_1|U_1|x^n, 0^n\rangle.$$

이 때, Σ 를 이루는 각각의 term을 계산하기 위해서 필요한 space는 poly-space^o므로 확률을 구하는 문제는 PSPACE에 속한다. 따라서 $BQP \subset PSPACE$ 이다.□

Note (Summary). $P \subset BPP \subset BQP \subset PSPACE$

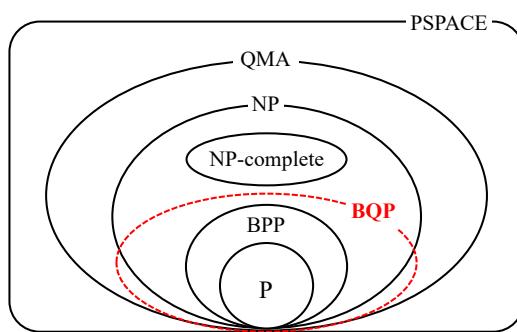


Figure 2.2: Computational complexity class

⁵ T 가 input qubit에 대해 polynomial이므로, 이 gate U 는 BQP에 속한다.

Chapter 3

Quantum Noise

Lecture 18

3.1 Introduction

18 Nov. 10:30

지금까지 우리는 quantum computer가 사용하는 system이 "완벽하다"고 가정하였다. (i.e., 외부 환경과 완벽히 독립된 단한 계) 하지만 실제로는 외부 환경과 상호작용하면서 noise가 발생한다. 따라서 이번 챕터에서는 *open system*에서의 quantum dynamics를 정의하고 이를 사용하여 noise를 나타내고 분석하는 방법을 다루고자 한다.

3.2 Axiomatic approach to quantum evolutions

Open system에서의 quantum evolution을 정의하기 위하여, 먼저 몇 가지 notation을 정의하고자 한다.

- $\mathcal{D}(\mathcal{H})$: Hilbert space \mathcal{H} 에 작용하는 density operator (density matrix)들의 집합
- $\mathcal{L}(\mathcal{H}_A, \mathcal{H}_B)$: Hilbert space \mathcal{H}_A 에서 Hilbert space \mathcal{H}_B 로 가는 linear operator들의 space¹

위의 표현을 이용하면, general quantum operation $\mathcal{E} : \mathcal{D}(\mathcal{H}_A) \rightarrow \mathcal{D}(\mathcal{H}_B)$ 을 나타낼 수 있다. ($\mathcal{E}(\rho_A) \in \mathcal{D}(\mathcal{H}_B)$ if $\rho_A \in \mathcal{D}(\mathcal{H}_A)$)는 general quantum operation을 나타낸다.²)

이제 quantum operation; channel이 convex linear이라고 가정하자. ($\lambda \in [0, 1]$ and $\rho_A, \sigma_A \in \mathcal{D}(\mathcal{H}_A)$)

$$\mathcal{E}(\lambda\rho_A + (1 - \lambda)\sigma_A) = \lambda\mathcal{E}(\rho_A) + (1 - \lambda)\mathcal{E}(\sigma_A) \quad (3.1)$$

Quantum operation \mathcal{E} 가 convex linear라는 것은 ensemble로 해석할 수 있다. 예를 들어, 어떤 동전을 던져서 그 결과에 따라 ρ_A, σ_A 두 상태중에서 하나의 상태로 준비한 뒤 quantum operation \mathcal{E} 를 적용하는 상황을 가정하자. 만약 두 주어진 상태가 ρ_A, σ_A 중에서 무엇인지 알지 못하는 상황이라면, mixed state에 operation을 가한 상황으로 해석되기 때문에 이는 Eq. (3.1)의 좌변에 해당된다. 반대로, 두 주어진 상태가 ρ_A, σ_A 중에서 무엇인지 알고 있다면, 이는 Eq. (3.1)의 우변에 해당될 것이다.

이제 quantum channel의 정의역과 공역을 density operator만이 아닌 모든 linear operator에 대해서 확장하자. 이때 quantum channel이 convex linearity를 만족하기 위한 조건은 다음과 같다.

1. A quantum channel \mathcal{E} is a linear map

$$\mathcal{E}(\alpha X_A + \beta Y_A) = \alpha\mathcal{E}(X_A) + \beta\mathcal{E}(Y_A) \quad (3.2)$$

2. A quantum channel \mathcal{E} is positive map : Density operator는 반드시 density operator로 사상되어야 하므로 연산 전 후 positivity를 보존해야 한다.

Definition 3.2.1 (Positive map). 만약 모든 positive semi-definite $X_A \in \mathcal{L}(\mathcal{H}_A)$ 에 대해서, 사상 결과 $\mathcal{M}(X_A)$ 가 positive semi-definite 이라면 $\mathcal{M} : \mathcal{L}(\mathcal{H}_A) \rightarrow \mathcal{L}(\mathcal{H}_B)$ 은 positive map이다.

¹ $\mathcal{L}(\mathcal{H})$: 자기 자신(Hilbert space \mathcal{H})으로 가는 square linear operator들의 spaces

² operator가 작용하기 전과 후, Hilbert space가 같지 않아도 된다는 것에 유의하라.

3. A quantum channel \mathcal{E} is *completely positive map* : 위의 조건에 더해, composite system과 entanglement를 고려해도 여전히 positivity를 유지하기 위해서 completely positive map의 조건으로 확장된다. 예를 들어, composite system $\rho_{RA} \in \mathcal{D}(\mathcal{H}_R \otimes \mathcal{H}_A)$ 에 대해서 quantum channel을 오직 system A 에만 가하더라도 (i.e., $I_R \otimes \mathcal{E}_A$) 전체 composite system의 positivity를 보존해야 한다. 수학적으로, $\{|i\rangle\}_R$ 를 \mathcal{H}_R 의 orthonormal basis라고 하자. 이때 모든 operator $X_{RA} \in \mathcal{L}(\mathcal{H}_R \otimes \mathcal{H}_A)$ 는 다음과 같이 표현할 수 있다.

$$X_{RA} = \sum_{i,j} |i\rangle \langle j|_R \otimes X_A^{i,j}$$

이때 $I_R \otimes \mathcal{E}_{A \rightarrow B}$ 의 작용은 다음과 같이 표현된다.

$$\begin{aligned} (\mathcal{I}_R \otimes \mathcal{E}_{A \rightarrow B})(X_{RA}) &= (\mathcal{I}_R \otimes \mathcal{E}_{A \rightarrow B}) \left(\sum_{i,j} |i\rangle \langle j|_R \otimes X_A^{i,j} \right) \\ &= \sum_{i,j} |i\rangle \langle j|_R \otimes \mathcal{E}_{A \rightarrow B}(X_A^{i,j}) \end{aligned}$$

즉, local operation은 reference system R 에 대해서는 아무런 효과를 주지 않는다.

Definition 3.2.2 (Completely positive map). 만약, 어떤 arbitrary size의 reference system R 에 대해서도 $I_R \otimes \mathcal{M}$ 이 positive map이라면, \mathcal{M} 은 completely positive map이다.

4. A quantum channel \mathcal{E} is *trace-preserving* : density operator는 그대로 density operator로 사상되어야 하므로 trace를 보존해야한다. 이를 linear operator로 확장하면 모든 linear operator에 대해 trace를 보존해야한다.

$$\text{Tr}(\mathcal{E}(X_A)) = \text{Tr}(X_A)$$

따라서 정리하자면, *Quantum channel*은 다음을 만족해야한다.

Definition 3.2.3. A quantum channel is a *linear, completely positive, trace preserving map*.

위의 3가지 조건을 일일이 확인하지 않고도 쉽게 확인할 수 있는 방법이 있다. 다음 Theorem에 따르면, 3가지 조건을 모두 만족하는 map은 Choi-Kraus decomposition이 가능하며, Choi-Kraus decomposition을 가지는 map은 3가지 조건을 모두 만족한다.

Theorem 3.2.1 (Choi-Kraus). A map $\mathcal{E} : \mathcal{L}(\mathcal{H}_A) \rightarrow \mathcal{L}(\mathcal{H}_B)$ is linear, completely positive, and tracepreserving if and only if it has a Choi-Kraus decomposition:

$$\mathcal{E}_{A \rightarrow B}(X_A) = \sum_{l=0}^{d-1} V_l X_A V_l^\dagger,$$

where $X_A \in \mathcal{L}(\mathcal{H}_A)$, $V_l \in \mathcal{L}(\mathcal{H}_B, \mathcal{H}_A)$ for all $l \in \{0, \dots, d-1\}$, and the Kraus operator satisfy the completeness relation and $d \leq \dim(\mathcal{H}_A) \times \dim(\mathcal{H}_B)$.

$$\sum_{l=0}^{d-1} V_l V_l^\dagger = I_A$$

Proof. See Appendix A. ■

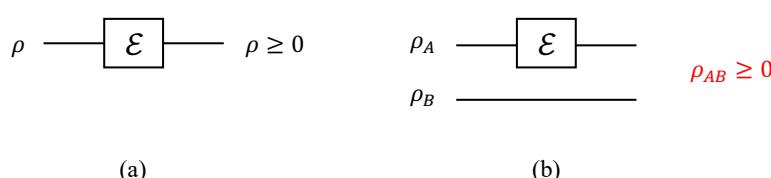


Figure 3.1: (a) positive map, (b) completely positive map

3.3 Interpretation of quantum channels

이번 섹션에서는 *Quantum channel*의 두 가지 해석을 다루고자 한다.

3.3.1 Noisy evolution as the loss of a measurement outcome

어떤 System의 density operator ρ 로 표현되고 measurement set $\{M_k\}$ 를 사용하여 측정을 수행한다고 하자. ($\sum_k M_k^\dagger M_k = I$) 이때, outcome k 를 얻을 확률은 잘 알다시피 다음과 같고

$$p(k) = \text{Tr} [M_k^\dagger M_k \rho]$$

post-measurement state는 다음과 같다.

$$\frac{M_k \rho M_k^\dagger}{p(k)}$$

그런데 만약 우리가 측정해서 얻은 outcome이 무엇인지 알지 못한다면 post-measurement state는 가능한 모든 outcome들에 대해 mixed state로 표현된다. Ensemble description으로 표현하면 다음과 같고

$$\left\{ p(k), \frac{M_k \rho M_k^\dagger}{p(k)} \right\}$$

이는 다음과 같은 density matrix로 표현된다.

$$\sum_k p(k) \frac{M_k \rho M_k^\dagger}{p(k)} = \sum_k M_k \rho M_k^\dagger$$

위 수식에서처럼 mixed state의 density matrix는 Choi-Kraus decomposition 표현과 동일하다. 따라서 quantum channel을 measurement를 수행하고 outcome을 확인하지 않는 연산으로 생각할 수 있다.

3.3.2 Noisy evolution from a unitary interaction

Quantum channel을 해석하는 또 다른 방법은 environment가 unitary interaction을 통해 우리의 system에 영향을 주는 상황으로 이해하는 것이다. Quantum system A 는 state ρ_A 로, environment system E 는 pure state $|0\rangle_E$ 로 초기화되었다고 하자.³ 전체 system에 대해 joint time evolution, U_{AE} 가 발생하면 우리는 environment system E 에 대한 접근 권한이 없기 때문에, partial trace를 취한 system A 에 대한 상태만을 얻을 수 있다.

$$\sigma_A = \text{Tr}_E [U_{AE} (\rho_A \otimes |0\rangle\langle 0|) U_{AE}^\dagger]$$

이 상태를 정리하면, Kraus representation을 얻을 수 있다. $(B_i \triangleq \langle i_E | U_{AE} | 0_E \rangle)$

$$\text{Tr}_E [U_{AE} (\rho_A \otimes |0\rangle\langle 0|) U_{AE}^\dagger] = \sum_i \langle i_E | U_{AE} (\rho_A \otimes |0\rangle\langle 0|) U_{AE}^\dagger | i \rangle_E = \sum_i B_i \rho_A B_i^\dagger$$

step 2 understand

B_i 가 completeness relation을 만족하는지 쉽게 확인할 수 있다.

$$\begin{aligned} \sum_i B_i^\dagger B_i &= \sum_i \langle i_E | U_{AE} | 0 \rangle_E^\dagger \langle i_E | U_{AE} | 0 \rangle_E = \sum_i \langle 0 |_E U_{AE}^\dagger | i \rangle_E \langle i |_E U_{AE} | 0 \rangle_E \\ &= \langle 0 |_E U_{AE}^\dagger \sum_i | i \rangle_E \langle i |_E U_{AE} | 0 \rangle_E = \langle 0 |_E U_{AE}^\dagger U_{AE} | 0 \rangle_E = I_A. \end{aligned}$$

반면, Kraus operator가 noisy evolution을 나타낸다는 사실도 쉽게 확인할 수 있다. Kraus operator $\{E_k\}$, $\sum_k E_k^\dagger E_k = I_A$ 에 대해 $\{|e_k\rangle\}$ 를 각각 E_k 에 대응되는 orthogonal basis set이라고 하자. $|\psi\rangle |e_0\rangle$ 에 대해서 다음과 같이 동작하는 operator U 를 정의하자.⁴

$$U|\psi\rangle |e_0\rangle \equiv \sum_k E_k |\psi\rangle |e_k\rangle,$$

³Environment를 pure state로 표현할 수 있는 이유는, environment가 pure state가 아니더라도 얼마든지 reference system을 추가하여 purification을 적용하면 pure state로 표현할 수 있기 때문이다.

⁴ $|e_0\rangle$ 는 environment의 임의의 standard state이다.

U 는 다음과 같이 composite system에 대해 norm을 보존한다. (by completeness relation of Kraus operator)

$$\langle \psi | e_0 | U^\dagger U | \varphi \rangle | e_0 \rangle = \sum_k \langle \psi | E_k^\dagger E_k | \varphi \rangle = \langle \psi | \varphi \rangle,$$

따라서, operator U 는 전체 composite system에 대해서 동작하는 unitary operator로 생각할 수 있다.

$$\text{Tr}_E [U (\rho \otimes |e_0\rangle\langle e_0|) U^\dagger] = \sum_k E_k \rho E_k^\dagger.$$

3.3.3 Freedom in the operator-sum representation

지금까지 우리는 operator-sum representation을 사용하여 general quantum channel을 표현할 수 있다는 것을 확인하였다. 그렇다면, 과연 Choi-Kraus decomposition은 unique한 표현인가?

이를 확인하기 위해서, single qubit에 작용하는 2가지 channel \mathcal{E}, \mathcal{F} 를 가정하자. 각 channel은 다음과 같은 Kraus operator $\{E_1, E_2\}, \{F_1, F_2\}$ 를 각각 가진다.

$$E_1 = \frac{I}{\sqrt{2}}, \quad E_2 = \frac{Z}{\sqrt{2}}, \quad F_1 = |0\rangle\langle 0|, \quad F_2 = |1\rangle\langle 1|.$$

놀랍게도 두 channel은 동일하다!

$$\mathcal{F}(\rho) = \frac{1}{2} [(E_1 + E_2)\rho(E_1^\dagger + E_2^\dagger) + (E_1 - E_2)\rho(E_1^\dagger - E_2^\dagger)] = E_1 \rho E_1^\dagger + E_2 \rho E_2^\dagger = \mathcal{E}(\rho).$$

이처럼 동일한 두 channel을 표현하는 서로 다른 Kraus operator가 존재하는 반례를 찾았으므로 Choi-Kraus decomposition은 unique하지 않다. Operator-sum representation은 다음 정리를 따른다.

Theorem 3.3.1. $\{E_i\}_{i=1}^m$ 과 $\{F_i\}_{i=1}^n$ 를 각각 \mathcal{E} 와 \mathcal{F} 에 대한 operation elements라고 하자. 두 집합 중에 서 더 크기가 작은 집합에 zero operator를 추가하여, 두 집합의 크기가 동일하도록 만들 수 있다. 이때 $E_i = \sum_j u_{ij} F_j$ 가 되도록하는 complex number u_{ij} 가 존재하면, $\mathcal{E} = \mathcal{F}$ 이다.

3.4 Examples

3.4.1 Bit flip and phase flip channels

Bit flip channel은 $1-p$ 의 확률로 X gate를 가하는 channel로, 다음의 operation elements를 가진다.

$$E_0 = \sqrt{p}I = \sqrt{p} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E_1 = \sqrt{1-p}X = \sqrt{1-p} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Kraus operator들을 이용하면 bit flip channel이 다음과 같이 동작함을 알 수 있다.

$$\mathcal{E}(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger = p \cdot \rho + (1-p) \cdot X \rho X^\dagger$$

Phase flip channel은 $1-p$ 의 확률로 Z gate를 가하는 channel로, 다음의 operation elements를 가진다.

$$E_0 = \sqrt{p}I = \sqrt{p} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E_1 = \sqrt{1-p}Z = \sqrt{1-p} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Phase flip channel은 dephasing이라고도 불리는데, 이는 output state를 다음과 같이 표현할 수 있기 때문이다. 만약 $p = 1/2$ 라면, output state는 대각성분만 남게된다.

$$p \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix} + (1-p) \begin{pmatrix} \rho_{00} & -\rho_{01} \\ -\rho_{10} & \rho_{11} \end{pmatrix} = \begin{pmatrix} \rho_{00} & (2p-1)\rho_{01} \\ (2p-1)\rho_{10} & \rho_{11} \end{pmatrix}.$$

Bit-phase flip channel은 $1-p$ 의 확률로 Y gate를 가하는 channel이다.

$$E_0 = \sqrt{p}I = \sqrt{p} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E_1 = \sqrt{1-p}Y = \sqrt{1-p} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

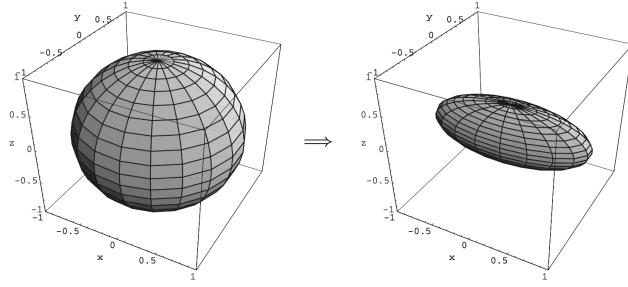


Figure 3.2: Bit flip channel [NC01]

3.4.2 Depolarizing channel

Depolarizing channel은 p 의 확률로 completely mixed state $I/2$ 가 되는 channel으로, $I/2$ 는 기존 state에 대해 아무런 정보도 제공해주지 않기 때문에 quantum noise에서 매우 중요하게 다뤄지는 noise type이다. Channel을 통과한 후 state는 다음과 같이 표현된다.

$$\mathcal{E}(\rho) = \frac{pI}{2} + (1-p)\rho$$

다음의 관계를 이용하면,

$$I = \frac{1}{2}(\rho + X\rho X + Y\rho Y + Z\rho Z)$$

다음과 같이 operator-sum representation으로 나타낼 수 있다.

$$\mathcal{E}(\rho) = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z)$$

확률을 더 간단하게 표현하기 위해 다음과 같이 나타내기도 한다.

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$$

3.4.3 Amplitude damping channel

마지막으로 소개할 noise는 physical noise의 개념에 더 가깝다.⁵ High energy state인 $|1\rangle\langle 1|$ 에서 점점 energy가 감소하면서 $|0\rangle\langle 0|$ 으로 상태가 변화하는 noise이다.

$$\mathcal{E}(\rho) = E_0\rho E_0^\dagger + E_1\rho E_1^\dagger$$

Operation elements는 각각 다음과 같다.

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, \quad E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}.$$

즉, E_1 은 $|1\rangle$ state를 $|0\rangle$ state로 변화시키며 E_2 는 $|0\rangle$ state의 amplitude를 감쇠시킨다. 따라서 이 channel을 사용하면 $|0\rangle$ state는 항상 그대로 유지되지만 $|1\rangle$ state에 대한 amplitude는 점차 감쇠되게 된다.

⁵앞서 소개한 noise들은 I operator에 대해서 여전히 I 로 동작함을 보장하지만 이 noise는 이를 보장하지 않는다.

Chapter 4

Quantum Error Correction and Fault Tolerance

Lecture 19

4.1 Introduction

20 Nov. 10:30

Quantum noise는 universal quantum computer를 설계하기 위해 가장 중요하게 다뤄져야 하는 요소이다. Chapter 3에서는 quantum error를 표현하는 방법을 배웠고 이제 이 챕터에서는 quantum error를 정정하는 방법을 소개하려고 한다.

4.2 Some distance measure

Quantum noise로 인해서 state가 얼마나 많이 달라졌는지를 평가하기 위해서, *distance measure*를 정의해야 한다. 이를 위해서 먼저 classical information에 대한 distance measure를 소개하고 이를 quantum information에 대해서 확장한 distance measure를 소개한다.

4.2.1 Distance measures for classical information

동일한 random variable X 에 대해, 서로 다른 두 probability distribution $\{p_x\}$ 와 $\{q_x\}$ 를 가정하자. 이 두 distribution의 차이를 나타내기 위해 사용할 수 있는 한 가지 방법은 *total variation distance (TVD)*이다:

$$D(p, q) \triangleq \frac{1}{2} \sum_x |p_x - q_x|$$

만약 두 분포가 동일하다면, TVD는 0이 된다. 반대로 두 분포가 *orthogonal*하다면, TVD는 1이된다. Total variation distance는 다음과 같은 성질을 가진다.

- Symmetry: $D(p, q) = D(q, p)$
- Non-negativity: $D(p, q) \geq 0$
- Triangle inequality: $D(p, q) \leq D(p, r) + D(r, q)$

반면, *fidelity*를 사용하여 두 분포의 차이를 나타낼 수도 있다.

$$F(p_x, q_x) \triangleq \sum_x \sqrt{p_x q_x}$$

마찬가지로 두 분포가 동일하다면, fidelity는 1이 되고 반대로 두 분포가 *orthogonal*하다면 fidelity는 0이 된다. Fidelity 역시 0과 1사이의 값을 가지며 TVD와 반비례하는 값을 가진다.

(*operational meaning*) TVD는 sample space의 가능한 모든 subset S 에 대해서 두 분포의 차이가 최대가 되게하는 S 의 space에 대한 TVD와 동일한 값을 가진다.

$$D(p, q) = \max_S |p(S) - q(S)| = \max_S \left| \sum_{x \in S} p_x - \sum_{x \in S} q_x \right|,$$

4.2.2 Trace distance

Trace distance는 TVD를 quantum distance measure로 확장한 것이다.¹ ($|A| \triangleq \sqrt{A^\dagger A}$)

$$D(\rho, \sigma) \equiv \frac{1}{2} \text{Tr}[|\rho - \sigma|],$$

Trace distance가 가지는 중요한 성질중에 하나는 바로 unitary operation에 대해서 invariant하다는 것이다.

$$D(\rho, \sigma) = D(U\rho U^\dagger, U\sigma U^\dagger)$$

Trace distance는 다음과 같이 나타낼 수도 있으며, 이는 가능한 모든 POVM set에 대해서 두 trace의 차이가 최대가 되도록 하는 POVM에 대한 trace distance로 정의할 수 있음을 의미한다. (TVD의 두 번째 정의)

$$D(\rho, \sigma) = \max_{\{M_x\}} |\text{Tr}[M_x \rho] - \text{Tr}[M_x \sigma]|,$$

Trace distance에 대해 중요한 특징이 존재하는데 이는 동일한 trace preserving quantum operator를 가했을 때, trace distance는 증가할 수 없다는 정리이다.

$$D(\mathcal{E}(\rho), \mathcal{E}(\sigma)) \leq D(\rho, \sigma)$$

4.2.3 Quantum fidelity

다음으로 살펴볼 Quantum fidelity는 fidelity를 quantum distance measure로 확장한 것이다.

$$F(\rho, \sigma) \triangleq \text{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}}$$

Quantum fidelity는 trace distance와 유사하게 다음과 같은 성질을 가진다.

- Symmetry: $F(\rho, \sigma) = F(\sigma, \rho)$
- Invariant of unitary: $F(\rho, \sigma) = F(U\rho U^\dagger, U\sigma U^\dagger)$
- Operation inequality: $F(\mathcal{E}(\rho), \mathcal{E}(\sigma)) \geq F(\rho, \sigma)$

만약, $\rho = \sum_i r_i |i\rangle\langle i|, \sigma = \sum_i s_i |i\rangle\langle i|$ 가 commute하다면 quantum fidelity는 classical fidelity와 같다.

$$F(\rho, \sigma) = \text{Tr} \left[\sqrt{\sum_i r_i s_i |i\rangle\langle i|} \right] = \sum_i \sqrt{r_i s_i} = F(r_i, s_i)$$

또한, 만약 두 state중에서 적어도 하나가 pure state라면, quantum fidelity는 다음과 같이 나타낼 수 있다.²

$$F(|\psi\rangle, \rho) = \text{Tr}[\sqrt{\langle\psi|\rho|\psi\rangle}] = \sqrt{\langle\psi|\rho|\psi\rangle}.$$

Quantum fidelity의 중요한 정의는 Uhlmann's theorem에 의해 주어진다. 이 정리는 양자 상태 사이의 fidelity를 pure state들의 overlap으로 생각할 수 있음을 주장한다.

Theorem 4.2.1 (Uhlmann's theorem). ρ 와 σ 가 quantum system Q 의 state일 때, quantum system Q 의 복사본인 다른 system R 에 대해 다음이 만족한다.

$$F(\rho, \sigma) = \max_{|\psi\rangle, |\varphi\rangle} |\langle\psi|\varphi\rangle|$$

$|\psi\rangle, |\varphi\rangle$ 는 ρ, σ 에 대해 RQ 로 purification하여 나타낸 가능한 모든 pure state이다.

Quantum fidelity는 가능한 모든 POVM에 대한 classical fidelity를 사용하여 정의할 수 있다.

$$F(\rho, \sigma) = \min_{\{\Pi_m\}} F(p_m, q_m)$$

마지막으로, trace distance와 fidelity는 다음과 같은 관계를 만족한다.

$$1 - F(\rho, \sigma) \leq D(\rho, \sigma) \leq \sqrt{1 - F(\rho, \sigma)^2}.$$

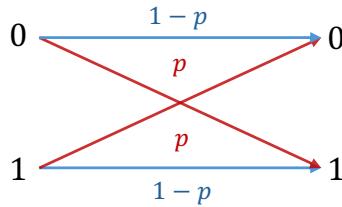


Figure 4.1: Classical bit flip

4.3 The basic code and Shor code

4.3.1 Classical error correction

Classical channel에서 발생할 수 있는 가장 간단한 오류인 bit flip error를 생각해보자. 아무런 인코딩도 하지 않는다면 수신자의 입장에서 bit 1을 받았을 때 원래 1인지 아니면 0이었는데 bit flip error가 발생한 것인지를 구분할 수 없다. 이를 해결하기 위해 0이나 1 대신에 비트를 반복해서 000, 111으로 인코딩하여 bit flip error를 해결할 수 있다. 만약 오류의 발생 확률 p 가 충분히 작다면 오류가 발생한 bit string이 001일 때 000 → 001의 확률이 111 → 001의 확률보다 작다는 사실을 이용하여 오류 정정을 할 수 있다.

*Repetition code*의 성공 확률은 오류가 발생하지 않을 확률과 하나의 단일 비트에만 오류가 발생했을 확률의 합이며,

$$(1 - p)^3 + 3p(1 - p)^2 = 1 + 2p^3 - 3p^2$$

실패 확률은 2개의 이상의 비트에 오류가 발생했을 확률이다.

$$P_e = 3p^2(1 - p) + p^3 = -2p^3 + 3p^2$$

그렇다면, 오류 확률 p 가 얼마나 작을 때 오류 정정 코드를 사용하는 것이 유리한지를 오류 정정을 하지 않았을 때의 실패 확률 p_e 와 비교하여 계산할 수 있다.

$$p_e = -2p^3 + 3p^2 < p \rightarrow p < \frac{1}{2}$$

4.3.2 The three qubit bit flip code

Classical communication에서 이미 검증된 repetition code를 quantum communication에 바로 적용할 수 없는 이유는 다음과 같다.

- No cloning : 주어진 임의의 양자상태를 복사할 수 없으므로 repetition이 불가능하다.
- Continuous error : 양자 채널은 bit flip error 말고도 다른 다양한 오류가 존재한다.
- Measurement : 오류 정정을 위해서 state를 확인하면 기존의 quantum state가 붕괴된다.

지금부터는 이러한 문제들을 어떤 방식을 도입하여 해결하였는지 소개하려고 한다. 먼저, $|\psi\rangle$ 에 p 의 확률로 X gate가 작용하여 state가 $X|\psi\rangle$ 로 변하는 bit flip error에 대해서만 생각해보자. 이를 해결하기 위해서 $|0_L\rangle, |1_L\rangle$ 을 physical qubit 3개로 표현되는 logical qubit으로 인코딩하자.³

$$a|0\rangle + b|1\rangle \rightarrow a|000\rangle + b|111\rangle \equiv a|0_L\rangle + b|1_L\rangle,$$

그러나, logical qubit을 사용하더라도 여전히 오류가 발생했는지 확인하기 위해서 중간에 상태를 관측하면 기존의 정보를 모두 잃어버리게 되므로 state를 관측하지 않고도 오류의 발생을 감지할 수 있는 다른 방법이 필요하다. 이를 위해서 *Syndrome measurement*를 제안한다.

- $P_0 = |000\rangle\langle 000| + |111\rangle\langle 111|$: 오류가 발생하지 않음
- $P_1 = |100\rangle\langle 100| + |011\rangle\langle 011|$: 첫 번째 qubit에 bit flip이 발생함
- $P_2 = |010\rangle\langle 010| + |101\rangle\langle 101|$: 두 번째 qubit에 bit flip이 발생함
- $P_3 = |001\rangle\langle 001| + |110\rangle\langle 110|$: 세 번째 qubit에 bit flip이 발생함

¹quantum density matrix에서 대성분은 확률을 의미 한다.

²계산이 간단하다.

³두 state $|0\rangle, |1\rangle$ 은 서로 orthogonal state이므로 복사할 수 있다.

이 syndrome measurement set을 사용하면, 발생할 수 있는 모든 오류⁴들의 subspace에 projection 시키기 때문에 측정 후에도 quantum state의 정보는 그대로 유지된다.

$$\frac{P_0 |\psi\rangle}{\langle \psi | P_0^\dagger P_0 |\psi\rangle} = \frac{a|000\rangle + b|111\rangle}{a\langle 000| + b\langle 111|(\langle 000\rangle\langle 000| + \langle 111\rangle\langle 111|)a|000\rangle + b|111\rangle} = \frac{a|000\rangle + b|111\rangle}{a^2 + b^2} = |\psi\rangle$$

Syndrome measurement set을 사용해서 얻은 outcome을 확인하면, 어떤 qubit에 bit flip이 발생하였는지 알 수 있고 그 qubit에 X gate를 가하여 오류를 정정할 수 있다.

Quantum repetition code의 실패 확률 역시 classical repetition code와 동일하게 $P_e = -2p^3 + 3p^2$ 이며, $p < 1/2$ 일 때 오류 정정을 하지 않을 때보다 더 정확도가 증가한다. 그러나 quantum bit flip error는 classical bit flip error처럼 단순하게 실패 확률을 분석해서는 안된다. 예를 들어, 오류가 발생하기 전 상태가 $|+\rangle$ 였다면, bit flip error는 아무런 영향을 주지 못하기 때문이다. 따라서 state distance의 관점에서 오류를 분석할 필요가 있다. (오류가 없는 이상적인 상태는 항상 pure state이므로 fidelity를 $F = \sqrt{\langle \psi | \rho | \psi \rangle}$ 로 쉽게 계산할 수 있으므로 fidelity를 사용하자.)

Quantum error correction의 목표는 fidelity를 증가시키는 것이다.⁵ 만약 오류 정정을 하지 않는다면, output state는 다음과 같이 표현된다.

$$\rho = (1-p)|\psi\rangle\langle\psi| + pX|\psi\rangle\langle\psi|X.$$

따라서 두 상태간의 fidelity는 다음과 같이 산된다. 주어진 항에서 $\langle\psi|X|\psi\rangle$ 는 확률을 나타내는 값이므로 음수가 될 수 없다. $|\psi\rangle$ 가 computational state일 때, $\langle\psi|X|\psi\rangle = 0$ 이므로 minimum fidelity는 $\sqrt{1-p}$ 이라는 사실을 알 수 있다.

$$F(|\psi\rangle\langle\psi|, \rho) = \sqrt{\langle\psi|\rho|\psi\rangle} = \sqrt{(1-p) + p\langle\psi|X|\psi\rangle\langle\psi|X|\psi\rangle} \quad (4.1)$$

반면, 오류 정정을 수행했다면 1 bit flip error가 발생했더라도 오류정정을 통해 pure state를 그대로 유지할 수 있으므로 output state는 다음과 같고

$$\rho_{QEC} = [(1-p)^3 + 3p(1-p)^2] |\psi\rangle\langle\psi| + \text{positive terms} \dots$$

fidelity는 다음의 lower bound를 가진다.

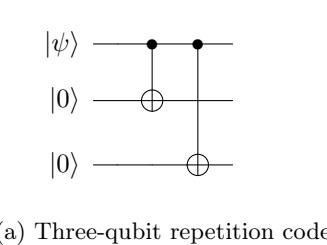
$$F(|\psi\rangle\langle\psi|, \rho_{QEC}) = \sqrt{(1-p)^3 + 3p(1-p)^2 + \text{positive terms}} \dots \geq \sqrt{(1-p)^3 + 3p(1-p)^2} \quad (4.2)$$

Eq. (4.1)보다 Eq. (4.2)의 값이 더 커지게 만들기 위해서는 $p < 1/2$ 여야 한다.

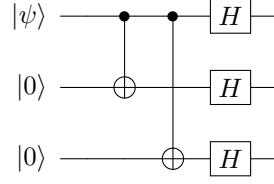
Syndrome measurement를 이해하는 한 가지 다른 방법이 있다. 앞에서 제안한 4개의 projector를 사하는 대신 2개의 observable을 측정한다고 생각해보자. 첫 번째로 $Z_1 Z_2$ ⁶를 측정하고

$$(Z_1 Z_2) + 1 : (|00\rangle\langle 00| + |11\rangle\langle 11|)_{12} \otimes I_3, \quad -1 : (|01\rangle\langle 01| + |10\rangle\langle 10|)_{12} \otimes I_3$$

이후 $Z_2 Z_3$ 을 측정한다. 각각의 observable을 측정했을 때, -1 이라는 outcome을 얻으면 해당 bit들의 값이 다르다는 것을 의미한다. 예를 들어 $Z_1 Z_2$ 의 outcome이 -1 이라면 첫 번째, 두 번째 qubit의 값이 다르다는 의미이다. 따라서 2개의 observable을 측정하여 얻은 outcome을 사용하여 어떤 qubit에 bit error가 발생하였는지를 추측하고 오류 정정을 수행할 수 있다.



(a) Three-qubit repetition code



(b) Three-qubit phase flip code

Figure 4.2: Three-qubit encoding circuits

⁴여기서는 single bit flip error만을 커버한다.

⁵두 상태가 유사할수록 fidelity가 증가한다.

⁶ $Z_1 Z_2 \triangleq +(|00\rangle\langle 00| + |11\rangle\langle 11|)_{12} \otimes I_3) - (|01\rangle\langle 01| + |10\rangle\langle 10|)_{12} \otimes I_3)$

4.3.3 The three qubit phase flip code

이번에는 phase flip error; Z error에 대해 생각해보자. Z error의 정정 역시 X error의 정정과 동일한 방식을 도입하여 해결할 수 있다. Computational basis(X-basis) 대신에 Z-basis를 사용하여 state를 인코딩하며 syndrome measurement set을 사용하여 오류를 감지하고 오류가 발생한 qubit에 Z gate를 가함으로서 phase flip error를 쉽게 해결할 수 있다.

$$|0_L\rangle \equiv |+++ \rangle, \quad |1_L\rangle \equiv |--- \rangle.$$

4.3.4 The Shor code

지금까지 소개한 방식은 특정한 type의 single-qubit error만 해결할 수 있는 방법이다. 그렇다면, 다양한 single qubit error를 한번에 정정할 수 있는 방법은 없을까? Shor code가 바로 *arbitrary single qubit error*를 모두 정정할 수 있는 방법이다. Shor code는 three qubit phase flip code와 bit flip code를 함께 적용한 간단한 방식을 사용한다.

1. Encode the qubit using *phase flip code*:

$$|0\rangle \rightarrow |+++ \rangle, \quad |1\rangle \rightarrow |--- \rangle$$

2. Encoding each of the qubits *bit flip code*:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \rightarrow \frac{|000\rangle + |111\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \rightarrow \frac{|000\rangle - |111\rangle}{\sqrt{2}}$$

따라서 위의 인코딩 과정을 거치면 1개의 logical qubit은 **9개**의 physical qubit으로 구현된다.

$$\begin{aligned} |0\rangle \rightarrow |0_L\rangle &\triangleq \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \\ |1\rangle \rightarrow |1_L\rangle &\triangleq \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}} \end{aligned}$$

이제 Shor code가 어떻게 single qubit error를 정정할 수 있는지 살펴보자. 만약 q_1 에 bit flip error가 발생했다면 observable Z_1Z_2, Z_2Z_3 의 측정 결과로부터 q_1 에 오류가 발생했다는 것을 알 수 있고 X gate를 가하여 오류를 해결할 수 있다. 이처럼 Shor code는 bit flip error를 해결할 수 있다. 다음으로 q_1 에 phase flip error가 발생했다고 하자. Phase flip은 computational basis에 대해서 qubit의 부호를 반전시키므로 $|000\rangle + |111\rangle$ 이었던 첫 번째 블록이 $|000\rangle - |111\rangle$ 으로 바뀌게 된다. 이 오류를 감지하기 위해서 우리는 각 블록($(q_1q_2q_3)$)의 부호를 비교할 수 있도록 syndrome measurement를 사용할 수 있다. 측정 결과로 첫 번째 블록과 두 번째 블록의 부호가 다르고 두 번째 블록과 세 번째 블록의 부호가 동일하다는 정보를 알게 되면, 첫 번째 블록 안의 어떤 qubit에서 phase flip이 발생했다고 추정할 수 있기 때문에 첫 번째 블록에 속한 qubit에 Z gate를 가하 phase flip error도 해결할 수 있다. Phase flip error를 감지하기 위해 사용하는 syndrome measurement는 다음과 같다.

$$X_1X_2X_3X_4X_5X_6, \quad X_4X_5X_6X_7X_8X_9$$

그렇다면 q_1 에 bit flip error와 phase flip error가 동시에 발생했을 때는 어떻게 될까? 이 경우에는 단순히 bit flip error를 정정하는 단계와 phase flip error를 정정하는 단계를 모두 수행하여 문제를 해결할 수 있다.

이제 마지막으로, Shor code가 *arbitrary single qubit error*를 정정할 수 있음을 보이고자 한다. 흥미롭게도, arbitrary single qubit error를 정정하는 과정 또한, 앞에서 설명했던 bit flip error와 phase flip error를 정정하는 두 방법을 적용하는 것으로 쉽게 해결할 수 있다. 이는 단일 큐비트에서 발생할 수 있는 연속적인 오류를 정해진 discrete subset of errors (i.e., X, Z, XZ)을 정정함으로써 모두 해결할 수 있다는 것을 의미한다. 이것이 왜 가능한 걸까? 첫 번째 qubit에 arbitrary single qubit error가 발생했다고 하자. 이 오류는 Kraus operator $\{E_i\}$ 를 사용하여 다음과 같이 표현될 수 있다.

$$\mathcal{E}(|\psi\rangle\langle\psi|) = \sum_i E_i |\psi\rangle\langle\psi| E_i^\dagger$$

전체 오류 대신, 합의 각 항 $E_i|\psi\rangle\langle\psi|E_i^\dagger$ 에 대한 정정을 생각해보자. 우리는 어떤 single qubit operator도 다음과 같이 Pauli operator의 linear combination으로 표현할 수 있다는 사실을 알고 있다.⁷

$$E_i = e_{i0}I + e_{i1}X_1 + e_{i2}Z_1 + e_{i3}X_1Z_1$$

⁷ $Y = XZ$

따라서 이 operator가 pure state $|\psi\rangle$ 에 작용한다면, 다음 4개의 항의 superposition 상태가 될 것이다.

$$|\psi\rangle, \quad X_1|\psi\rangle, \quad Z_1|\psi\rangle, \quad X_1Z_1|\psi\rangle$$

Shor code에서 사용하는 syndrome measurement는 I, X, Z, XZ 가 가해졌을 때 얻을 수 있는 error들의 subspace로 projection시키는 역할을 수행하기 때문에, syndrome measurement를 가하면 error state들의 superposition state가 봉괴해서 하나의 state가 된다. 각각의 상태는 Shor code의 오류 정정 과정을 사용할 수 있으므로(apply X, Z, XZ), 최종적으로 오류가 정정된 state로 변하게 된다. 이것이 바로 Shor code가 arbitrary single qubit error를 정정할 수 있는 이유이다.

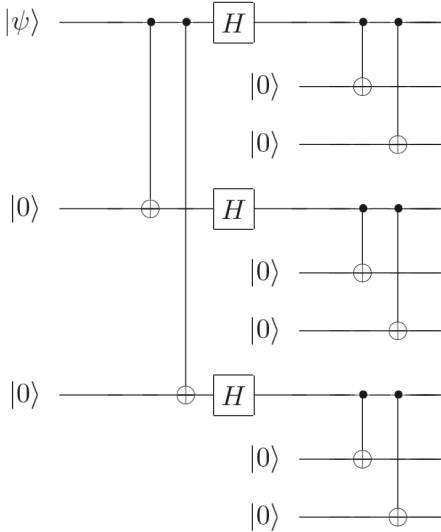


Figure 4.3: Shor code circuit

Lecture 20

4.4 Theory of QEC

25 Nov. 10:30

이전 섹션에서 우리는 Shor code를 통하여 arbitrary single qubit error를 정정할 수 있음을 보였다. 그렇다면 여러 bit에 동시에 error가 발생하는 경우에도 오류 정정을 수행할 수 있을까? 더 나아가, Quantum error correction(QEC)와 관련된 일반적인 이론을 정립할 수 있을까? 이 섹션에서는 QEC를 다루기 위해 사용하는 기초적인 틀에 대해서 소개하고, QEC가 가능하기 위한 조건에 대해서 살펴보자 한다.

기본적인 아이디어는 Shor code에서 도입한 개념을 일반화 하는 것이다. 양자 상태(i.e., physical qubit)는 unitary operator를 통해서 오류 정정 코드(i.e., logical qubit)로 인코딩된다. 오류 정정 코드는 기존 양자 상태보다 더 큰 Hilbert space의 subspace C 로 정의된다.⁸ C 에 대한 projector는 P 로 나타낸다. 예를 들어, three qubit bit flip code의 projector는 다음과 같다.

$$P = |000\rangle\langle 000| + |111\rangle\langle 111|$$

인코딩 이후 코드는 noisy channel을 통과하면서 noise에 영향을 받게 되고 이후 syndrome measurement를 수행한다. Syndrome measurement를 수행하여 어떤 종류의 오류가 발생하였는지 확인하여 *error syndrome*을 결정한다. Error syndrome을 확인한 뒤 recovery operation이 수행되어 다시 original state로 복원된다.

이러한 아이디어를 시각적으로 보여주는 것이 바로 Fig. 4.4이다. 왼쪽에 있는 정육면체가 바로 original state의 subspace를 의미한다. 이 vector space안에 있는 original state에 서로 다른 종류의 error가 발생하면 오른쪽의 그림처럼 A_1, A_2, \dots, A_6 의 서로 다른 vector space로 vector space가 변하게 된다. 이때 (B)는 error subspace들이 서로 disjoint하기 때문에 noise channel을 통과한 후 state가 어떤 subspace에 위치하는지를 판단한 뒤 쉽게 오류 정정이 가능하다. 그러나, (A)의 경우에는 서로 다른 error subspace간에 겹치는 부분이 발생하기 때문에 모호성이 존재하게 된다.

⁸예를 들어, 3-qubit repetition code에서 오류 정정 코드가 사용하는 vector space C 는 $\text{span}\{|000\rangle, |111\rangle\}$

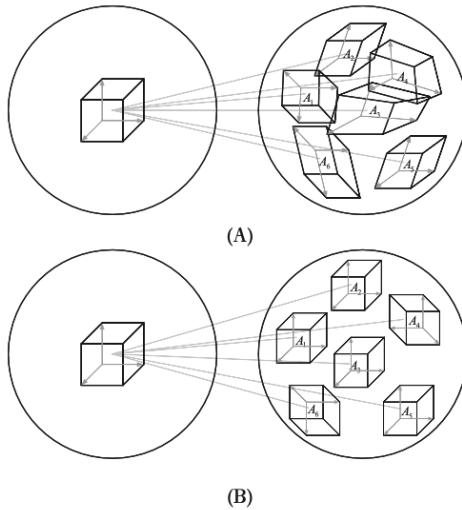


Figure 4.4: Code space

따라서 QEC가 가능하려 각각의 *error syndrome*⁹ original Hilbert space의 **undeformed** and **orthogonal** subspace를 나타내야한다.

QEC의 general theory를 정립하기 위해서는 noise와 오류 정정 과정에 대해서 최대한 적은 가정을 해야 한다. 예를 들어 Shor code가 사용했던 가정들인 QEC가 detection & recovery의 phase로 이루어지고 error 가 특정한 qubit에만 발생하며 error 발생 확률 p 는 충분히 작다와 같은 가정을 하지 않는다. 대신에 우리는 다음의 2가지 가정만을 사용한다.

- Quantum noise는 quantum operation \mathcal{E} 로 표현된다.
- Complete error-correction 과정은 quantum operation \mathcal{R} 로 표현된다. (*error-correction operation*)

위의 정의를 이용하면, QEC를 위한 조건을 다음과 같이 서술할 수 있다:

Note. To satisfy the QEC conditions, any state ρ whose support lies in the code C , must fulfill the following requirement:

$$(\mathcal{R} \circ \mathcal{E})(\rho) \propto \rho$$

Equality 대신에 proportionality를 사용한 이유는 error channel⁹ trace-non-preserving operation일 수도 있기 때문이다.⁹

4.4.1 QEC conditions

QEC condition은 주어진 error-correcting code가 특정한 noise channel \mathcal{E} 에 대해 오류 정정을 수행할 수 있는지 확인하기 위한 equation들의 집합이다. QEC condition을 이용하면 다양한 종류의 error-correcting code를 만들 수 있다.

Theorem 4.4.1 (Quantum error-correction conditions). C 를 quantum code, P 를 C 에 대한 projector, 그리고 \mathcal{E} 가 Kraus operator $\{E_i\}$ 를 갖는 quantum operation이라고 하자. 이 \mathcal{E} 에 대해 오류 정정을 수행할 수 있는 \mathcal{R} 이 존재하기 위한 필요충분조건은:

$$PE_i^\dagger E_j P = \alpha_{ij} P. \quad (4.3)$$

이때, $\alpha_{ij} \in \mathbb{C}$ 는 Hermitian matrix의 원소이다.

Proof. Theorem 4.4.1을 증명하기 위해서는 necessary condition과 sufficient condition 두 가지 방향을 모두 증명해야한다. 먼저 sufficiently; Equation (4.3)이 성립하면 \mathcal{R} 이 존재한다는 것을 보이자.

⁹ \mathcal{E} 는 P condition만을 \mathcal{R} 은 CPTP condition을 모두 만족한다고 가정

$\{E_i\}$ 를 QEC condition을 만족하는 operation elements들의 집합이라고 하자. α 가 Hermitian matrix이기 때문에 diagonalization $d = u^\dagger \alpha u$ 를 수행할 수 있다. Diagonalization으로 얻은 unitary u 를 사용하여 새로운 operation $F_k \triangleq \sum_i u_{ik} E_i$ 를 정의한다. Theorem. 3.3.1에 의하여 F_k 는 E_k 와 동일한 quantum channel \mathcal{E} 를 나타내는 Kraus operator이다. 따라서 Eq. (4.3)에 E_k 대신 F_k 를 대입하면 다음을 유도할 수 있다. 이는 단순화된 QEC condition으로 생각할 수 있다.

$$PF_k^\dagger F_l P = \sum_{ij} u_{ki}^\dagger u_{jl} P E_i^\dagger E_j P = \sum_{ij} u_{ki}^\dagger u_{jl} \alpha_{ij} P = d_{kl} P.$$

이제 polar decomposition을 수행하여 syndrome measurement를 정의할 수 있다. $F_k P$ 에 대한 polar decomposition은 어떤 unitary U_k 에 대해 $F_k P = U_k \sqrt{P F_k^\dagger F_k P} = \sqrt{d_{kk}} U_k P$ 로 표현할 수 있고 이로부터 다음 관계식을 얻는다. 즉, F_k 는 coding subspace에 대하여 회전 연산^{blue}을 수행하고 $\sqrt{d_{kk}}$ 만큼 scaling한 효과를 나타낸다.

$$F_k P = \sqrt{d_{kk}} U_k P$$

따라서 F_k 의 효과를 projector $P_k \triangleq U_k P U_k^\dagger = F_k P U_k^\dagger / \sqrt{d_{kk}}$ 로 나타낼 수 있다. 이때, $k \neq l$ 일 때, projector들이 서로 orthogonal하다는 것을 알 수 있다:

$$U_l P_k = P_l^\dagger P_k = \frac{U_l P F_l^\dagger F_k P U_k^\dagger}{d_{ll} D_{kk}} = 0.$$

즉, QEC condition으로부터 우리는 서로 orthogonal한 subspace로 error syndrome을 정의할 수 있다. 이제 이렇게 정의된 syndrome으로부터 syndrome measurement를 수행해서 측정한 뒤, 복구를 위해 U_k^\dagger 를 적용하기만 하면된다.^{blue} 이러한 detection-recovery step으로부터 다음의 quantum operation \mathcal{R} 을 정의할 수 있다.

1. Apply syndrome measurement P_k to the system.
2. Apply recovery operation U_k^\dagger to the system.

$$\mathcal{R}(\sigma) = \sum_k U_k^\dagger P_k \sigma P_k U_k$$

이제 이렇게 정의한 quantum operation \mathcal{R} 이 정말로 오류를 정정하는지 임의의 quantum state ρ 에 대해 적용해보자. 다음의 관계를 사용하면

- Eq. (4.4): ρ 가 이미 code space에 존재하기 때문에 projector P 를 곱해도 변하지 않음
- Eq. (4.5): P_k 의 definition을 대입
- Eq. (4.6): $U_k^\dagger U_k = I$, $F_k^\dagger F_l$ 은 $\{F_k\}$ 가 orthogonal이므로 δ_{kl}

$$U_k^\dagger P_k F_l \sqrt{\rho} = U_k^\dagger P_k^\dagger F_l P \sqrt{\rho} \quad (4.4)$$

$$= \frac{U_k^\dagger U_k P F_k^\dagger F_l P \sqrt{\rho}}{\sqrt{d_{kk}}} \quad (4.5)$$

$$= \delta_{kl} \sqrt{d_{kk}} P \sqrt{\rho} = \delta_{kl} \sqrt{d_{kk}} \sqrt{\rho} \quad (4.6)$$

\mathcal{R} 을 적용한 결과가 다음과 같이 원래의 state로 복원된다는 것을 수 있다. (F_l : noise, P_k : detect, U_k^\dagger : recovery)

$$\mathcal{R}(\mathcal{E}(\rho)) = \sum_{kl} U_k^\dagger P_k F_l \rho F_l^\dagger P_k U_k = \sum_{kl} \delta_{kl} d_{kk} \rho \propto \rho$$

이제 반대로 necessity; error-correction operation \mathcal{R} 이 QEC condition을 만족함을 보이자. $\{E_k\}$ 가 operator elements $\{R_j\}$ 를 갖는 quantum error-correction operation \mathcal{R} 을 사용하여 완벽한 오류 정정이 가능한 error operator들의 집합이라고 하자. Code space로 ρ 를 변환하는 quantum operation을 $\mathcal{E}_C(\rho) \triangleq \mathcal{E}(P \rho P)$ 로 정의하자. 그렇다면 \mathcal{R} 은 완벽한 오류 정정을 수행할 수 있기 때문에 $\forall \rho$ 에 대해서 다음의 관계를 만족한다.

$$\mathcal{R}(\mathcal{E}_C(\rho)) \propto P \rho P$$

Linearity에 의하면, 비례상수는 ρ 와 독립적인 어떤 constant c 이다. 따라서 이를 이용하여 위의 식을 operator elements들로 다시 나타내면 다음과 같다.

$$\sum_{ij} R_j E_i P \rho P E_i^\dagger R_j^\dagger = c P \rho P$$

즉, 위의 등식이 의미하는 바는 $\{R_j E_i\}$ operation elements를 갖는 어떤 quantum operation은 단일 operation element $\sqrt{c}P$ 를 가지는 operation과 동일하다는 의미이다. Theorem. 3.3.1에 의하면, 다음을 만족하는 complex number c_{ki} 가 존재한다는 것을 알 수 있다.

$$R_k E_i P = c_{ki} P$$

따라서 $P E_i^\dagger R_k^\dagger = c_{ki}^* P$ 역시 만족하며, 이를 이용하면 operation element에 대해 다음 관계식을 얻는다.

$$P E_i^\dagger R_k^\dagger R_k E_j P = c_{ki}^* c_{kj} P$$

R 은 trace-preserving operation^a으로 completeness relation을 만족하기에 다음을 얻는다. (이때 α_{ij} 는 Hermitian matrix^b이다.)

$$\sum_k P E_i^\dagger R_k^\dagger R_k E_j P = P E_i^\dagger E_j P = \alpha_{ij} P, \quad \alpha_{ij} \equiv \sum_k c_{ki}^* c_{kj}$$

따라서 완벽한 오류 정정을 수행할 수 있는 system은 QEC condition을 만족한다. 이로써 두 가지 방향의 증명이 끝났다. ■

^aunitary

^bsyndrome projector로 measurement set을 만들기 위해서 만약 현재 가지고 있는 projector만으로 completeness relation이 성립되지 않는다면 필요한 경우 projector의 개수를 늘릴 수 있다.

4.4.2 Discretization of the errors

QEC condition은 우리에게 quantum error correction에 대해서 많은 정보를 제공하지만, 특정한 하나의 \mathcal{E} 에 대한 정리이다. 그렇다면 이를 일반화 하는 방법은 없을까? 만약 일반화가 가능하여 전체 클래스의 noise로부터 quantum state를 보호할 수 있다면 유용할 것이다.

Theorem 4.4.2. C 가 quantum code^a이고 \mathcal{R} 은 operation elements $\{E_i\}$ 를 갖는 noise process \mathcal{E} 에 대해 QEC condition을 만족하는 error-correction operation^b이라고 하자. \mathcal{F} 가 E_i 의 linear combination으로 만들어지는 operation elements $\{F_i\}$ 를 갖는 noise process라고 하자. ^a 그러면 error-correction operation \mathcal{R} 은 code C 에 대한 noise process \mathcal{F} 의 오류도 정정할 수 있다.

^ai.e., $F_j = \sum_i m_{ji} E_i$ for some matrix m_{ji} of complex numbers

Proof. See Appendix A. (from [NC01]) ■

위의 Theorem을 이용하면, code C 에 대해서 error-correction operator \mathcal{R} 로 정정할 수 있는 error process \mathcal{E} 대신에 error operators $\{E_i\}$ 에 대해 논의할 수 있다. 다음의 QEC condition을 만족하는 error operator들의 linear combination으로 표현되는 operation elements를 가지는 어떠한 임의의 noise process^b도 \mathcal{R} 을 사용하여 완벽한 오류 정정이 가능하다!

$$P E_i E_j^\dagger P = \alpha_{ij} P.$$

예를 들어, \mathcal{E} 를 single qubit에 작용하는 error process라고 하자. \mathcal{E} 의 operation elements는 Pauli operator들의 linear combination으로 표현할 수 있다. 따라서, Pauli operator로 인한 error를 정정할 수 있는 Shor code가 모든 arbitrary single qubit error를 정정할 수 있던 것이다.

$$P \sigma_i^1 \sigma_j^1 P = \alpha_{ij} P$$

4.4.3 Independent error models

이제 single qubit error에서 더 나아가 multiple qubit error를 다루어보자. 먼저, 우리는 각각의 qubit에 error가 서로 independent하게 발생한다는 가정을 할 것이다. 직관적으로, 만약 noise process가 서로 다른 qubit에 독립적으로 발생한다면 noise의 발생 확률이 충분히 작다면 QEC는 fidelity를 향상시킬 것이라 기대할

수 있다. 이를 위해, 어떤 single qubit error도 정정할 수 있는 QEC code를 depolarizing channel에 대해 사용하는 경우를 생각해보자. Depolarizing channel은 다음과 같은 noise process이다.

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$$

먼저 encoding을 수행하지 않는 경우, fidelity는 다음과 같다.

$$F(\rho, \mathcal{E}(\rho)) \geq \sqrt{1 - 2p/3}$$

Single qubit을 n qubit quantum code로 인코딩 할 때, noise process가 independent하다는 가정을 사용하면 n qubit에 대한 noise channel의 동작은 다음과 같다.¹⁰

$$\mathcal{E}^{\otimes n}(\rho) = \underbrace{(1-p)^n \rho}_{\text{no error}} + \underbrace{\sum_{j=1}^n \sum_{k=1}^3 (1-p)^{n-1} \frac{p}{3} \sigma_k^j \rho \sigma_k^j}_{\text{single qubit error}} + \dots$$

가정에 의해, \mathcal{R} 을 수행하면 single qubit error를 정정할 수 있기 때문에 다음 상태로 변한다.

$$(\mathcal{R} \circ \mathcal{E}^{\otimes n})(\rho) = [(1-p)^n + n(1-p)^{n-1}p] \rho + \dots$$

따라서 이를 대입하면, encoding을 수행했을 때 얻을 수 있는 fidelity는 다음과 같다. 따라서 p 가 충분히 작을 때 fidelity가 향상됨을 알 수 있다. 여기서 한 가지 중요한 점은 encoding에 사용하는 qubit의 개수 n 이 증가할 수록 fidelity가 더 향상된다는 것이다.

$$F \geq \sqrt{(1-p)^{n-1}(1-p+np)} = 1 - \frac{\binom{n}{2}}{2} p^2 + O(p^3)$$

4.4.4 Degenerate codes

Shor code와 같이 quantum error correction code는 classical code와 유사하게 동작하는 것처럼 보인다. 그러나 quantum error에서만 나타나는 독특한 성질을 갖는 *degenerate code* class가 존재한다. Shor code를 예시로 들어보자. Z_1 error와 Z_2 error는 code word에 서로 동일한 작용을 하기 때문에, 같은 error syndrome에 대응된다. 이러한 현상을 degenerate라고 한다. 이와 같이 서로 다른 오류가 동일한 subspace에 mapping하는 경우는 classical error에서는 발생하지 않기 때문에, classical error correction code를 기반으로 설계된 QEC가 잘 동작하지 않을 수 있다. 그러나 이는 quantum code의 잠재성을 보여주기도 한다. 각 error들이 orthogonal한 subspace에 mapping될 필요가 없기에 더 많은 정보를 담을 수 있기 때문이다.

4.5 Constructing quantum codes

지금부터는 우리가 앞에서 다루었던 QEC에 대한 다양한 이론을 기반으로 실제 quantum code의 예시를 살펴보고자 한다. 여기서 소개하려는 quantum code인 CSS code는 classical linear code를 기반으로 만들어졌다.

4.5.1 Classical linear codes

Linear $[n, k]$ code C 는 k bit information을 n bit code word로 인코딩하는 방법이다. 이때 C 는 generator matrix G 로부터 만들어진다. G 는 $n \times k$ 의 크기를 가지며 G 는 0 또는 1의 원소만을 가진다. (\mathbb{Z}_2) 이후 주어진 k bit message x 에 대해서 matrix G 를 가하여 n bit code word $y = Gx$ 를 얻는다.¹¹

예를 들어, 앞에서 소개했던 3-bit repetition code는 다음과 같은 generator matrix로 만들어진다.

$$G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

이 matrix는 1 bit information을 3 bit codeword로 각각 다음과 같이 인코딩한다.

$$G[0] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad G[1] = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

¹⁰ σ_k^j 는 j 번째 qubit에 대한 pauli operator k 를 의미한다.

¹¹ 이때 연산은 modulo 2에 대해 수행된다.

G 로 인해서 만들어지는 codeword들이 span하는 code space C 를 정의하기 위해서는 G 의 column vector가 linearly independent해야한다. 이 조건이 지켜져야만 모든 k bit message x 를 unique하게 인코딩할 수 있기 때문이다. 그 외에는 G 의 설계에 대해 어떠한 제한도 없다.

일반적인 error-correction code와 달리 linear code만이 가지는 장점으로는 간결성이다. Linear code가 아닌 어떤 function f 를 사용하여 encoding을 한다면, 가능한 모든 message인 2^k 에 대해서 모두 $f(x)$ 의 값을 설정해주어야한다. 그러나 matrix 표현을 사용하면 $n \times k$ 개의 element만으로 2^k 개의 encoding 방법을 정의할 수 있다.

다음으로 parity check matrix H 를 정의하자. $n - k \times n$ matrix H 는 0 과 1의 원소로 채워져 있다. 이 parity check matrix에 대해 $[n, k]$ code는 G 로 만들어지는 n bit codeword y 에 대해 다음을 만족해야한다.

$$Hy = 0$$

즉, 정상적으로 생성되어 error가 발생하지 않은 codeword는 parity check matrix를 가했을 때 항상 0 벡터를 얻는다. 반대로 말하자면, parity check matrix를 가했을 때 0 벡터가 아닌 값을 얻는다면 codeword에 오류가 발생했음을 알 수 있다!

G, H 는 다음의 과정을 거쳐 서로 변환할 수 있다.

- $G \rightarrow H$: G 의 column과 orthogonal한 linearly independent vectors $n - k$ 개를 선택하여 그 transpose를 H 의 row로 선택한다.
- $H \rightarrow G$: H 를 span하는 linearly independent vectors k 개를 선택하여 G 의 column vector가 되도록 만든다.

$$G = (y_1 \quad y_2 \quad \cdots \quad y_k), \quad H = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_{n-k}^T \end{pmatrix}$$

H 를 이용하면 matrix의 linearity 덕분에 error를 찾는 것이 매우 간단해진다. y 에 error e 가 더해져서 생긴 word를 $y' = y + e$ 라고 할 때, parity check matrix에 대해 다음이 성립한다.

$$Hy' = Hy + He = He.$$

따라서 He ; error syndrome을 확인하여 어떤 error가 발생했는지를 추정할 수 있다. 만약, single bit error라면 각각의 error $e_1 = (1 \ 0 \ \cdots \ 0)^T, e_2 = (0 \ 1 \ \cdots \ 0)^T, \dots, e_n = (0 \ 0 \ \cdots \ 1)^T$ 에 대해서 error syndrome He_i 를 미리 계산하여 기록해두고 우리가 얻은 He 의 값이 어떤 값과 동일한지를 비교함으로서 어떤 bit에 error가 발생했는지를 추정할 수 있다.

반면 조금 더 일반적으로 classical linear code를 사용하여 error-correction을 수행하기 위해 *distance*라는 개념을 도입한다. x, y 가 각각 n bit codeword일 때, 두 codeword간의 Hamming distance $D(x, y)$ 는 x 와 y 에서 서로 다른 bit의 개수로 정의된다. 또한 codeword x 의 Hamming weight $wt(x)$ 는 x 에서 0이 아닌 bit의 개수로 정의된다. (i.e., $wt(x) = D(x, 0)$.) 이를 이용하면 Hamming distance를 두 codeword의 modulo 2 addition의 Hamming weight로 생각할 수 있다. (i.e., $D(x, y) = wt(x + y)$) Distance를 도입하면, 직관적으로 $y' = y + e$ 에 대해서 $D(z, y')$ 를 최소화하 codeword z 를 찾아서 그 값으로 정정하면 효과적으로 오류 정정을 수행할 수 있으리라고 기대할 수 있다 ($p < 1/2$). 단, 이를 수행하기 위해서는 가능한 모든 codeword 2^k 에 대해서 distance를 계산해야하므로 classical liner code에서는 이러한 optimization process를 더 효과적으로 수행할 수 있는 code를 설계하기 위해 노력하고 있다.

반면, code C 에 대한 distance는 그 subspace에 속하는 codeword간의 최소 distance로 정의된다.

$$d(C) = \min_{x, y \in C, x \neq y} D(x, y)$$

이때, 앞에서 논의한 바에 따르면 $D(x, y) = wt(x + y)$ 이며, linear code에서는 $x + y \in C^\perp$ 으로 $x \leftarrow x + y$ 로 두면 다음과 같이 더 간단하게 나타낼 수 있다.

$$d(C) = \min_{x \in C, x \neq 0} wt(x)$$

이렇게 정의되는 $d(C)$ 는 우리가 설계한 linear code의 성질을 나타낸다. $d \triangleq d(C)$ 라고 표현하면, C 는 $[n, k, d]$ code이다. d 가 중요한 이유는 $d \geq 2t + 1$ 이면, t -bit error가 발생했어도 단순히 $d(y, y') \geq t$ 인 유일한 y 를 찾아서 정정할 수 있기 때문에 t bit error를 정정할 수 있다.

마지막으로 liner code C 와 관계된 몇 가지 중요한 code들을 소개하고 마무리하고자 한다. C 가 generator matrix G 와 parity check matrix H 를 갖는 $[n, k]$ code라고 하면 다음을 정의할 수 있다.

- C^\perp : dual of C C^\perp 는 C 의 모든 codeword에 대해 직교하는 codeword들로 구성된다. C^\perp 의 generator matrix와 parity check matrix는 H^T 와 G^T 이다.
- $C \subset C^\perp$: weakly self-dual code
- $C = C^\perp$: self-dual code

Lecture 21

4.5.2 Calderbank-Shor-Steane codes

2 Dec. 10:30

지금부터 소개하고자 하는 code는 CSS(Calderbank-Shor-Steane) code이다. 이 code는 QEC code class의 큰 subset을 이루며, stabilizer code의 subclass이다.¹² C_1, C_2 가 각각 $[n, k_1], [n, k_2]$ 인 classical linear code라고 하자. 이때, 두 코드가 $C_2 \subset C_1$ 의 관계를 만족하며 C_1, C_2 가 모두 t bit error를 정정할 수 있는 code라고 가정하자. 이를 이용하면 t qubit error를 해결할 수 있는 $[n, k_1 - k_2]$ quantum code를 구성할 수 있다.

$x \in C_1$ 인 codeword일 때, quantum codeword는 다음과 같이 C_2 의 가능한 모든 codeword들의 superposition으로 정의된다.

$$|x + C_2\rangle \equiv \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x + y\rangle.$$

$x - x' \in C_2$ 인 codeword $x' \in C_1$ 에 대해서, $|x + C_2\rangle = |x' + C_2\rangle$ 라는 것을 쉽게 확인할 수 있다.¹³ 따라서 $|x + C_2\rangle$ 는 오직 C_1/C_2 ¹⁴의 coset에만 종속된다. 이를 이용하면 집합 $x + C_2$ 를 다음과 같이 정의할 수 있다.^{15 16}

$$x + C_2 \triangleq \{x + y : y \in C_2\}$$

더 나아가, x 와 x' 가 각각 C_2 의 서로 다른 coset에 속한다면 $x + y = x' + y'$ 인 $y, y' \in C_2$ 가 존재하지 않는다. 즉, 이 경우 $|x + C_2\rangle$ 와 $|x' + C_2\rangle$ 는 서로 orthonormal state이다. C_2 의 coset의 개수는 $|C_1|/|C_2|$ 이므로 CSS code의 dimension은 $2^{k_1 - k_2}$ 개이다. 따라서 CSS는 $[n, k_1 - k_2]$ quantum code임을 알 수 있다.

이제 이렇게 정의된 CSS code를 사용하여 어떻게 error를 감지할 수 있을지 생각해보자. Theorem. 4.4.2에 의하여 bit flip, phase flip error만 감지하고 정정할 수 있으면 모든 arbitrary error를 해결할 수 있음을 이용하여 bit flip, phase flip을 어떻게 감지하고 정정하는지 소개한다. Bit flip error는 bit flip이 발생한 bit들의 위치에 대응되는 원소의 값만 1인 error vector e_1 로 표현되며 phase flip error도 이와 유사하게 e_2 로 표현하자. 이 표현을 이용하면 $|x + C_2\rangle$ 에 noise가 발생한 state를 다음과 같이 표현할 수 있다.

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y) \cdot e_2} |x + y + e_1\rangle$$

먼저 bit flip error가 발생한 위치를 파악하기 위해서 syndrome을 저장하는 ancilla qubit을 추가한다. 이후, 다음과 같이 주어진 state에 parity matrix를 적용하여 그 결과를 ancilla qubit에 저장하는 연산을 거치면,¹⁷ linearity에 의해 ancilla qubit에는 He_1 가 저장된다.

$$|x + y + e_1\rangle |0\rangle \rightarrow |x + y + e_1\rangle |H(x + y + e_1)\rangle = |x + y + e_1\rangle |He_1\rangle$$

이 연산을 noisy state에 적용하면 다음 결과를 얻는다.

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y) \cdot e_2} |x + y + e_1\rangle |H_1 e_1\rangle$$

따라서 ancilla qubit을 측정하여 어떤 위치에 error가 발생했는지를 파악하고 그 qubit에 X gate를 적용하여 오류를 정정할 수 있으므로 다음과 같은 상태가 된다.¹⁸

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y) \cdot e_2} |x + y\rangle$$

¹²CSS code \subseteq stabilizer code \subseteq QEC code

¹³ $x - x'$ 가 codeword C_2 에 속하기 때문에, 모든 codeword들의 superposition에 따라 정의되는 state에는 C_2 의 원소를 더하거나 빼더라도 여전히 동일한 상태가 된다.

¹⁴Group C_1 의 subgroup C_2 의 모든 left coset들의 집합

¹⁵Group theory에 대한 설명은 Appendix A를 참고하라.

¹⁶See <https://en.wikipedia.org/wiki/Coset>

¹⁷uncomputation을 통해서 이를 수행하는 operator를 만들 수 있다.

¹⁸ancilla qubit 제거

다음으로 parity flip error를 정정하는 방법을 고안하자. 먼저 각 qubit에 Hadamard gate를 적용하여 다음과 같은 상태를 얻는다. ($z' = z + e_2$)

$$\frac{1}{\sqrt{|C_2|2^n}} \sum_{z \in \{0,1\}^n} \sum_{y \in C_2} (-1)^{(x+y) \cdot (e_2+z)} |z\rangle = \frac{1}{\sqrt{|C_2|2^n}} \sum_{z' \in \{0,1\}^n} \sum_{y \in C_2} (-1)^{(x+y) \cdot z'} |z' + e_2\rangle$$

$z' \in C_2^\perp$ 이라고 가정하면, $\sum_{y \in C_2} (-1)^{y \cdot z'} = |C_2|$ 임을 쉽게 알 수 있다. 반대로 $z' \notin C_2^\perp$ 이라면 $\sum_{y \in C_2} (-1)^{y \cdot z'} = 0$ 이다. 따라서 다음과 같이 상태를 재구성할 수 있다.

$$\frac{1}{\sqrt{2^n / |C_2|}} \sum_{z' \in C_2^\perp} (-1)^{x \cdot z'} |z' + e_2\rangle$$

따라서 bit-flip error를 정정하는 것과 동일한 방식을 이용하여 H_2e_2 state를 측정하고 syndrome 정보에 따라 error가 발생한 위치에 Z gate를 가하면 다음과 같이 오류 정정된 상태를 얻는다.

$$\frac{1}{\sqrt{2^n / |C_2|}} \sum_{z' \in C_2^\perp} (-1)^{x \cdot z'} |z'\rangle$$

마지막으로 모든 오류를 정정하고 원래의 상태로 돌아오기 위해, Hadamard gate를 다시 적용하면 다음 상태가 된다.

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x + y\rangle$$

4.5.3 The Steane code

Linear error-correcting codes의 한 종류인 Hamming codes를 정의한다. H 를 $r \geq 2$ 일 때, 길이가 r 인 $2^r - 1$ bit string으로 구성된 matrix라고 하자. 이 parity check matrix는 $[2^r - 1, 2^r - 1 - r]$ linear code; *Hamming code*이다. $r = 3$ 인 $[7, 4]$ Hamming code의 parity check matrix는 다음과 같다.

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (4.7)$$

이 행렬의 각 column vector가 서로 다르기 때문에, 7 dimensional vector에 대해 one-hot encoding 형태로 표현되는 single qubit error는 쉽게 감지하고 정정할 수 있다. 또한, $d(C)$ 의 정의를 이용하면 이 Hamming code의 distance가 3이라는 것을 알 수 있다. (i.e., $[7, 4, 3]$)

이제 Hamming code를 이용하여 CSS code를 만들수 있으며, 이를 *Steane code*라고 부른다. $C_1 \triangleq C$ 그리고 $C_2 \triangleq C^\perp$ 라고 두자. CSS code를 만들기 위해서는 $C_2 \subset C_1$ 인지를 확인해야한다. 정의에 의해 $C_2 = C^\perp$ 이므로 C_2 의 parity check matrix는 C 의 generator matrix와 동일하다:

$$H[C_2] = G[C_1]^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

이를 C_1 의 parity check matrix (see Eq. (4.7))와 비교하면 $H[C_2]$ 의 row vector들의 span은 $H[C_1]$ 의 row vector들의 span을 포함한다. 즉, 만약 $H[C_2]x = 0$ 이라면, $H[C_1]x = 0$ 이므로 $C_2 \subset C_1$ 이다. 더 나아가 $C_2^\perp = (C^\perp)^\perp = C$ 이므로 $C_1 (= C)$ 과 $C_2^\perp (= C)$ 는 모두 single bit error를 정정할 수 있는 $d(C) = 3$ code이다. 또한 $C_1 = [7, 4]$ code이고 $C_2 = [7, 3]$ code이므로 CSS(C_1, C_2)는 single bit error를 정정할 수 있는 $[7, 1]$ ¹⁹ quantum code임을 알 수 있다. Steane code는 $|0_L\rangle \Delta |0 + C_2\rangle$ 로 logical qubit을 인코딩하고 $|1_L\rangle$ 은 C_2 에 속하지 않는 C_1 의 element를 사용하여 인코딩 한다. (e.g., 1111111)

4.6 Stabilizer codes

Stabilizer code는 classical linear code와 비슷한 구조를 가지는 quantum code로 QEC class에서 굉장히 중요한 class의 quantum code이다. Stabilizer code를 이해하기 위해서 먼저 양자역학의 연산을 표현하는 방법인 *stabilizer formalism*을 소개하고자 한다.

¹⁹ 4-3

Note (Quantum Error Correction Sonnet).

We cannot clone, perforce; instead, we split
 Coherence to protect it from that wrong
 That would destroy our valued quantum bit
 And make our computation take too long.

Correct a flip and phase – that will suffice.
 If in our code another error's bred,
 We simply measure it, then God plays dice,
 Collapsing it to X or Y or zed.
 We start with noisy seven, nine, or five

And end with perfect one. To better spot
 Those flaws we must avoid, we first must strive
 To find which ones commute and which do not.

With group and eigenstate, we've learned to fix
 Your quantum errors with our quantum tricks.

by Daniel Gottesman

4.6.1 The stabilizer formalism

Stabilizer formalism을 설명하기 위해서 2-qubit의 bell state를 고려하자.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Bell state는 특정한 product of operator를 가하더라도 상태가 변하지 않는다는 특징을 가진다. (e.g., $X_1X_2|\psi\rangle = |\psi\rangle$, $Z_1Z_2|\psi\rangle = |\psi\rangle$) 이럴 때, $|\psi\rangle$ state가 X_1X_2 와 Z_1Z_2 operator에 의해 stabilize 된다고 한다. 사실 $|\psi\rangle$ 가 X_1X_2 와 Z_1Z_2 에 의해 stabilize되는 유일한 state이다.²⁰

Stabilizer의 기본 아이디어는 대부분의 quantum state들이 state 대신 stabilizer를 사용하여 나타내는 것이 더 쉽다는 것이다! 그 이유에 대해서 살펴보기 위해서 group theory를 도입하려고 한다. n qubit system에 대한 Pauli operator들의 group G_n 이 stabilizer에서 주로 다루는 group이다. Single qubit에 대한 Pauli group은 다음과 같이 표현된다.

$$G_1 \equiv \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}$$

이때, coefficient $\pm 1, \pm i$ 를 포함하는 이유는 Pauli group에서 곱셈이 잘 정의되도록 만들기 위해서이다. n qubit에 대한 Pauli group은 Pauli matrix들의 n -fold tensor products로 정의된다.

이제 Pauli group을 사용하여 stabilizer를 조금 더 정확하게 정의할 수 있다. S 가 G_n 의 subgroup이고 V_S 를 S 에 속하는 모든 원소들에 의해 고정되는 n qubit state들의 집합으로 정의하자. 다시 말해, subgroup S 의 원소에 대해서 stabilize 되는 quantum state들의 vector space를 V_S 라고 한다.

→ V_S is the vector space stabilized by S , and S is said to be the stabilizer of the space V_S , since every element of V_S is stable under the action of elements in S .

Stabilizer formulism의 간단한 예시를 살펴보자. 3 qubit system에 대해 $S \leq G_3$ 을 다음과 같이 선택하자.

$$S = \{I, Z_1Z_2, Z_2Z_3, Z_1Z_3\}$$

먼저 Z_1Z_2 에 의해 stabilize되는 vector space는 $\text{span}\{|000\rangle, |001\rangle, |110\rangle, |111\rangle\}$ 이며, Z_2Z_3 으로 stabilize되는 vector space는 $\text{span}\{|000\rangle, |100\rangle, |011\rangle, |111\rangle\}$ 이다. 따라서 stabilizer S 에 의해 고정되는 vector space는 stabilizer set의 각 element에 대해 모두 고정될 수 있는 state들의 vector space이므로, 두 subspace에서 공통으로 사용되는 vector $|000\rangle, |111\rangle$ 에 대해서 $V_S = \text{span}\{|000\rangle, |111\rangle\}$ 로 정의된다.²¹

이 예시에서 vector space를 결정하기 위해 Z_1Z_2, Z_2Z_3 만 확인한 이유는 이 두 원소가 S 의 generator이기 때문이다.²² (i.e., $S = \langle Z_1Z_2, Z_2Z_3 \rangle$) 즉, S 의 모든 원소에 대해서 고려하지 않고 generator element만을 고려하면 더 간단하게 표현할 수 있다.²³

²⁰without the global phase

²¹ $Z_1Z_3 = Z_1Z_2Z_2Z_3 = Z_1Z_3$ 이므로 고려하지 않아도 된다.

²²generator element들의 곱셈만으로 group의 모든 원소를 생성할 수 있을 때.

²³Group G 에 대한 generator는 최대 $\log(|G|)$ 개의 generator만을 가지기 때문이다.

Pauli group의 임의의 subgroup S 가 모두 stabilizer로 사용될 수 있는 것은 아니다. 다음 2가지 조건을 만족하는 group S 는 stabilizer로 정의될 수 있다:

- $\forall s_i, s_j \in S, [s_i, s_j] = 0$: S 의 원소들은 서로 commute한다.
- $-I \notin S$: $-I$ 는 S 의 원소가 아니다.

만약 non-commute인 Pauli matrix가 포함되어있다면 다음 관계를 이끌어낸다. 이는 자명한 vector space를 나타내기 때문에²⁴ non-trivial vector space에 대한 stabilizer로 사용될 수 없다.

$$|\psi\rangle = MN|\psi\rangle \rightarrow |\psi\rangle = -NM|\psi\rangle = -|\psi\rangle$$

두 번째 조건 역시, $|\psi\rangle = -|\psi\rangle$ 를 만족하는 vector space를 나타내므로 자명한 vector space를 나타낸다.

만약, 특정한 generator g_i 를 제거하면 그 generator로 만들어지는 group은 더 이상 기존의 group과 같지 않으며 더 작은 크기의 group을 만들게 된다. 즉, g_1, \dots, g_l generator들은 모두 서로 독립이어야 한다.

$$\langle g_1, \dots, g_{i-1}, g_{i+1}, \dots, g_l \rangle \neq \langle g_1, \dots, g_l \rangle$$

Generator들이 independent한지를 확인하는 한 가지 간단한 방법은 바로 check matrix이다. $S = \langle g_1, \dots, g_l \rangle$ 일 때, check matrix는 $l \times 2n$ 크기의 행렬이며 행렬의 각 row는 generator g_i 와 관련되어 있다. 행렬의 column은 $[1, n]$ 까지는 그 generator에 Pauli X_k 가 곱해져 있는지를 표현하기 위한 공간이며 $[n+1, 2n]$ 까지는 그 generator에 Pauli Z_k 가 곱해져 있는지를 표현하기 위한 공간이다. 이 2개의 column만 있어도 충분한 이유는 Pauli Y 는 XZ 로 decomposition 할 수 있기 때문이다.

예를 들어, $g_1 = Z_1 X_2 Y_3 Z_5$ 라면 다음과 같이 표현할 수 있다.

$$\begin{array}{c} g_1 \\ g_2 \\ \vdots \\ g_{l-1} \\ g_l \end{array} \left[\begin{array}{ccccc|cccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ \vdots & & \vdots & & & & & & \vdots \\ \vdots & & \vdots & & & & & & \vdots \\ \vdots & & \vdots & & & & & & \vdots \\ \vdots & & \vdots & & & & & & \vdots \end{array} \right]$$

또한 check matrix를 사용하여 두 generator g 와 g' 가 교환 가능한지를 확인하는 방법은 다음 공식을 만족하는 것인지 확인하는 것이다. 이때 $r(g)$ 는 generator g 에 대한 $2n$ 차원 row vector를 의미한다.

$$r(g)\Lambda r(g')^T = 0, \quad \Lambda = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}, \quad (\Lambda \in \mathbb{N}^{2n \times 2n})$$

정리하면, 다음 두 가지 명제를 보일 수 있다.

Proposition 4.6.1. $S = \langle g_1, \dots, g_{n-k} \rangle$ 에서 $-I$ 가 S 의 원소가 아니라고 하자. g_1, \dots, g_l 이 모두 독립이기 위한 필요충분조건은 check matrix의 row가 서로 linearly independent한 경우이다.

Proposition 4.6.2. $S = \langle g_1, \dots, g_{n-k} \rangle$ 가 G_n 에서 $n-k$ 개의 서로 독립이며 교환 가능한 element들에 의해 생성되고 $-I \notin S$ 라고 하자. 그렇다면 V_S 는 2^k dimensional vector space이다.^a

^aStabilizer code의 입장에서 이는 n qubit을 k qubit으로 인코딩 함을 의미한다.

Lecture 22

4.6.2 Transformation of stabilizers

4 Dec. 10:30

앞에서 우리는 vector space V_S 를 나타내기 위해서 stabilizer를 활용했다. 더 나아가, 이번에는 stabilizer를 사용하여 quantum dynamics를 나타낼 수 있음을 보이고자 한다. Group S 에 의해 stabilize되는 vector space V_S 에 대해 unitary operation U 을 가하는 경우를 생각해보자. $|\psi\rangle \in V_S$ 일 때, S 의 어떤 element에 대해서도 다음이 성립한다.

$$\underbrace{U|\psi\rangle}_{|\varphi\rangle} = Ug|\psi\rangle = UgU^\dagger \underbrace{U|\psi\rangle}_{|\varphi\rangle}$$

²⁴ $|\psi\rangle = -|\psi\rangle$ 를 만족하는 vector space는 영벡터만 가능하기 때문이다.

따라서 state $U|\psi\rangle$ 는 UgU^\dagger 에 의해서 stabilize되며, 따라서 vector space UV_S 는 USU^\dagger 에 의해 stabilize된다는 사실을 알 수 있다. USU^\dagger group은 다음과 같이 정의된다.

$$USU^\dagger \equiv \{UgU^\dagger | g \in S\}$$

더 나아가, $S = \langle g_1, \dots, g_l \rangle$ 이라면 unitary operator U 가 stabilizer S 에 미치는 영향을 분석하기 위해 USU^\dagger 의 generator $\langle Ug_1U^\dagger, \dots, Ug_lU^\dagger \rangle$ 만을 계산하면 된다.

이 접근법을 사용하면, 어떤 unitary operator U 에 대해 stabilizer의 generator들의 변화(i.e., UgU^\dagger)가 특히 매력적인 형태를 보인다는 것을 이용할 수 있다. 예를 들어, 단일 Hadamard gate는 다음과 같이 동작한다.

$$HXH^\dagger = Z; \quad HYH^\dagger = -Y; \quad HZH^\dagger = X$$

즉, H 는 Pauli matrix로 정의되는 stabilizer에 적용되어, 그대로 Pauli matrix로 정의되는 stabilizer를 만들 어낸다. 이 관계를 이용하면 어떤 state $|\psi\rangle$ 에 대한 H 의 연산을 수행하는 것이 아니라 $|\psi\rangle$ 을 stabilize하는 operator g 에 대한 H 의 연산을 수행하여 $H|\psi\rangle$ 의 state를 추정할 수 있다. (See Fig. 4.5)

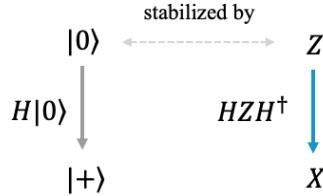


Figure 4.5: state transformation and generator transformation

이 방식이 강력한 이유는 qubit의 개수가 증가할 때 진가를 발휘한다. n qubit에 대한 $H^{\otimes n}$ 연산의 결과를 표현하기 위해서는 2^n 개의 amplitude를 저장해야하지만 이를 generator로 나타내게 되면 n 개의 element만 필요로 한다. ($|0\rangle^{\otimes n}$: stabilized by $\langle Z_1, \dots, Z_n \rangle \rightarrow |+\rangle^{\otimes n}$: stabilized by $\langle X_1, \dots, X_n \rangle$)

H gate 말고도 다양한 unitary gate들이 이러한 관계를 이용할 수 있다. Fig. 4.5에 나와있듯이 CNOT, H, S, P gate들은 모두 pauli group에 가해져서 pauli group을 보존한다. 이러한 역할을 하는 U 의 집합을 G_n 의 normalizer라고 부른다. 그중에서도 특히 Pauli group의 normalizer를 Clifford group이라고 한다.

$$\forall U \in N(G_n), \quad UG_nU^\dagger = G_n$$

다양한 gate들이 Clifford group에 속한다는 것을 알 수 있다. 그렇다면, Clifford group에 속하지 않는 gate가 존재할까? Non-Clifford group에 속하는 2가지 중요한 gate는 $\pi/8$ gate (i.e., T gate) 그리고 Toffoli gate이다. 다음과 같이 T gate를 X gate에 가하면 그 결과는 Pauli gate들의 product로 표현할 수 없다.

$$TXT^\dagger = \frac{X + Y}{\sqrt{2}}$$

그렇다면, Stabilizer formalism을 사용하여 measurement도 표현할 수 있을까? 지금부터는 computational basis에서 measurement를 stabilizer formalism으로 기술하는 방법을 설명하고자 한다. Pauli observable $g \in G_n$ 을 측정하다고 하자. 이때, 편의를 위하여 coefficient -1, $\pm i$ 가 없는 Pauli matrix들의 곱이라고 가정하자.²⁵ 관측을 수행하는 system은 stabilizer $S = \langle g_1, \dots, g_l \rangle$ 를 가지는 $|\psi\rangle$ state에 있다고 하자.²⁶

이러한 상황에서는 다음과 같은 2가지 case가 존재한다.

- g 와 $\forall g' \in \langle g_1, \dots, g_l \rangle$ commute이다.
- g 와 $\exists g' \in \langle g_1, \dots, g_l \rangle$ anti-commute이다. (g_1 anti-commute인 generator라고 하자.)

g 는 Pauli operator들의 product이므로 eigenvalue ± 1 을 가지므로 이 observable을 측정한 outcome은 ± 1 이며, 그 때 사용하는 measurement operator(projector)는 다음과 같다.

$$+1 : \frac{I + g}{2}, \quad -1 : \frac{I - g}{2}$$

²⁵ -1 은 결과를 flip하는 것으로 쉽게 구현할 수 있으며 ± 1 은 observable은 Hermitian이어야 한다는 조건을 위배하기 때문이다.

²⁶ 1dim state: outcome이 2개

Operation	Input	Output
controlled-NOT	X_1	X_1X_2
	X_2	X_2
	Z_1	Z_1
	Z_2	Z_1Z_2
H	X	Z
	Z	X
S	X	Y
	Z	Z
X	X	X
	Z	$-Z$
Y	X	$-X$
	Z	$-Z$
Z	X	$-X$
	Z	Z

Figure 4.6: Stabilizer generators transformation

먼저 첫 번째 case일 때, measurement를 분석해보자. 임의의 generator g_i 에 대해서 g 가 commute하기 때문에 다음이 성립하는데 g_i 는 $g|\psi\rangle$ 를 stabilize하고 $|\psi\rangle$ 는 g 에 의해 stabilize되므로 $g|\psi\rangle \in V_S$ 이다.

$$g_i g |\psi\rangle = gg_i |\psi\rangle = g |\psi\rangle$$

따라서 이를 이용하면, $|\psi\rangle$ 에 대한 observable g 의 관측 확률은 다음과 같다.

$$\begin{aligned} g &: p(1) = 1; \quad p(-1) = 0 \\ -g &: p(1) = 0; \quad p(-1) = 1 \end{aligned}$$

또한 measurement operator가 단순히 state를 V_S 로 projection하는 것이기 때문에 post-measurement state도 변하지 않는다.

그렇다면, 두 번째 case에서는 어떻게 될까? Outcome probability는 다음과 같다.

- Eq. (4.8): $g|\psi\rangle = |\psi\rangle$ 그리고 g, g_1 는 anti-commute임을 이용
- Eq. (4.9): Trace의 cyclic property, $g_1 = g_1^\dagger$ ²⁷임을 이용

$$p(+1) = \text{Tr} \left[\frac{I+g}{2} |\psi\rangle\langle\psi| \right] = \text{Tr} \left[\frac{I+g}{2} g_1 |\psi\rangle\langle\psi| \right] = \text{Tr} \left[g_1 \frac{I-g}{2} |\psi\rangle\langle\psi| \right] \quad (4.8)$$

$$= \text{Tr} \left[\frac{I-g}{2} |\psi\rangle\langle\psi| g_1 \right] = \text{Tr} \left[\frac{I-g}{2} |\psi\rangle\langle\psi| g_1^\dagger \right] \quad (4.9)$$

$$= \text{Tr} \left[\frac{I-g}{2} |\psi\rangle\langle\psi| \right] = p(-1), \quad (4.10)$$

$p(+1) = p(-1) = 1$ 으로 $p(+1) = p(-1) = 1/2$ 라고 추정할 수 있다. Post-measurement state는 각각 다음과 같고 각 상태는 새로운 stabilizer $\langle g, g_2, \dots, g_l \rangle, \langle -g, g_2, \dots, g_l \rangle$ 을 가진다.

$$+1 : \frac{I+g}{\sqrt{2}} |\psi\rangle, \quad -1 : \frac{I-g}{\sqrt{2}} |\psi\rangle$$

4.6.3 Gottesman-Knill theorem

Stabilizer formalism을 사용할 때, quantum computer의 연산 능력에 대해 한 가지 놀라운 정리를 얻을 수 있다. 이 정리는 quantum computation class의 강력함이 얼마나 애매한지를 보여준다 (...)

²⁷ $g_1^2 = \pm I$ 인데 $-I \notin S$ 이므로

Theorem 4.6.1 (Gottesman-Knill theorem). 다음의 과정을 거치는 quantum circuit은 **classical computer**로 **polynomial time**²⁸에 **simulation** 할 수 있다.

1. prepare $|0\rangle^{\otimes n}$ (in computational basis)
2. apply unitary gates in *Clifford group*
3. measure in computational basis (include conditioned on the outcome of measurement)

Lecture 23

4.6.4 Stabilizer code constructions

9 Dec. 10:30

이제 다시 우리의 원래 주제로 돌아와서 stabilizer formalism을 사용하여 어떻게 QEC를 만들지 생각해보자.

$[n, k]$ stabilizer code $C(S)$ 는 G_n 의 subgroup S 에 의해 stabilize되는 vector space V_S 로 정의된다.²⁸ (i.e., $C(S) \triangleq V_S$) 이때, S 는 앞에서 가정했던 것처럼 (1) $-I \notin S$ 이고 (2) $n - k$ 개의 linearly independent and commute인 generator $S = \langle g_1, \dots, g_{n-k} \rangle$ 를 가진다는 것이다.

그렇다면 stabilizer code는 어떤 *logical basis state*를 사용할까? Vector space V_S 의 2^k 개의 orthonormal basis vector를 선택하여 사용할 수도 있지만, 실제로는 이보다 더 체계적인 방식을 사용한다.

1. $\bar{Z}_1, \dots, \bar{Z}_k \in G_n$ ²⁹을 선택하여 $g_1, \dots, g_{n-k}, \bar{Z}_1, \dots, \bar{Z}_k$ 가 서로 linearly independent, commute한 원소를 갖는 집합을 생성하도록 한다.
2. 따라서 logical qubit $|x_1, \dots, x_k\rangle_L$ 은 다음의 stabilizer를 갖는 state로 정의할 수 있다.

$$\langle g_1, \dots, g_{n-k}, (-1)^{x_1} \bar{Z}_1, \dots, (-1)^{x_k} \bar{Z}_k \rangle.$$

그럼 stabilizer code에서 발생하는 error는 무엇일까? $E \in G_n$ 에 대해 3가지 종류의 에러가 존재한다.

- E anti-commute with an element in generator : anti-commute이므로 에러가 발생하면 codeword의 vector space와 orthogonal한 subspace로 mapping되기 때문에 error가 발생했음을 확인하기도 쉽고 쉽게 해결할 수 있다. (e.g., if E is anti-commute with g_1 , then $Eg_1E^\dagger = -g_1$ ³⁰)
- $E \in S$: stabilizer의 성질덕분에 $E|\psi\rangle = |\psi\rangle$ 이므로 오류가 발생하지 않는다.
- E commute with all element in generator, but $E \notin S$: 에러가 발생하면 codeword의 vector space 공간에 존재하는 다른 codeword로 mapping되기 때문에 가장 까다로운 오류이다.

S 의 모든 원소와 commute인 원소 $E \in G_n$ 들의 집합을 *centralizer*, $Z(S)$ 라고 한다. 이때, $Z(S) - S$ 로 표현되는 집합이 공집합이 아니기 때문에 3번째 type에 해당하는 에러가 발생할 수 있는 것이다. (이때 stabilizer에 대한 centralizer는 stabilizer의 normalizer, $N(S)$ 와 동일하다. Normalizer는 S 의 모든 원소에 대해 $EgE^\dagger \in S$ 가 되도록 하는 원소 $E \in G_n$ 들의 집합이다.)

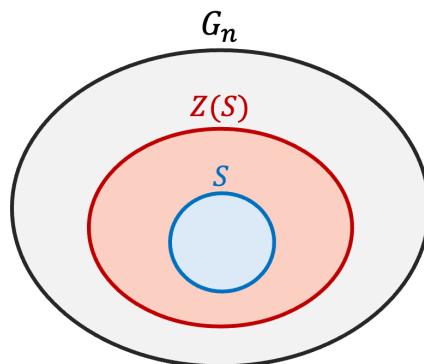


Figure 4.7: Error classes

²⁸ S 는 $-I \notin S$ 이고 $n - k$ 개의 independent, commute인 generator를 가진다고 가정.

²⁹ \bar{Z}_k 는 logical gate로 k 번째 logical qubit으로 인코딩된 physical qubit에 모두 Z gate를 적용한다.

³⁰ observable g_1 을 측정한 결과가 -1 이므로 오류가 발생했다는 것을 쉽게 알 수 있다.

이러한 다양한 에러들의 종류에 대해, stabilizer code에서 QEC condition은 다음과 같이 정리된다.

Theorem 4.6.2 (Error correction conditions for stabilizer codes). Let S be the stabilizer for a stabilizer code $C(S)$. Suppose $\{E_j\}$ is a set of operators in G_n such that $E_j^\dagger E_k \notin N(S) - S$ for all j and k . Then $\{E_j\}$ is a correctable set of errors for the code $C(S)$.

→ 즉, 집합 $N(S) - S$ 에 속하지 않는 원소는 stabilizer code를 사용하여 정정될 수 있다.

Proof. 증명에 앞서, 먼저 original QEC condition이 다음과 같이 정의됨을 기억하자.

Remark.

$$PE_j^\dagger E_k P = \alpha_{ij} P \quad (4.11)$$

where α_{ij} is a Hermitian matrix and P is a projector onto code space (e.g., $C(S)$)

우리가 관찰한 stabilizer code 에러의 3가지 종류에 따르면, $E_j^\dagger E_k \in N(S) - S$ 에 속하는 에러는 (1) $E_j^\dagger E_k \in S$ 또는 (2) $E_j^\dagger D_k \in G_n - N(S)$ 이다.

먼저 첫 번째 에러에 대해 P 는 code space에 대한 projector으로 S 의 원소에 의해 stabilize된다. 따라서 E_j^\dagger, E_k 가 S 의 원소이므로 $PE_j^\dagger E_k P = P$ 의 관계를 성립한다. $\alpha_{ij} = 1$ 일 때 Eq. (4.11)를 만족한다. 반면, 두 번째 에러는 normalizer $N(S)$ 의 원소가 아니기 때문에 적어도 하나의 generator와는 anti-commute하다. 이 원소를 g_1 이라고 하자. $\langle g_1, \dots, g_{n-k} \rangle$ 가 S 의 generator set이면, projector P 를 다음과 같이 쓸 수 있다.

$$P = \frac{\prod_{l=1}^{n-k} (I + g_l)}{2^{n-k}}$$

$E_j^\dagger g_1 E_k = -g_1$ 임을 이용하면 $E_j^\dagger E_k$ 를 가한 상태는 다음과 같다.

$$E_j^\dagger E_k P = (I - g_1) E_j^\dagger E_k \frac{\prod_{l=2}^{n-k} (I + g_l)}{2^{n-k}} \quad (4.12)$$

이때, Eq. (4.12)는 $(I - g_1)$ 이 곱해져있고 반대로 P 는 $(I + g_1)$ 이 곱해져있는데, $(I + g_1)(I - g_1) = I - g_1^2 = 0$ ^a이므로 $PE_j^\dagger E_k P = 0$ 의 관계를 성립한다. $\alpha_{ij} = 0$ 일 때 Eq. (4.11)를 만족한다.

따라서 어떤 종류의 에러이든지 $E_j^\dagger E_k \in G_n - N(S)$ 라면 QEC condition을 만족하기 때문에 오류 정정이 가능하다. ■

^aPauli group은 $g_i^2 = I$ 이므로?

그렇다면, (정정가능한) 에러가 발생했을 때, stabilizer code는 이를 어떻게 감지하고 정정하는가? $\langle g_1, \dots, g_{n-k} \rangle$ 가 S 의 generator이고 $\{E_j\}$ 가 정정가능한 에러들의 집합이라고 하자. 오류감지를 위하여 generator; observable을 차례대로 측정한 결과로 error syndrome $\beta_1, \beta_2, \dots, \beta_{n-k}$ 를 얻을 수 있다. 에러 E_j 가 발생했을 경우 error syndrome은 $+1, \dots, \beta_j, \dots, 1$ 로 표현된다. 이는 $E_j g_l E_j^\dagger = \beta_l g_l$ 를 만족한다. 그럼 해당 error syndrome에 대한 복구연산자 E_j^\dagger 를 적용하여 정정할 수 있다.

이때 동일한 error syndrome을 가지는 서로 다른 에러 $E_j, E_{j'}$ 가 존재할 수도 있다. 하지만 이는 에러를 발생시킨 연산자가 무엇이든 관계없이 E_j^\dagger 나 $E_{j'}^\dagger$ 를 통하여 정정할 수 있다. (by $E_j^\dagger E_{j'} P E_{j'}^\dagger E_j = P$)

Stabilizer code에서도 classical code *distance*를 정의하는 방법과 거의 유사한 방식으로 distance를 정의할 수 있다. 에러 $E \in G_n$ 의 가중치(weight)는 non-trivial한 Pauli matrix의 수로 정의된다.³¹ 예를 들어, $X_1 Z_4 Y_8$ 의 가중치는 3이다. $C(S)$ 의 거리는 $N(S) - S$ 에 있는 에러에 대한 minimum distance로 정의된다. $C(S)$ 가 $[n, k, d]$ 코드이고 $d \geq 2t + 1$ 이라면, 최대 t 큐비트에서 발생하는 오류를 정정할 수 있다.

4.6.5 Examples

The three qubit bit flip code

앞에서 다루었던 repetition-three qubit code에 대한 예시를 살펴보자. 이 코드는 $|000\rangle, |111\rangle$ 을 basis state로 가지기 때문에 logical code space는 stabilizer $S = \langle Z_1 Z_2, Z_2 Z_3 \rangle$ 을 가진다. Bit-flip error 집합 $\{I, X_1, X_2, X_3\}$ 에 대해, three qubit bit flip code는 두 원소끼리 가능한 모든 product³²로 정의되는 집합의

³¹자명한 Pauli matrix는 I 를 이야기한다. I 는 가해지더라도 오류정정을 할 필요가 없기 때문이다.

³²e.g., $X_1 X_2 \in \{E_j\}$, but $X_1 X_2 X_3 \notin \{E_j\}$

에러를 모두 정정할 수 있다. 이는 $\{E_j\}$ 에 속한 모든 오류가 S 의 generator인 Z_1Z_2, Z_2Z_3 둘 중 하나와 반드시 anti-commute하기 때문이다.

The nine qubit Shor code

Shor code의 code space에 대한 8개의 generator는 Fig. 4.8에 나와있다. 이를 이용하면 발생할 수 있는 다양한 에러에 대해서, generator의 원소와 anti-commute인지 아닌지를 확인하는 것만으로 이 code가 해당 오류를 정정할 수 있는지를 분석할 수 있다. (e.g., X_1Y_4 는 Z_1Z_2 와 anti-commute이므로 정정가능한 오류)

Name	Operator
g_1	$ZZI III III$
g_2	$I ZZIII III$
g_3	$I I I ZZI III$
g_4	$I III I ZZI III$
g_5	$I III III ZZI$
g_6	$I III III I ZZ$
g_7	$XXXXXX I II$
g_8	$I II XXXXXX$
Z	$XXXXXXXXX$
\bar{X}	$ZZZZZZZZZ$

Figure 4.8: generator for Shor code (and logical Z and X)

The five qubit code

에러 정정을 위해서 encoding을 사용할 때, single qubit error를 감지하고 정정할 수 있게하는 logical qubit의 가장 작은 크기는 **5 qubit**이다. Five qubit code에 대한 stabilizer 역시 Fig. 4.9에 나와있으며, 모든 single qubit error가 five qubit code의 generator와 anti-commute한다는 사실을 쉽게 파악할 수 있다.

Name	Operator
g_1	$XZZXI$
g_2	$IXZZX$
g_3	$XIXZZ$
g_4	$ZXI XZ$
\bar{Z}	$ZZZZZ$
\bar{X}	$XXXXX$

Figure 4.9: generator for five qubit code (and logical Z and X)

CSS codes and the seven qubit code

앞에서 다루었듯이, CSS code는 $C_2 \subset C_1$ 인 classical linear code C_1, C_2 를 이용한다. 이 code에 대해서 다음과 같은 check matrix를 정의할 수 있다.³³ 이 행렬이 stabilizer code를 정의한다는 것을 보이기 위해서는 $H(C_2^\perp)H(C_1)^T = 0$ 을 만족한다는 것을 확인해야하지만 $C_2 \subset C_1$ 관계 때문에 $H(C_2^\perp)H(C_1)^T = [H(C_1)G(C_2)]^T 0$ 가 된다.

$$\begin{pmatrix} H(C_2^\perp) & 0 \\ 0 & H(C_1) \end{pmatrix}$$

CSS code의 한 종류인 Steane code의 경우에는 다음과 같은 logical Z, X gate를 사용한다.

$$\bar{Z} \equiv Z_1Z_2 \cdots Z_7; \quad \bar{X} \equiv X_1X_2 \cdots X_7.$$

³³각 행이 generator, 각 column이 qubit을 의미하며 왼쪽은 Z, 오른쪽은 X를 나타내기 위해 사용된다.

4.6.6 Quantum circuits for encoding, decoding, and correction

Stabilizer code의 특징 중 하나는 encoding, decoding, error correction 절차를 잘 구성할 수 있다는 점이다. $[n, k]$ stabilizer code $C(S)$ 가 generator $\langle g_1, \dots, g_k \rangle$ 와 logical gate $\bar{Z}_1, \dots, \bar{Z}_k$ 를 가질 때, 전체 과정을 나타내보자.

1. (encode) logical $|0\rangle_L$ state를 준비 : physical initial state $|0\rangle^{\otimes n}$ 에 대해서 $n - k$ 개의 generator, 그리고 k 개의 logical operator를 측정하면 측정결과에 따라, 적절한 Pauli operator를 하하여 logical init state를 만들 수 있다. 정정하기전의 state는 다음과 같은 stabilizer를 가진다.

$$\langle \pm g_1, \dots, \pm g_{n-k}, \pm Z_1, \dots, \pm Z_k \rangle$$

2. (decode) QC에서는 logical state를 측정한 결과로부터 기존의 값(decode)을 알 수 있기 때문에, decoding 과정이 별도로 필요하지 않다.

3. (measurement) generator나 logical Pauli gate; observable을 측정하기 위해서 Fig. 4.10와 같은 회로를 구성한다. 이는 M 에 대한 eigenvalue를 측정한다는 의미로 해석할 수 있다. Pauli group G_n 에 속한 원소들은 모두 ± 1 의 eigenvalue를 가진다.

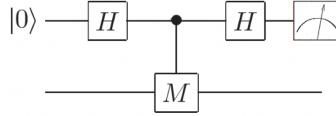


Figure 4.10: Measurement observable

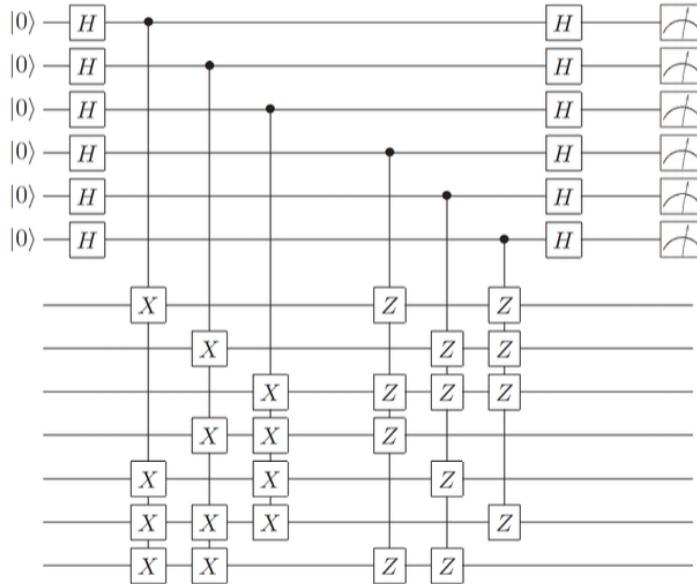


Figure 4.11: Stean code measurement

Lecture 24

4.7 Fault-tolerant quantum computation

11 Dec. 10:30

지금까지 우리는 QEC에서 진행되는 절차가 모두 *noiseless*라는 이상적인 가정에서 QEC가 가능하다는 것을 주장했었다. 그러나, 실제 quantum computer는 여전히 noise가 존재하고 이로인하여 QEC 과정에서도 오류가 발생할 수 있다. 그렇다면 실제 device에서는 QEC가 불가능할까? 놀랍게도, 1개의 gate를 가할 때 오류가 발생할 확률인 p 가 특정 threshold보다 작은 경우 arbitrary long quantum circuit도 작은 오류확률을 가지며 효율적으로 수행할 수 있다는 사실이 알려져있다.

4.7.1 Fault-tolerance: the big picture

Fault-tolerance quantum computation은 각 연산을 적용할 때마다 에러가 발생할 가능성을 고려하면서도 안정적인 연산을 수행할 수 있도록 하는 방법이다. 이 섹션에서는 양자 컴퓨터에서 *error propagation*, 그리고 *error accumulate*가 왜 fault-tolerance에 문제가 되는지에 대해서 설명한 뒤, fault-tolerant가 어떻게 구현되는지를 예시로 들고자 한다. 마지막으로 **concatenation**을 통해 어떤 arbitrary long quantum circuit도 신뢰성있게 실행할 수 있다는 threshold theorem에 대해서 설명하려고 한다.

Fundamental issues

Fault-tolerance computation을 위해서 (1) QEC code로 인코딩된 logical qubit을 이용하며, (2) 각 quantum *logical gate*가 적용될 때마다 error correction을 실시하는 단순한 아이디어를 생각해볼 수 있다. (See Fig. 4.12) 그러나 이런 방법으로도 충분하지 않다. 다음의 2가지 문제점이 존재한다.

- Logical (encoded) gate는 *error propagation*을 야기시킬 수 있다. 예를 들어, CNOT gate라면 control qubit가 가지고 있던 오류가 target qubit에도 전달된다.
- QEC에서 수행하는 error correction procedure 자체가 faulty 할 수 있다.

따라서 Fault-tolerant quantum computation을 위해서는, encoded gate를 잘 설계해야 하낟.

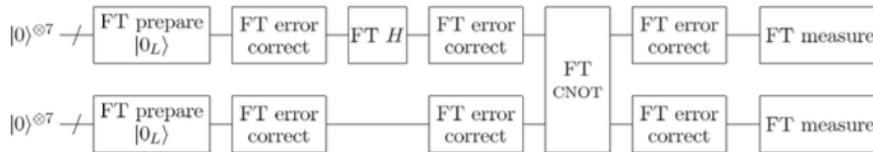


Figure 4.12: Fault-tolerance circuit

Fault-tolerant operations: definitions

Fault-tolerance computation을 구현하기에 앞서, Fault-tolerance의 기준이 무엇인지를 정의하고자 한다.

Definition 4.7.1 (Fault-tolerance). If only one component in the procedure fails then the failure cases at most one error in each encoded block of qubits output from the procedure.

즉, 어떤 procedure³⁴가 fault-tolerance하다고 이야기하기 위해서는 그 procedure를 이루는 각 component(e.g., gate, measurement, state preparation, ...)에 오류가 발생하였더라도 procedure가 끝났을 때 output logical qubit안에는 최대 1개의 에러만이 발생해야 한다. 이러한 single qubit error는 우리가 앞에서 소개했던 매커니즘을 이용하여 충분히 오류정정을 수행할 수 있다.

특히 fault-tolerant measurement도 이러한 특성을 만족해야하는데, 만약 procedure를 이루는 각 component의 에러 확률중에서 가장 높은 값이 p 일 때, 측정 결과는 $O(p^2)$ 의 오류확률을 가져야 한다.

본 강의에서는 단순화를 위하여 Pauli-error $\{I, X, Y, Z\}$ 에 대해서만 고려한다. 예를 들어, X_1 오류가 있는 상태에서 CNOT을 하면 $C(X)X_1I = C(X)X_1C(X)^\dagger C(X) = X_1X_2C(X)$ 가 된다. 즉, CNOT이 올바르게 적용되었지만 bit flip error가 다른 qubit으로 전파되었다.

Example: fault-tolerant CNOT

본격적으로 fault-tolerant CNOT을 구현하는 procedure에 대해서 분석해보자. Fig. 4.13에서 표현한 것처럼 FT CNOT은 4가지의 component를 가진다. 각 component에 오류가 발생하더라도 encoded qubit안에서 2개 이상의 오류가 발생할 확률이 $O(p^2)$ 가 된다는 것을 보이려고 한다.

- 각 logical qubit에 단일 에러가 존재하는 경우 : CNOT을 통과하면서 첫 번째 logical qubit에 존재하던 에러가 두 번째 logical qubit으로 전파되면서 두 번째 logical에 두 개의 오류를 유발할 수 있다. 만약 이전 단계의 회로들이 모두 FT로 설계되었다면, 각 logical qubit에 단일 에러가 발생할 확률은 $c_0 p$ 이하이다.³⁵ 두 logical qubit에 에러가 독립적으로 발생한다고 가정하면, CNOT 통과이후 단일 logical qubit에서 2개 이상의 에러를 가질 확률은 $c_0^2 p^2$ 이하이다.

³⁴e.g., encoded gate를 구현하는 procedure

³⁵ c_0 은 오류가 발생할 수 있는 위치를 나타내는 상수

- 하나의 logical qubit에 단일 에러가 존재하고 CNOT에서 추가로 단일 에러가 발생하는 경우 : 이 에러 이벤트의 발생확률은 $c_1 p^2$ 이다.
- CNOT에서 2개의 에러가 발생하는 경우
- CNOT에서 단일 에러가 발생하고 syndrome measurement를 수행하면서 추가로 단일 에러가 발생하는 경우
- etc ...

FT CNOT procedure에서 발생할 수 있는 모든 error event에 대해서 logical qubit에 두 개 이상의 에러가 발생할 확률은 모두 cp^2 이하 이다. 따라서, FT CNOT을 구현하는 회로의 각 구성 요소가 p 확률로 에러를 발생시킬 수 있어도 FT CNOT은 $1 - cp^2$ 확률로 성공한다.

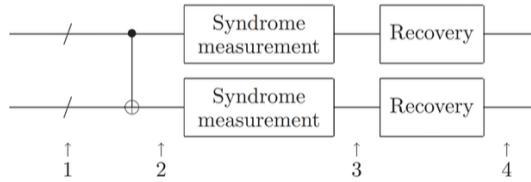


Figure 4.13: Fault-tolerant CNOT procedure

Concatenated codes and the threshold theorem

앞에서 보인 FT CNOT은 기존의 성공확률에 비해서 더 높은 성공확률을 보이기는 하지만, 여전히 그렇게 까지 강력해보이지는 않는다. 이를 위해서 한 가지 강력한 아이디어를 도입하고자 한다.

아이디어는 FT procedure를 **recursively** 반복하는 것이다. 예를 들어, Fig. 4.14에서 사용하는 two-level concatenation을 분석해보자. 회로를 이루는 component에 대해 physical error probability가 p 일 때, 첫 번째 encoding을 통해서 logical error rate cp^2 을 얻을 수 있다. 그 다음으로 logical qubit $|0\rangle_{L_1}$ 을 physical qubit으로 생각하고 두 번째 encoding을 도입할 수 있다. (e.g., $|0\rangle_{L_2} = |0_{L_1} 0_{L_1} 0_{L_1}\rangle$) 따라서 error rate $p_1 \Delta cp^2$ 에 대하여, 두 번째 encoding을 통해서 얻을 수 있는 error rate는 $p_2 \triangleq c(cp^2)^2 = c^3 p^4$ 이다. 이를 반복하면 k -level concatenation을 통해서 $(cp)^{2^k}/c$ error rate를 얻을 수 있다. k -level concatenation을 위해 필요한 cost(size of simulating circuit)는 original circuit size의 d^k 배이다. 이때, d 는 FT procedure를 위해 필요한 maximum number of operations을 의미한다.

Original circuit size가 problem size n 에 대해 polynomial size (i.e., $p(n)$)를 가질 때, target accuracy ϵ 을 달성하기 위해서 필요한 반복횟수 k 는 얼마일까? Target accuracy를 달성하기 위해서는, 회로를 이루는 각각의 gate에 대한 target accuracy가 $\epsilon/p(n)$ 이라는 것을 쉽게 알 수 있다. 따라서, k -concatenation FT gate에 대해서 다음을 만족해서 target accuracy를 달성할 수 있을 것이다.

$$\frac{(cp)^{2^k}}{c} \leq \frac{\epsilon}{p(n)}.$$

이때, physical error $p < p_{\text{thresh}} \Delta 1/c$ 의 조건을 만족하면 $k \rightarrow \infty$ 에 따라서 우리가 원하는 임의의 정확도를 얻을 수 있다.³⁶ 그렇다면, target accuracy를 달성하기 위해 필요한 circuit의 크기는 얼마일까? d^k 에 위의 관계식으로부터 얻은 k 를 대입하면 다음을 얻을 수 있고

$$d^k = O(\text{poly}(\log p(n)/\epsilon)),$$

원본 회로의 크기가 $p(n)$ 이므로 FT simulate circuit의 size는 다음과 같다.

$$O(\text{poly}(\log p(n)/\epsilon)p(n))$$

이를 *threshold theorem*이라고 한다.

4.7.2 Fault-tolerant quantum logic gate

우리는 $\{H, S, CNOT, T\}$ gate가 universal gate set이라는 사실을 이미 알고 있다. 따라서 Fault-tolerant $H, S, CNOT$ 그리고 T gate를 구현할 수 있으면³⁷ Fault-tolerant gate procedure를 구축할 수 있고 이를 이용하여 Fault-tolerant computation을 수행할 수 있다.

³⁶ 만약 $p = p_{\text{thresh}} \Delta 1/c$ 라면 k 가 증가하더라도 $(cp)^{2^k}/c$ 가 constant가 되기 때문에 target error를 달성할 수 없다.

³⁷ 여기서 이야기하는 FT는 Fig. 4.13와 같은 전체 과정이 아니라 이 과정의 구성요소인, FT CNOT을 의미한다.

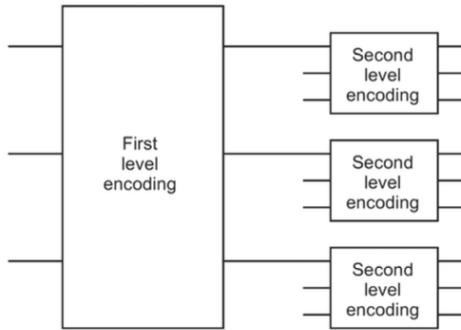


Figure 4.14: Concatenation

Normalizer operations

$\{H, S, CNOT\}$ gate는 Clifford group에 속하는 게이트이기 때문에 쉽게 구현할 수 있다. Stean code에서 사용하는 logical gate는 다음과 같이 정의된다.

$$\bar{Z} = Z_1 Z_2 \cdots Z_7; \quad \bar{X} = X_1 X_2 \cdots X_7.$$

즉, logical gate를 구현하기 위해서는 단순히 logical qubit을 이루는 각각의 physical qubit에 *single qubit gate*를 가하면 된다. 각 gate는 하나의 qubit에만 작용하기 때문에, error propagation이 일어나지 않는다. 따라서 logical gate를 구현하기 위해서 physical operation을 각 qubit에 서로 **individually** 적용하면 그 logical gate는 fault-tolerant 하다는 사실을 알 수 있다. (*transversality property*)

Theorem 4.7.1 (Easten-Knill theorem). No quantum error correcting code can transversely implement a universal gate set.

마찬가지로 CNOT gate 역시 transversality property를 가지도록 구현할 수 있다. Control qubit으로 사용되는 logical qubit에 있는 모든 physical qubit과 target qubit으로 사용되는 logical qubit에 있는 모든 physical block을 CNOT gate로 연결한다. 이 연산은 동일한 block안에서 서로 연결되지 않기 때문에 독립적으로 동작하기 때문에 fault-tolerance하다.

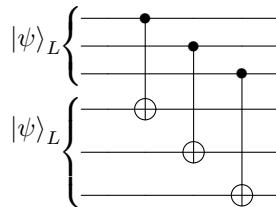


Figure 4.15: Logical CNOT gate

Fault-tolerent $\pi/8$ gate

반면, T gate의 구현은 앞선 방식들과는 다른 방법을 따른다. 이 과정은 (1) 특정한 magic state $|\Theta\rangle$ 로 state preparation을 수행한 뒤, 앞에서 이미 구현한 FT normalizer operation들을 이용하여 회로를 구성한다.

- initial state : $|\theta\rangle |\psi\rangle$. FT T gate는 첫 번째 qubit에 $T|\psi\rangle$ 를 저장하는 방식으로 동작한다.

$$|\Theta\rangle = \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}.$$

- apply FT CNOT : 다음을 얻는다.

$$\frac{1}{\sqrt{2}} \left[|0\rangle(a|0\rangle + b|1\rangle) + e^{i\pi/4}|1\rangle(a|1\rangle + b|0\rangle) \right] = \frac{1}{\sqrt{2}} \left[(a|0\rangle + be^{i\pi/4}|1\rangle)|0\rangle + (b|0\rangle + ae^{i\pi/4}|1\rangle)|1\rangle \right]$$

3. measure second qubit : 두 번째 qubit의 측정 결과가 0일 때는 post-state가 우리가 원하는 $T|\psi\rangle$ 이므로 측정 결과가 1일 때만 추가적인 정정을 필요로 한다.

4. if outcome is 1, then apply SX

$$SX = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

SX 를 가하면 우리가 원하는 state를 얻을 수 있다.

$$a|0\rangle + be^{i\pi/4}|1\rangle.$$

그렇다면, initial state $|\Theta\rangle$ 는 어떻게 생성할 수 있을까? Measurement outcome이 그 observable에 해당하는 eigenvalue를 반환하는 것이라는 걸 기억하자. $|0\rangle$ 가 +1을 eigenvalue로 가지는 Z 의 eigenvector이고 $|\Theta\rangle$ 가 +1을 eigenvalue로 가지는 $e^{-i\pi/4}SX$ 의 eigenvector라는 사실을 이용하자.

따라서 Clifford group의 특징을 이용하면, $Z \rightarrow HZH^\dagger \rightarrow THZH^\dagger T^\dagger = e^{-i\pi/4}SX$ 이므로 이를 이용하여 $|0\rangle \rightarrow H|0\rangle \rightarrow T|0\rangle = |\Theta\rangle$ 임을 알 수 있다. 따라서 H, T gate를 차례대로 가함으로서 Θ 상태를 만들 수 있다. 단, 이 과정은 **faulty**를 허용하지 않는 방법이다.³⁸ 따라서 대신 $e^{-i\pi/4}SX$ 로 $|0\rangle$ 을 측정함으로서 Θ state를 준비할 수 있다. 한 가지 주의해야 할 점은 HT gate를 적용한 뒤 측정결과가 -1이라면, Z gate를 가하여 $|\Theta\rangle$ 로 변환해야 한다.

$$\begin{aligned} e^{-i\pi/4}SX|\Theta'\rangle &= -|\Theta'\rangle, \\ \Rightarrow e^{-i\pi/4}SX(Z|\Theta'\rangle) &= (Z|\Theta'\rangle) = |\Theta\rangle \end{aligned}$$

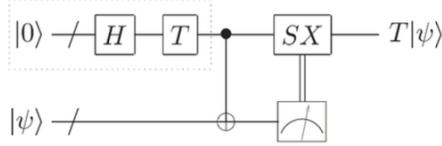


Figure 4.16: Fault-tolerance T gate

Fault-tolerant measurement

그렇다면, 마지막으로 fault-tolerance measurement는 어떻게 구현할 수 있을까? Measurement는 QEC에서 syndrome measurement를 진행할 때나, state preparation을 위해서, 또는 실제 quantum circuit의 마지막 연산 단계에서 결과를 확인하기 위해 사용될 수도 있다. 따라서 FT measurement를 구현하는 것은 quantum computing에서 매우 중요한 단계라고 말할 수 있다.

다음 그림은 일반적은 measurement gate를 나타낸다. 이처럼 measurement 단계에서 controlled operation을 사용하여 QEC에 필요한 정보를 수집하기 때문에 첫 번째 qubit $|0\rangle$ 에서 발생한 에러가 다른 qubit에 모두 전파될 수 있다.

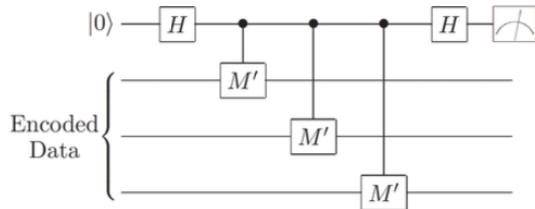


Figure 4.17: Measurement

Three qubit encoded code에서의 fault-tolerance measurement는 Fig. 4.18에 나와있다.

³⁸fault-tolerance T gate를 모르는데 T gate를 사용함

1. prepare : 관측을 위해 사용되는 ancilla qubit을 cat-state $|00\cdots 0\rangle + |11\cdots 1\rangle$ 로 준비시킨다. prepare step은 FT가 아니기 때문에, 앞에서 수행한 회로가 제대로 된 연산을 수행했는지 검증하는 단계를 필요로 한다.³⁹
2. verify : 주어진 ancilla qubit이 cat state인지 확인하기 위해서 모든 qubit들의 combination에 대해 두 qubit이 동일한 parity를 가지는지를 측정한다. ($Z_i Z_j$)
3. controlled-M : M 의 eigenvalue를 측정하기 위해서 ancilla qubit과 data qubit 사이에 1-to-1 controlled M 을 가한다.
4. decode : 측정을 위해 decode를 수행한다.

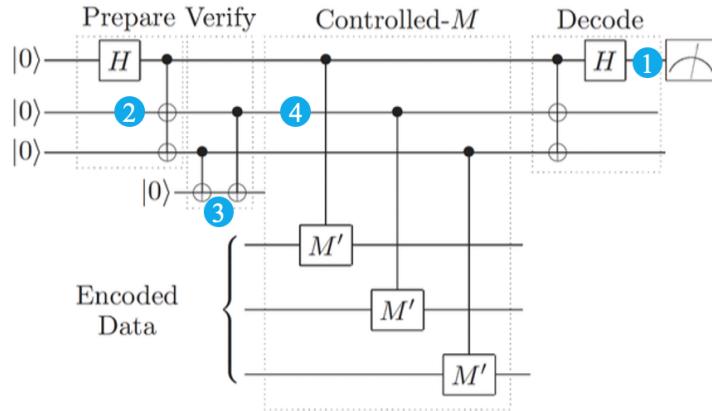


Figure 4.18: Fault-tolerance measurement

이 과정을 따르는 measurement가 왜 fault-tolerance한지 각 component에 오류가 발생했을 경우에 따라 분석해보자.

- (1) decode 이후에 오류가 발생 : 측정 과정을 3번 반복하여, 과반수를 얻은 결과를 택한다. 이 경우 측정이 잘못될 확률은 최대 $O(p^2)$ 이 된다.⁴⁰ (majority vote)
- (2) prepare 과정에서 오류가 발생 : verify 과정에서 오류가 발생했다는 사실을 확인할 수 있으므로 정정할 수 있다.
- (3) verify 과정에서 오류가 발생 : 오류가 있는 ancilla qubit을 이용하면 Z error의 경우 encoded data를 변경시키지 않지만, X, Y 는 encoded data에 오류를 일으킬 수 있다.
→ Z error는 majority vote로 해결할 수 있으며, X, Y 와 같은 single qubit error는 QEC를 통해서 충분히 정정될 수 있다.
- (4) controlled-M을 수행할 때 오류가 발생 : (3)에서 이야기 한 것처럼 오류가 전파되어 encoded data를 손상시킬 수 있지만, 이는 single qubit error이고 QEC로 정정될 수 있다.

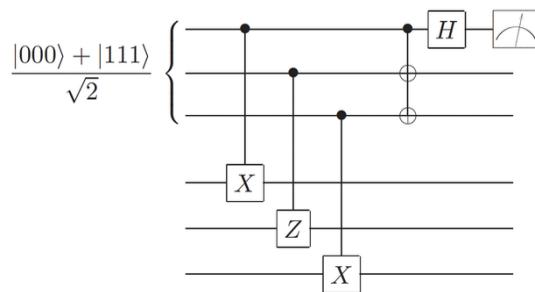


Figure 4.19: Stabilizer measurement

³⁹왜 H^n 으로 안하지...?

⁴⁰encoded data로 연결되지 않기 때문에 데이터 손상은 없다.

Appendix

Appendix A

Details of the proof

- A.1 Optimality of the quantum search algorithm with M solutions
- A.2 Period finding algorithms when r doesn't divide 2^t
- A.3 Choi-Kraus theorem

Bibliography

- [BCK15] Dominic W Berry, Andrew M Childs, and Robin Kothari. “Hamiltonian simulation with nearly optimal dependence on all parameters”. In: *arXiv preprint arXiv:1501.01715* (2015).
- [Chi+21] Andrew M Childs et al. “Theory of trotter error with commutator scaling”. In: *Physical Review X* 11.1 (2021), p. 011020.
- [Dha15] Sayandip Dhara. *Quantum Order Finding and Factorization*. July 2015. doi: [10.13140/RG.2.1.4954.9925](https://doi.org/10.13140/RG.2.1.4954.9925).
- [Haa19] Jeongwan Haah. “Product decomposition of periodic functions in quantum signal processing”. In: *Quantum* 3 (2019), p. 190.
- [HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.
- [NC01] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Vol. 2. Cambridge university press Cambridge, 2001.
- [Np] *NP(complexity)*. URL: [https://en.wikipedia.org/wiki/NP_\(complexity\)](https://en.wikipedia.org/wiki/NP_(complexity)).