PHY365: Quantum Information

QiLin Xue

Fall 2021

Contents

1 Quantum Coins

Consider a quantum coin that can be in a superposition of heads and tails. We can write its state as a vector:

$$|\Psi\rangle = \alpha|H\rangle + \beta|T\rangle \tag{1.1}$$

which lives in the **Hilbert Space.** Inner products of these vectors can be written as

$$\langle\Psi_1|\Psi_2\rangle. \tag{1.2}$$

Born's Rule tells us we can compute the probability of tails to be $|\beta|^2$ and the probability of heads is $|\alpha|^2$. When there are two quantum coins, there can be four combinations of heads and tails, written as:

$$|\Psi\rangle = \alpha|HH\rangle + \beta|HT\rangle + \gamma TH\rangle + \delta|TT\rangle. \tag{1.3}$$

In quantum mechanics, we can construct the following state:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}|HH\rangle + \frac{1}{\sqrt{2}}|TT\rangle, \tag{1.4}$$

which represents **entanglement.** If we measure the first coin, we can instantly know the outcome of the second coin, even if they are lightyears apart.

1.1 Building a Better Computer

How might we use quantum coins to help us build a "better" computer? Before we begin to understand and answer this question, let us understand some key concepts.

First, we can measure **information** as the number of bits (binary digits) that are needed to specify a message. Each bit in a computer requires a physical system that has two possible configurations.

- In semiconductor circuits, we use voltage.
- Magnetization is sometimes also used (i.e. in hard drives).
- Pits in optical storage.
- Paper tape with holes in it

Now let's extend the idea to quantum bits, i.e. **qubits**. Let us use $|0\rangle$ and $|1\rangle$ to represent the two possible states of a quantum coin, and we can write a qubit as

$$|\Psi_1\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.5}$$

which isn't necessarily interesting. If we have two qubits, we can write the state as

$$|\Psi_2\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{1.6}$$

where the following notation are equivalent:

$$|00\rangle = |0\rangle|0\rangle = |0\rangle \otimes |0\rangle \tag{1.7}$$

where $\otimes$ is the **tensor product** of two vectors. To make it easier to write, we can also write it as:

$$|\Psi_2\rangle = \alpha|0_2\rangle + \beta|1_2\rangle + \gamma|2_2\rangle + \delta|3_2\rangle. \tag{1.8}$$

For three qubits, we have

$$|\Psi_3\rangle = \alpha|000\rangle + \beta|001\rangle + \gamma|010\rangle + \delta|011\rangle + \epsilon|100\rangle + \zeta|101\rangle + \eta|110\rangle + \theta|111\rangle. \tag{1.9}$$

Therefore, N qubits will have 2^N possible states. This suggests that quantum memory can get big, fast.

1.1.1 Quantum Parallelism

However, this is not the only difference. Each qubit operation, i.e. $|0\rangle \longleftrightarrow |1\rangle$ affect *all* the probability amplitudes. This also suggests that quantum computers can be extremely efficient.

However, when we make measurements, N qubits only leads to N bits of information. Therefore, even though it is very efficient and quick, there is only a small amount of output.

Example 1: Consider $f : \mathbb{Z}^+ \to \mathbb{R}$ a periodic function that maps $x \in [0, 2^L - 1]$ (i.e. takes in an L bit integer). There is some X such that $f(x + X) = f(x)$ and we wish to find X.

In a classical computer, we would evaluate $f(x)$ for multiple values of x. In general, we would expect around 2^{L-1} calls in the routine.

However, in a quantum computer, we need L qubits to store values of x (i.e. in the. argument register) and L qubits to store the result of $f(x)$ in the function register. Through a series of bit flips, we can create the state

$$|x\rangle|0\cdots 0\rangle \tag{1.10}$$

where the first braket is the input and the second braket is the function register. Then suppose we have a **quantum operation** $\hat{U}_f$ defined such that

$$\hat{U}_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle. \tag{1.11}$$

But if we prepare the initial state of the register not in x, but in a superposition (achieved via a **Hadamard gate**), then we can write:

$$\hat{U}_f \frac{1}{N}\left(\sum_{x=0}^{2^k-1}|x\rangle\right)|0\rangle = \frac{1}{N}\underbrace{\sum_{x=0}^{2^k-1}|x\rangle|f(x)\rangle}_{\text{massively entangled state}} \quad . \tag{1.12}$$

The difference is that all values of $f(x)$ are generated by a single call on $\hat{U}_f$. If we now apply something called the **Quantum Fourier Transform**

$$\hat{U}_{QFT}\sum_x |x\rangle|f(x)\rangle = \frac{1}{N}\sum_x |x\rangle|\tilde{f}(x)\rangle, \tag{1.13}$$

where $\tilde{f}$ is the **fourier transform,** which you will get a discrete graph of vertical bars separated a distance by $\frac{n}{X}$. If we do this a few times, we can extract what X is.

Quantum computers allow us in principle to evaluate periods very efficient. This is a very important problem in **number theory** since period finding helps a great deal in factoring.

Consider coprime n, a and define

$$f(x) = a^x \text{ mod } n. \tag{1.14}$$

This is a periodic function with period r. If we can figure out what r is, then

$$\gcd(a^{r/2} \pm 1, n) \tag{1.15}$$

is a factor of n. This is known as **Shor's Algorithm.**