CMSC 657: Introduction to Quantum Information Processing
Lecture 7

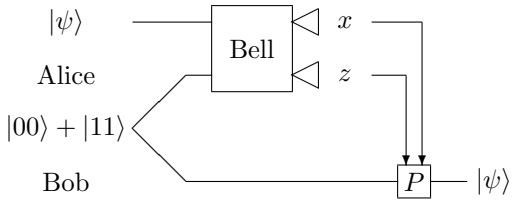
Instructor: Daniel Gottesman

Fall 2024

1 Quantum Protocols

1.1 Quantum Teleportation

We were talking about quantum teleportation.



The initial entangled state is $|\Phi^+\rangle$, one of the Bell states. The Bell states are "maximally" entangled states and are also called *EPR pairs* due to the Einstein-Podalsky-Rosen argument which highlighted the non-local nature of quantum mechanics. Based on the 2-bit outcome of Alice's measurement, Bob does P, which is one of the 4 Pauli matrices.

$$xz = 00, \quad P = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1}$$

$$xz = 01, \quad P = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2}$$

$$xz = 10, \quad P = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

$$xz = 11, \quad P = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{4}$$

Let's work out one of these cases in detail: Let's label the qubits I (the input to be transmitted), A (Alice's qubit from the entangled state), and B (Bob's qubit from the entangled state), and suppose $|\psi\rangle_I = \alpha|0\rangle + \beta|1\rangle$. (Actually this works the same way if Alice's qubit is entangled with a reference system that is uninvolved in the circuit. Bob's output qubit then takes over the entanglement from Alice's qubit.) Initially, before the Bell measurement, the state of all three qubits is

$$|\phi_0\rangle = \frac{\alpha}{\sqrt{2}}|0\rangle_I|0\rangle_A|0\rangle_B + \frac{\alpha}{\sqrt{2}}|0\rangle_I|1\rangle_A|1\rangle_B + \frac{\beta}{\sqrt{2}}|1\rangle_I|0\rangle_A|0\rangle_B + \frac{\beta}{\sqrt{2}}|1\rangle_I|1\rangle_A|1\rangle_B. \tag{5}$$

Here I have put back in the normalization $1/\sqrt{2}$ for the entangled state $|00\rangle + |11\rangle$ since we are about to make a measurement. We may as well assume that $|\psi\rangle$ is normalized.

Suppose the Bell measurement outcome is Π_{01}. We can expand Π_{01} on the I and A qubits as follows:

$$\Pi_{01,IA} = \frac{1}{2}(|00\rangle_{IA}\langle 00|_{IA} \otimes I_B - |00\rangle_{IA}\langle 11|_{IA} \otimes I_B - |11\rangle_{IA}\langle 00|_{IA} \otimes I_B + |11\rangle_{IA}\langle 11|_{IA} \otimes I_B). \quad (6)$$

Acting on the state $|\phi_0\rangle$, the projector eliminates the $|01\rangle_{IA}$ and $|10\rangle_{IA}$ terms, leaving

$$\Pi_{01,IA}|\phi_0\rangle = \frac{\alpha}{2\sqrt{2}}(|00\rangle_{IA}|0\rangle_B - |11\rangle_{IA}|0\rangle_B) + \frac{\beta}{2\sqrt{2}}(-|00\rangle_{IA}|1\rangle_B + |11\rangle_{IA}|1\rangle_B) \quad (7)$$

$$= \frac{1}{2\sqrt{2}}(|00\rangle_{IA} - |11\rangle_{IA}) \otimes (\alpha|0\rangle_B - \beta|1\rangle_B) \quad (8)$$

$$= \frac{1}{2}|\Phi^-\rangle_{IA} \otimes (\alpha|0\rangle_B - \beta|1\rangle_B). \quad (9)$$

Note that the IA qubits are left in the $|\Phi^-\rangle$ state, as they had to be, since we projected on that state. The probability of this outcome is $\|\Pi_{01,IA}|\phi_0\rangle\|^2 = 1/4$, and the residual state of Bob's qubit B (since we no longer need the IA qubits) is

$$|\phi_1\rangle = \alpha|0\rangle_B - \beta|1\rangle_B. \quad (10)$$

Because the outcome was $xz = 01$, Bob then applies Z, which takes $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto -|1\rangle$. The output state is then

$$Z|\phi_1\rangle = \alpha|0\rangle_B + \beta|1\rangle_B = |\psi\rangle_B. \quad (11)$$

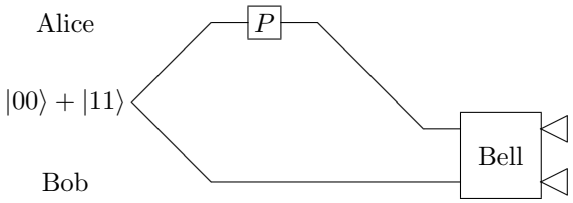
As desired, Bob's qubit now has the original state of the I qubit.

The Bell measurement can be viewed as determining if the I qubit and the A qubit are the same or different in two different bases, and the original Bell state $|\Phi^+\rangle$ ensures that the A and B qubits are themselves the same in those bases. Thus, the Pauli at the end corrects for the basis or bases in which the I qubit differs from the A/B qubits.

Teleportation simplifies the task of quantum communication. It is useful in building quantum networks or in shunting information around in a quantum computer. Teleportation is also a useful building block for other quantum protocols.

1.2 Superdense Coding

Related to quantum teleportation is superdense coding. If Alice and Bob share an EPR pair and the ability to send qubits, they can send 2 classical bits by sending a single qubit.



Alice encodes via I, X, Y, or Z (one of the Paulis):

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (12)$$

She is thus sending one of the four Bell states $|\Phi^\pm\rangle, |\Psi^\pm\rangle$, since these Paulis convert the Bell states into each other.

Bob receives Alice's qubit and measures in the Bell basis, the basis given by these four states. The four possible outcomes correspond to the four possible values Alice could send, so Bob receives 2 classical bits even though only one qubit was transmitted. In the process, their pre-shared entangled state is used up.

Note that the same Bell state is used regardless of what classical bits are transmitted. Superdense coding could be used to increase communication rates by preparing EPR pairs ahead of time, before you know what you want to send.

2 Basic Complexity Theory

2.1 Languages and Complexity

Complexity theory is the study of the difficulty of computational problems. Generally, we would like to classify problems by how long it takes to solve them. However, there are some issues with this program. For starters, what precisely does it mean for a problem to be hard to solve?

Do we set an amount of time T, and problems that take longer than T to solve are hard whereas problems that take less time than that are easy? But this depends too obviously on the speed of the machine used to run the program. We'd like to say a problem is hard or easy in a machine-independent way.

Do we instead count the number of computational steps needed to solve the problem? This is better, but it still depends on the machine a little bit, since it may matter what universal gate set we are using. Moreover, if you think about it more carefully, any given problem has a particular answer y, and there is always a short program that outputs y.

That is: *It does not make sense to define the complexity of a single problem.* Instead, we must define complexity of a *set* of related problems. Moreover, we'd also like to distinguish between problems that take many steps simply because the input size is very large and problems that take many steps because solving them is a very involved process. Therefore, we study infinite sets of problems and study how the number of computational steps needed scales with the size of the input.

Definition 1. *$\{0,1\}^*$ is the set of all bit strings of any length. The size $|x|$ of $x \in \{0,1\}^*$ is the length of the bit string. A* language *is a subset of $\{0,1\}^*$.*

Interpretation: $\{0,1\}^*$ is the set of possible inputs to a yes/no question (a *decision problem*). L is the set of inputs for which the correct answer is "yes."

Example 1 (Primality Testing)**.** *Given an integer x, determine if x is prime.*

In the example of primality testing, L is the set of prime numbers, written in binary.

Definition 2. *An algorithm $A(x)$* decides *language L if $\forall x \in \{0,1\}^*$, $A(x) = 0$ (i.e., no) if $x \notin L$ and $A(x) = 1$ (yes) if $x \in L$. x is called an* instance *of L (a "yes instance" if $x \in L$ and a "no instance" if $x \notin L$). In a* promise problem*, the instance is drawn from a proper subset of $\{0,1\}^*$.*

The following algorithm decides Primality Testing, known as PRIMES: Try dividing y into x $\forall 1 < y < x$. If $\exists y | x$, output 0, otherwise output 1.

Typically, we compute complexity as a function of the size $|x|$ of the instance. When x is a number written in binary, $|x| = \lceil \log_2 x \rceil$. In the case of this primality testing algorithm, we must try dividing a total of x different numbers, and each division takes time about $(\log x)^2$ (using standard long division), so the total complexity is about $x \log^2 x$. Since the input size is $\log x$, the complexity of this algorithm is potentially exponential. Of course, there are some inputs (like even numbers) for which the algorithm returns an answer very quickly, but for other inputs it could take a very long time. There exist much better algorithms than this one.

To give the complexity of deciding a language L, we should use the complexity of the *best* (shortest) algorithm $A(x)$ that decides L. But how exactly do we want to quantify complexity?

We will consider *worst case* complexity, which means that the algorithm $A(x)$ must succeed for all inputs. It may work faster on some inputs than others, and we calculate the run time on the worst possible input of a given size.