Quantum states, qubits, logic gates, and algorithms

Anton Frisk Kockum

June 2, 2022

In this lecture, we will study the basics of the circuit model of quantum computation. This introduces quantum bits, quantum gates, and other components in close similarity with concepts in classical computing and gives us the tools to begin investigating whether quantum computers can ever outperform classical computers. For these lecture notes, we have partly relied on material from Refs. [Nielsen and Chuang, 2000, Aaronson, 2018, Kockum and Nori, 2019]. This lecture has been adapted from the first lectures in the course "Quantum Computing" given at Chalmers University of Technology as a PhD course in 2019 (see the course homepage for full lecture notes as well as videos from those lectures) and as a master course in 2020 and 2021 (see this YouTube playlist for videos of the lectures in the 2021 edition).

1 Components of the circuit model

Loosely speaking, a computation requires a system that can represent data, a way to perform manipulation of that data, and a method for reading out the result of the computation. In the circuit model of quantum computation, we use:

- **Quantum bits (qubits)** to represent the data.
- **State preparation** to initialize the qubits in the input state we need to begin the computation.
- **Quantum gates** on the qubits to manipulate the data.
- **Measurements** on the qubits to read out the final result.

Below, we first say a few words about what qubits are. We then discuss various quantum gates, and what is required of such gates to allow us to perform any quantum computation we would like. We assume for now that it is possible to initialize our quantum computer in some simple state, and that we can read out the state of the qubits at the end of a computation.

2 Quantum bits

In a classical computer, the most basic unit of information is a *bit*, which can take two values: 0 and 1. In a quantum computer, the laws of quantum physics allow phenomena like superposition and entanglement. When discussing information processing in a quantum world, the most basic unit is therefore a *quantum bit*, usually called *qubit*, a two-level quantum system with a ground state $|0\rangle$ and an excited state $|1\rangle$. Unlike a classical bit, which only has two possible states, a quantum bit has infinitely many states: all superpositions of $|0\rangle$ and $|1\rangle$,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{1}$$

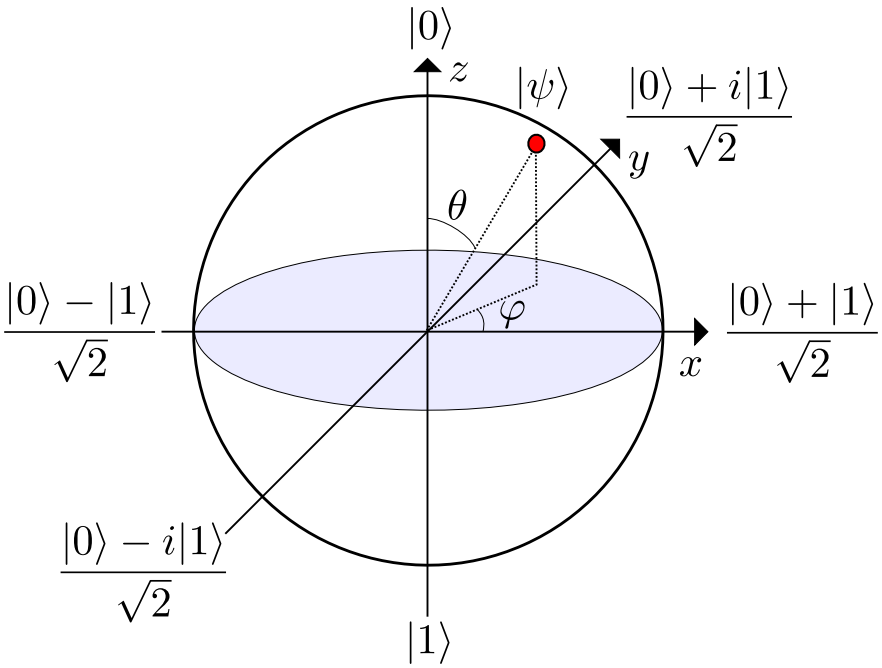

Figure 1: The Bloch-sphere representation of a qubit state. The north pole is the ground state $|0\rangle$ and the south pole is the excited state $|1\rangle$. To convert an arbitrary superposition of $|0\rangle$ and $|1\rangle$ to a point on the sphere, the parameterization $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$ is used.

where α and β are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. A measurement on this qubit state (in the basis of $|0\rangle$ and $|1\rangle$) gives the result 0 with probability $|\alpha|^2$ and the result 1 with probability $|\beta|^2$.

A useful tool for visualizing a qubit state is the *Bloch sphere* shown in Fig. 1. A state of the qubit is represented as a point on the surface of the sphere, which has radius 1. The two states of a classical bit correspond to the north and south poles on the sphere. The two states on opposite ends of the x axis are often denoted

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2}$$

If there are N qubits in a system, the total state of that system can be a superposition of 2^N different states: $|000\ldots00\rangle$, $|100\ldots00\rangle$, $|010\ldots00\rangle$, …, $|111\ldots10\rangle$, $|111\ldots11\rangle$. Note that N classical bits can be in *one* of these 2^N states, but not in a superposition of several of them. To store all the information about a general N-qubit state, one needs to keep track of the 2^N amplitudes in the superposition. This means that at least 2^N classical bits are required to represent the quantum system. This explains why it is hard for classical computers to simulate some quantum systems, and gives a first hint that quantum computers can be more powerful than classical ones (at least when it comes to simulating quantum systems).

There are many physical implementations of qubits, e.g., superconducting qubits, trapped ions, natural atoms, etc. These implementations are a topic for another lecture after this one. In the following, we assume that we have access to qubits, but do not care much about how they are made.

3 Single-qubit gates

Operations on a single qubit moves its state from $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ while preserving the norm $|\alpha|^2 + |\beta|^2 = 1 = |\alpha'|^2 + |\beta'|^2$. Such operations (gates) can be described by 2×2 unitary matrices.

Here we list some of the most common ones. First are the Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3}$$

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

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{5}$$

The Pauli matrices generate rotations around the corresponding axes on the Bloch sphere when exponentiated, e.g.,

$$R_x(\theta) = \exp(-i\theta X/2) = \cos(\theta/2)I - i\sin(\theta/2)X = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \tag{6}$$

where I is the identity matrix.

The X gate is the quantum equivalent of the classical NOT gate:

$$X(\alpha\left|0\right\rangle + \beta\left|1\right\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta\left|0\right\rangle + \alpha\left|1\right\rangle. \tag{7}$$

By adding the X and Z gates, one obtains the Hadamard gate

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{X+Z}{\sqrt{2}}. \tag{8}$$

This gate transforms the qubit state from the $\left|0\right\rangle, \left|1\right\rangle$ basis to the $\left|+\right\rangle, \left|-\right\rangle$ basis. The Hadamard gate is often used to create superposition states at the beginning of a quantum algorithm.

The Z gate applies a phase factor -1 to the $\left|1\right\rangle$ part of the qubit state. Two gates that apply other phase factors are often given their own names. One is the T, or $\pi/8$, gate:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix} = \exp(i\pi/8)\begin{pmatrix} \exp(-i\pi/8) & 0 \\ 0 & \exp(i\pi/8) \end{pmatrix}. \tag{9}$$

The other is the phase, or S, or P, gate

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = T^2. \tag{10}$$

4 Multi-qubit gates

To realize useful quantum algorithms, we also need to be able to make two or more qubits interact through multi-qubit gates. One way to achieve this is to let the state of one qubit control whether a certain single-qubit gate is applied to another qubit. One such gate is the controlled-NOT (CNOT) gate:

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{11}$$

Note that the two-qubit Hilbert space is spanned by the basis vectors $\left|00\right\rangle$, $\left|01\right\rangle$, $\left|10\right\rangle$, and $\left|11\right\rangle$, in that order. This is the tensor product of the two single-qubit Hilbert spaces. If you are not familiar with the tensor

product, we recommend you to follow the dedicated Qiskit tutorial [Qiskit, 2021]. The action of the CNOT gate is thus to do nothing if the first qubit is in state $|0\rangle$ ($|00\rangle$, $|01\rangle$ changes to $|00\rangle$, $|01\rangle$), and to apply NOT to the second qubit if the first qubit is in state $|1\rangle$ ($|10\rangle$, $|11\rangle$ changes to $|11\rangle$, $|10\rangle$).

The controlled-Z (CZ) gate can be defined in the same manner:

$$\mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{12}$$

More generally, the controlled application of a single-qubit unitary U to the second qubit takes the form

$$\begin{pmatrix} I_2 & 0_2 \\ 0_2 & U \end{pmatrix}. \tag{13}$$

There are also two-qubit gates that are not controlled operations. For example, the SWAP gate

$$\mathrm{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{14}$$

swaps the states $|01\rangle$, $|10\rangle$ to $|10\rangle$, $|01\rangle$.

It is also possible to define gates involving more than two qubits. For three-qubit gates, the most well-known ones are the Toffoli and Fredkin gates. The Toffoli gate is a controlled-controlled-NOT (CCNOT), i.e., the state of the third qubit is flipped if and only if both the first two qubits are in state $|1\rangle$:

$$\mathrm{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{15}$$

The Fredkin gate is a controlled-SWAP (CSWAP) gate, swapping the states of the second and third qubits if and only if the state of the first qubit is $|1\rangle$:

$$\mathrm{Fredkin} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{16}$$

Most practical implementations of quantum computing have limited connectivity between qubits, only allowing for pairwise interactions between nearest-neighbour qubits. This can prohibit direct implementations of multi-qubit gates with three or more qubits, but it turns out that any multi-qubit gate can be decomposed into a number of single- and multi-qubit gates.

5 Universal quantum computation

Are all the gates we specified above enough to carry out any quantum computation that we would like? Could we do any quantum computation using just a small subset of the gates above? These are questions about *universality*.

For classical computers, a set of gates is called universal if, by applying enough gates from this set in a sequence, it is possible to express any Boolean function on any number of bits. The classical NAND gate turns out to be universal all on its own, but there are other sets of gates that are not universal. For example, the set $\{\text{AND}, \text{OR}\}$ is not universal, because these gates cannot express non-monotone Boolean functions; changing an input bit from 0 to 1 in a circuit with these gates will never result in an output bit changing from 1 to 0.

For quantum computers, a gate set is called *universal* if the gates therein can be used to approximate any unitary transformation on any number of qubits to any desired precision. To understand what makes a gate set universal for quantum computing, let us see how a gate set can fail to be universal:

- **Inability to create superposition states**
  Some gates, e.g., X and CNOT, only change states in the computational basis into other states in the computational basis (e.g, $\text{CNOT}\,|11\rangle = |10\rangle$). These gates can maintain superposition states, but they cannot create new ones.

- **Inability to create entanglement**
  The Hadamard gate can create a superposition (e.g., $H\,|0\rangle = |+\rangle$), but it only acts on a single qubit, so it cannot create entanglement between two or more qubits. Clearly, no single-qubit gate can. Since an unentangled state of N qubits can be written $(\alpha_1\,|0\rangle + \beta_1\,|1\rangle) \otimes \ldots \otimes (\alpha_N\,|0\rangle + \beta_N\,|1\rangle)$, it can be specified using only $2N$ amplitudes, which a classical computer would be able to simulate efficiently.

- **Inability to create non-real amplitudes**
  A gate set like $\{\text{CNOT}, H\}$ would be able to create both entanglement and superposition states. However, the matrices specifying these gates only have real entries. Thus, they would never be able to create states with complex amplitudes.

- **The Gottesman-Knill theorem**
  If we take our gate set to be $\{\text{CNOT}, H, S\}$, we circumvent all the objections above. However, it turns out that this still is not enough to achieve universal quantum computation. This is the

  Gottesman-Knill theorem [Gottesman, 1999]: A quantum circuit using only the following elements can be simulated efficiently on a classical computer:

  1. Preparation of qubits in computational basis states,
  2. Quantum gates from the Clifford group (H, CNOT, S), and
  3. Measurements in the computational basis.

  The Clifford group is the group of operations that map Pauli matrices onto (possibly different) Pauli matrices, and is generated by the gates (H, CNOT, S). Quantum circuits with only these gates are also called stabilizer circuits (this concept is also important in quantum error correction). If one starts in the computational basis, the single-qubit states that can be reached using these circuits are the ± 1 points on the x, y, and z axes of the Bloch sphere. This restriction of the state space turns out to make the circuit classically simulatable even though the state space has states with both superposition, entanglement, and complex amplitudes.

What is then a universal gate set? Actually, replacing the Hadamard gate in the last gate set considered above, $\{\text{CNOT}, H, S\}$, with almost any other gate makes the set universal. We can also replace S with some other gate. One universal gate set is $\{\text{CNOT}, H, T\}$. Almost any two-qubit gate on its own is also universal.

We thus now know that we can implement any unitary operation on N qubits in a realistic experimental architecture that allows for a few single- and two-qubit gates. Does that mean that we can run quantum algorithms that work better than classical algorithms for some problems? This is too large a question to answer in the remainder of this lecture, but we can at least get started.

6 Comparing quantum and classical computers

Could a quantum computer find the answer to a problem that *cannot* be solved on a classical computer, even if there are no restrictions on the resources available to the classical computer? The answer is no. The classical computer could always simulate the quantum computation by storing, to enough precision, the 2^N amplitudes for the whole state of the quantum computer's N qubits, and changing these amplitudes according to the gates applied in the quantum algorithm. However, there is clearly a possibility that this classical simulation requires a lot more resources than the quantum computation itself. So a better question to ask is: are there problems that a quantum computer could solve *faster* than a classical computer? And before answering that, we need to say a bit more about universal gate sets.

7 The Solovay–Kitaev theorem

In Sec. 5, we saw that there are many gate sets that are universal for quantum computing, i.e., they can approximate any unitary transformation on any number of qubits to any desired precision ϵ. However, it is a different question whether that approximation is fast and efficient. For example, If the number of gates needed would scale exponentially in $1/\epsilon$, there would probably not be any practical use for such a universal gate set. Luckily, there is a theorem telling us that the scaling for any universal gate set is much more benign:

Solovay–Kitaev theorem: Let G a finite subset of $SU(2)$ and $U \in SU(2)$. If the group generated by G is dense in $SU(2)$, then for any $\varepsilon > 0$ it is possible to approximate U to precision ε using $O\big(\log^4(1/\varepsilon)\big)$ gates from G.

The proof can be found in Appendix 3 of Ref. [Nielsen and Chuang, 2000]. What the theorem essentially says is that if we have a gate set that can approximate any single-qubit rotation, then we only need relatively few gates from that set to do the approximation, so we can do the approximation fast. By adding some two-qubit gate to make the gate set universal, it can be shown that the total number of gates needed to approximate U on N qubits is at most $O\big(4^N \log^4(1/\varepsilon)\big)$. More recent results have shown that for certain gate sets, the number of gates can be brought down from $O\big(\log^4(1/\varepsilon)\big)$ to $O(\log(1/\varepsilon))$. There are also good algorithms for finding the gate sequences needed to achieve the Solovay-Kitaev bound.

This theorem, together with the observations in previous sections, is what gives us confidence that quantum computing has potential to be very useful. We now know how to find a universal gate set to approximate any unitary transformation, and we know that we can implement that approximation efficiently.

8 Algorithms and compilation

Upcoming lectures and exercises will feature examples of quantum algorithms. You now have all the necessary tools to understand the components of such algorithms: it is basically just gates (unitary matrices) acting

on quantum states (multiplying vectors).

It is important to note that quantum algorithms need to be compiled/transpiled to fit on the quantum hardware you want to run them on. Each hardware provider offers a universal gate set, but these gate sets can differ. For example, the IBM quantum computers can usually implement a few specific single-qubit X or Y rotations, arbitrary single-qubit Z rotations, and the two-qubit CNOT gate. Even though you can create a program with other gates, those gates will be converted into sequences of gates from this universal gate set. For example, a SWAP gate can be implemented as a sequence of three CNOT gates (exercise: show how).

Further compilation is necessary to deal with the fact that, in most hardware, not all qubits can talk directly to all other qubits. If you have written a program that includes, say, a CZ gate between qubits number 1 and 10 in your circuit, and these qubits then are physically located on opposite sides of a large quantum-computing chip, SWAP gates need to be inserted in your circuit to bring the two qubit states that are to interact physically closer on the chip. Once such constraints have been taken into account, a good compiler should also "compress" the circuit as much as possible, reducing the number of sequential operations by using various circuit identities to merge gates into fewer ones.

The compilation/transpilation is done by the provider backend without the user needing to worry about it. However, there is still a lot of room for improvement in the compilation algorithms in use today, and the noise in existing quantum hardware makes it imperative to keep circuits short to work well, so for an important project it can be useful to check the compiled circuit and see if it can be improved.

Exercises

1. (Nielsen & Chuang 4.7) Show that $XYX = -Y$ and use this to prove that $XR_y(\theta)X = R_y(-\theta)$.

2. (NC 4.13) It is useful to be able to simplify cicuits by inspection, using well-known identities. Prove the following three identities:

$$HXH = Z; \quad HYH = -Y; \quad HZH = X.$$

3. (NC 4.14) Use the previous exercise to show that $HTH = R_x(\pi/4)$, up to a global phase.

4. (NC 4.17) Construct a CNOT gate from one CZ gate and two Hadamard gates, specifying the control and target qubits.

5. Construct a Fredkin gate using three Toffoli gates.

6. Starting from the two-qubit state $|00\rangle$, can you create the following state using the gate set $\{\text{CNOT}, \text{H}, \text{S}\}$? If yes, show how. If no, explain why.

$$\frac{1}{\sqrt{2}}\Big(|00\rangle + e^{i\pi/4}\,|11\rangle\Big)$$

References

[Aaronson, 2018] Aaronson, S. (2018). Lecture Notes for Intro to Quantum Information Science.

[Gottesman, 1999] Gottesman, D. (1999). The Heisenberg Representation of Quantum Computers. In Corney, S. P., Delbourgo, R., and Jarvis, P. D., editors, *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, Cambridge, MA. International Press.

[Kockum and Nori, 2019] Kockum, A. F. and Nori, F. (2019). Quantum Bits with Josephson Junctions. In Tafuri, F., editor, *Fundamentals and Frontiers of the Josephson Effect*, pages 703–741. Springer.

[Nielsen and Chuang, 2000] Nielsen, M. A. and Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.

[Qiskit, 2021] Qiskit (2021). Multiple qubits and entangled states.