CS 599 P1: Introduction to Quantum Computation — Boston University, Fall 2025
Instructor: Alexander Poremba — **Scribe:** Alivia Pavel

LECTURE #5: REVERSIBLE COMPUTING & QUANTUM CIRCUIT MODEL

Date: 9/16/25

The goal of this lecture is to understand reversible computing and the quantum circuit model. We will compare quantum circuits and gates to classical, digital circuits and gates. We will explore similarities and differences between these two models of computation. In comparing the two, we will finally answer the question *"Can a quantum computer simulate a classical digital computer?"*

1 Quantum information processing

Before we can understand the relationship between classical and quantum computation, let us briefly review the basic ingredients behind quantum computations.

Quantum Bits or "Qubits". The fundamental unit of quantum information is the qubit:

- (**1 qubit**) A qubit is a complex linear combination of two logical states $|0\rangle$ and $|1\rangle$, i.e.,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad \text{such that } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1.$$

- (**n qubits**) An n-qubit state is a complex linear combination of 2^n logical states, i.e.,

$$|\psi\rangle = \sum_{x=(x_1,\ldots,x_n)\in\{0,1\}^n} \alpha_x\,|x_1,\ldots,x_n\rangle \quad \text{such that } \alpha_x \in \mathbb{C} \text{ and } \sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1.$$

Quantum Gates. The states of a quantum computation evolve via a sequence of linear transformations; these are typically referred to as *quantum gates*.

- Quantum gates are *unitary operators* that perform operations on qubits, and thus map qubits to qubits; in general, they are complex-valued matrices U such that

$$|\psi\rangle \mapsto U\,|\psi\rangle \quad \text{such that } U^\dagger U = I = UU^\dagger \text{ and } (U)^{-1} = U^\dagger.$$

  The *unitarity* condition is important because it ensures that:

  1. Quantum gates preserve the probabilistic nature of qubits: the total probability sums to 1.
  2. Quantum gates are inherently reversible: every quantum gate has an inverse, which is consistent with the postulate that closed quantum systems evolve in a reversible manner.

Let us now consider a simple example of a single-qubit quantum gate.

Example: The Pauli-X gate or the "Bit Flip Gate"

The X-gate is a linear transformation that acts on a single qubit, and can be represented as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } (X)^\dagger = X.$$

Its action on the computational basis is as follows:

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$
$$X|b\rangle = |b \otimes 1\rangle, \quad \text{for } b \in \{0,1\}$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ —[X]— } \beta|0\rangle + \alpha|1\rangle$$

We also countered the notion of a two-qubit gate; these types of gates act on a tensor product of basis states. Recall that a two-qubit state lives in a tensor product space $\mathbb{C}^2 \otimes \mathbb{C}^2$. Letting A denote the system of the first qubit, and letting B denote the system of the second qubit, we can represent the computational basis of the two-qubit system by the four basis states

$$\{|0\rangle_A \otimes |0\rangle_B, |0\rangle_A \otimes |1\rangle_B, |1\rangle_A \otimes |0\rangle_B, |1\rangle_A \otimes |1\rangle_B\}.$$

Let us now consider the following example of a two-qubit gate.

Example: Two-Qubit Gates

Consider the product gate consisting of an X gate on the A system and a Z gate on the B system, where

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ and } (Z)^\dagger = Z.$$

Letting $|\psi\rangle_{AB}$ be the quantum sate of the form

$$|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}|0\rangle_A \otimes |0\rangle_B + \frac{1}{\sqrt{2}}|0\rangle_A \otimes |1\rangle_B$$

we find that the output of $(X_A \otimes Z_B)|\psi\rangle_{AB}$ is of the form

$$|\psi\rangle_{AB} \left\{ \begin{array}{l} \text{—[X]—} \\ \\ \text{—[Z]—} \end{array} \right\} = \tfrac{1}{\sqrt{2}}|1\rangle_A \otimes |0\rangle_B - \tfrac{1}{\sqrt{2}}|1\rangle_A \otimes |1\rangle_B$$

Now that we have reviewed the essentials of quantum information processing, we will switch gears and compare this to the concept of a classical digital computer.

2 Classical digital computation

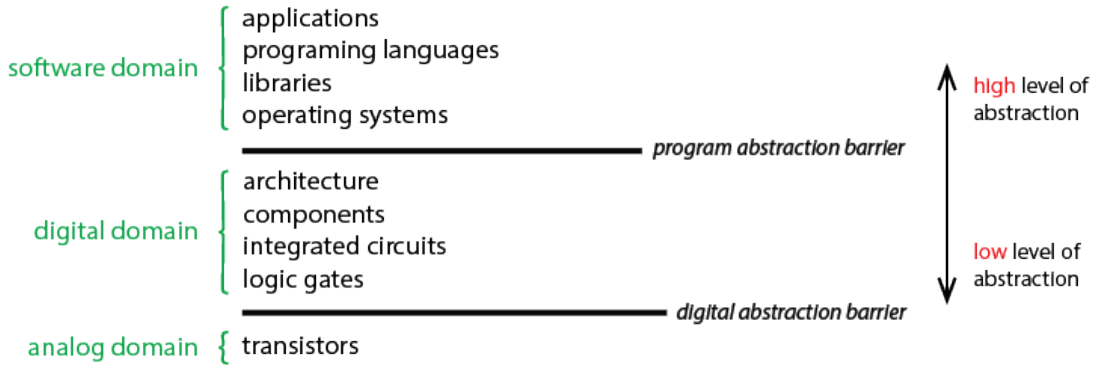
Can a quantum computer simulate a classical digital computer? To properly answer this question, we first need to be precise about what we mean by a "classical computer."

What does a classical digital computer do? At the most fundamental level, classical computers are function evaluation machines. They compute Boolean functions of the form

$$f : \{0,1\}^n \mapsto \{0,1\}^m ,$$

where f is a function that maps n input bits to m output bits. This might seem overly simplistic, but it's remarkably general: everything a classical computer does, from web browsing to video games, to scientific simulations, and more, can ultimately be reduced to evaluating some Boolean function on binary inputs.

How does a classical digital computer work? The magic of classical computers lies in how they build up from simple components to compute these complex functions. Classical computers compute Boolean functions via a sequence of logic gates operating under *many* layers of abstraction:



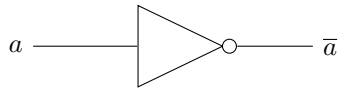
All software and functions of a classical computer can be reduced to a series of logic gates and bits. This hierarchical structure means that to understand what quantum computers need to simulate, we must understand these basic logic gates that form the foundation of all classical computation.

3 Classical Logic Gates

Now we'll examine the basic building blocks that make classical computation possible. These gates perform simple logical operations on one or two bits, but when combined, they can implement any computable function. Understanding these gates is essential because we'll later need to show how quantum computers can implement each of these operations.

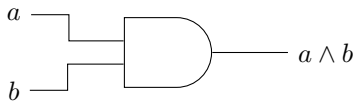
Let us now consider some of the most fundamental logical gates.

The NOT gate: The simplest gate is the NOT gate, which implements logical negation (flipping a bit):



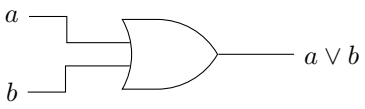
$$a \mapsto \overline{a} = a \oplus 1$$

The AND gate: The AND gate implements logical conjunction - it outputs 1 only when both inputs are 1:



$$(a, b) \mapsto a \wedge b = a \cdot b$$

The OR gate: The OR gate implements logical disjunction - it outputs 1 when at least one input is 1:



$$(a, b) \mapsto a \vee b$$

Universal gate sets

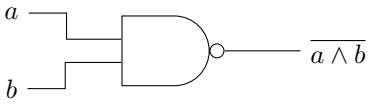
The gate set consisting of the following three logical gates

$$\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$$

is *universal* for classical computation; meaning it can represent any Boolean function as a logical circuit consisting solely of the aforementioned gates. This universality property means that any computation a classical computer can perform can be built from these three basic operations.

Another important gate is the `NAND` gate, which by itself is also universal for classical computation and can be used to compute any Boolean function.

The NAND gate The `NAND` gate ("NOT AND") produces a "0" output only when all inputs are true "1":



$$(a, b) \mapsto \overline{a \wedge b}$$

Logical Gate Truth Tables. To fully describe the input-output behavior of these logical gates, we state the complete truth tables for NOT, AND, OR, and NAND below:

NOT	
Input	Output
1	0
0	1

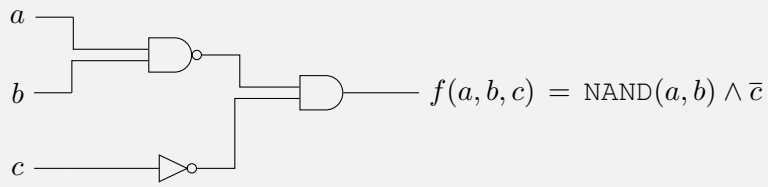
AND		
Input		Output
a	b	a $\wedge$ b
0	0	0
0	1	0
1	0	0
1	1	1

OR		
Input		Output
a	b	a $\vee$ b
0	0	0
0	1	1
1	0	1
1	1	1

NAND		
Input		Output
a	b	$\overline{a \wedge b}$
0	0	1
0	1	1
1	0	1
1	1	0

Example: Logical Circuit with Three Inputs

Here is an example of a logical circuit constructed from the universal set of logic gates. The circuit takes three binary inputs (a, b, c) and computes the function:



This demonstrates how simple logic gates can be combined to implement more advanced functions.

With this foundation in classical computation established, we can now examine the fundamental challenge that arises when we try to connect classical and quantum computation: the problem of irreversibility.

4 The Physics of Information

Before we can understand why reversibility matters in computation, we need to explore the deep connection between information processing and and the laws of physics. At this point, the reader might wonder:

- if digital computation requires us to *erase* information, say when resetting the value of a bit, and
- if the laws of physics require that all processes in nature must be inherently reversible (at least, in principle), then
- how can such irreversible operations possibly be performed in a physical system, such as a computer?

In 1961, Rolf Landauer made an important observation about the thermodynamic relationship between information and energy. This is now known as *Landauer's principle*.

Landauer's Principle

Any logically irreversible manipulation of information, such as the erasure of a logical bit, must be accompanied by a corresponding change in entropy, and thus in the release of *heat*.
Concretely, erasing a single bit of information requires a minimum energy dissipation of $k_B T \ln(2)$, where k_B is Boltzmann's constant and T is the temperature of the system.

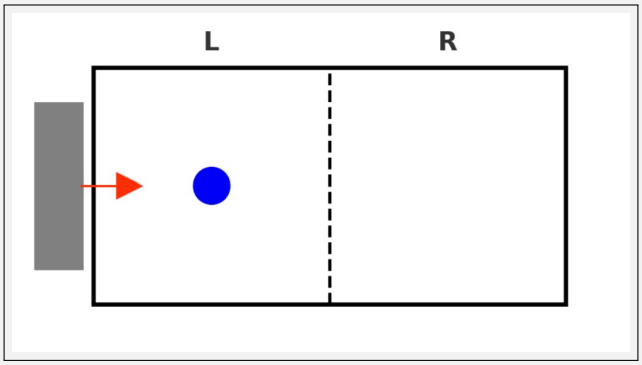
This principle establishes a fundamental physical limit: irreversible computation necessarily dissipates energy. Therefore, for any sufficiently long computation, this energy cost becomes a serious constraint.

To illustrate Landauer's principle, consider the following physical instantiation of a logical bit.

Example: Molecule in a box

A single molecule is put in a box which is divided by a removable partition. Once we insert the partition into the box, this simple physical system encodes a *logical bit*:

- The bit is "1" if the molecule is in the left half of the box (denoted by L); and
- The bit is "0" if the molecule is in the right half of the box (dentoed by R).



To **erase** this bit (i.e., to reset it to the standard state "0"), we remove the partition and push a piston from the left to the right, thereby compressing the contents of the box. This ensures that the molecule is now in the right-hand side of the box. According to **Landauer's principle**, this compression step requires work $k_B T \ln(2)$ and generates heat which is released into the environment. While this process is still reversible *in principle*—at least globally, when also taking the environment into account—erasing even a single bit of information nevertheless comes with a thermodynamic cost.

5 Reversible Computing

In the 1970s, Charles Bennett made an important observation that greatly influenced the field of quantum computation. Bennett observed that **Landauer's principle only applies to irreversible operations**. If we could make computation *reversible*, we could, in principle, compute without energy dissipation.

Bennett showed that any classical computation can be made reversible by

- keeping a complete history of the computation;
- using only reversible logic gates; and
- never erasing information.

This insight directly connects to quantum computation because quantum mechanics is fundamentally reversible—all quantum operations are unitary and therefore invertible.

6 Reversible Computing and Classical Circuits

Now that we understand the physics behind information processing, we can examine how this applies to computational systems and why reversibility becomes crucial when we transition to quantum computing.

Problems with irreversibility. Classical logic gates like `AND` and `OR` are *irreversible*, meaning we lose information during computation. This creates both physical and computational problems:

- `AND` gate: If output is 0, we cannot actually determine if input was $(0, 0)$, $(0, 1)$, or $(1, 0)$.
- `OR` gate: If output is 1, we cannot determine if input was $(0, 1)$, $(1, 0)$, or $(1, 1)$.

According to Landauer's principle, this irreversibility leads to information loss and unavoidable energy dissipation into the environment. Evolution in closed quantum systems, by contrast, is inherently reversible, and thus all quantum operations must also be reversible. So if we want quantum computers to simulate classical computers, we need to find a way to make classical computation reversible.

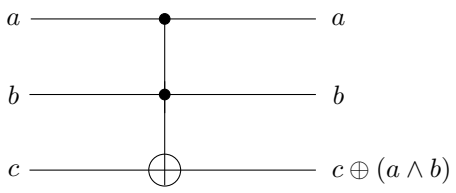
Following Bennett's insight, the solution is to use *reversible logic gates* instead. These gates have special properties that preserve information and avoid energy dissipation:

- the number of inputs bits equals the number of output bits;
- the mapping is bijective (one-to-one and onto); and
- we can always recover the input from the output.

Following Bennett's insight, no information would be lost during such a computation. By solely using reversible gates, we can nevertheless implement any classical computation while satisfying the reversibility requirements of quantum mechanics.

A reversible NAND gate. Previously, we saw that a `NAND` gate is universal for classical computation: a circuit comprised of `NAND` gates can compute any Boolean function $f : \{0, 1\}^n \to \{0, 1\}^m$.

To make the gate *reversible*, we can simply "carry the input bits around." The reversible analog of a `NAND` gate is called a `Toffoli` gate; it is described by the following circuit notation:



In other words, a `Toffoli` gate computes the following:

$$(a, b, c) \mapsto (a,\ b,\ c \oplus (a \wedge b)).$$

By setting $c = 1$, we recover precisely the `NAND` gate from before. Notice that the gate is reversible because we still have the input bits; in particular, executing the gate twice in a row onto its output resets the input back to its initial value. We will now see how to use the same principle of reversible computing to evaluate an arbitrary Boolean function on a quantum computer.

7 Can Quantum Computers Simulate Classical Computers?

We're now ready to answer our main question. Armed with our understanding of reversible computation and quantum gates, we can provide a definitive answer with a clear explanation of why it works.

The Answer: Yes! Quantum computers can efficiently simulate classical computers. Suppose we wish to compute an arbitrary Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ which, in general, is not going to be reversible. Following Bennett's insights, we will simply compute f in a reversible manner, just like in the `Toffoli` gate from before. Suppose we wish to compute $f(x)$, for some input $x \in \{0,1\}^n$.

First, we initialize n qubits and set them to the value $x \in \{0,1\}^n$; in other words, we have

$$\underbrace{|x_1, x_2, \ldots, x_n\rangle}_{n \text{ many "input" qubits}} .$$

Next, we append m additional qubits which we will use to store the output of the function, i.e.,

$$\underbrace{|x_1, x_2, \ldots, x_n\rangle}_{n \text{ many "input" qubits}} \otimes \underbrace{|0, 0, \ldots, 0\rangle}_{m \text{ many "output" qubits}} ,$$

resulting in a total of $n + m$ qubits of memory. By carefully[1] replacing each logical gate (in the description of the classical circuit which computes f) with its corresponding reversible variant, it is possible to perform the following reversible evaluation U_f of the Boolean function f:

$$\underbrace{|x_1, x_2, \ldots, x_n\rangle}_{n \text{ many "input" qubits}} \otimes \underbrace{|0, 0, \ldots, 0\rangle}_{m \text{ many "output" qubits}} \overset{U_f}{\mapsto} \underbrace{|x_1, x_2, \ldots, x_n\rangle}_{n \text{ many "input" qubits}} \otimes \underbrace{|f(x_1, x_2, \ldots, x_n)\rangle}_{m \text{ many "output" qubits}} .$$

Because we still have access to the input $x \in \{0,1\}^n$ of f, this is reversible **even if f in itself is not reversible!** To illustrate why this is the case, let us fully describe what the the corresponding $(n+m)$-qubit unitary operator U_f would look like. Recall that, to describe a linear operator, we need to fully specify how it acts on all of the $n + m$ many computational basis states.

Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_m)$. Then, we let U_f be defined as follows:

$$U_f \; |x\rangle \otimes |y\rangle \quad = \quad |x\rangle \otimes |y \oplus f(x)\rangle .$$

Clearly, this is reversible: if we apply U_f twice in a row, this resets the input qubits back to their initial state. Moreover, by setting $y = (0, 0, \ldots, 0)$, we can compute f as before! Most importantly, however, U_f is also *unitary*, and thus a perfectly valid reversible quantum operation.

To see how U_f acts on a general quantum input state on $n + m$ qubits, we can consider an arbitrary $(n + m)$-qubit input state of the form

$$|\psi\rangle = \sum_{\substack{x \in \{0,1\}^n \\ y \in \{0,1\}^m}} \alpha_{x,y} \, |x\rangle \otimes |y\rangle .$$

[1] This may require additional "workspace qubits" which can hold intermediate output values; for simplicity, we ignore these subtleties in this lecture.

Then, by linearity, we find that U_f acts as follows:

$$U_f |\psi\rangle = \sum_{\substack{x\in\{0,1\}^n \\ y\in\{0,1\}^m}} \alpha_{x,y} |x\rangle \otimes |y \oplus f(x)\rangle .$$

Important Note: The Converse Question

A related, and perhaps even more interesting, question is:

"*Can classical computers efficiently simulate quantum computers?*"

This is believed to be **false** in general. While classical computers can simulate quantum computers (by keeping track of all the complex amplitudes), this simulation generally requires exponential amounts of resources as the number of qubits grows. Quantum computers appear to have computational advantages for certain problems (like *Shor's algorithm for factoring*), suggesting that quantum computation is fundamentally more powerful than classical computation.

8 Quantum Circuit Model

A quantum circuit is a graphical representation of a quantum algorithm, where quantum information is processed through a sequence of quantum gates. The model consists of the following components:

- **Quantum Input States:** Each wire in the circuit corresponds to a quantum bit (qubit), initialized in a known state such as $|0\rangle$, $|1\rangle$, or a superposition state. The initial state of the system is typically denoted as $|\psi_{\text{in}}\rangle$.
- **Wires:** Horizontal lines represent the evolution of individual qubits over time. Each wire is labeled with a qubit identifier (e.g., q_1, q_2, ...), or the label of the system, say A or B.
- **Single-Qubit Gates:** These gates act on individual qubits and include operations such as the Hadamard gate (H), Pauli gates (X, Y, Z), and rotation gates. They modify the state of a single qubit.
- **Multi-Qubit Gates:** These gates involve interactions between two or more qubits. Common examples include the controlled-NOT (CNOT) gate, controlled-phase gates, and general n-qubit unitary operations such as U acting on multiple wires simultaneously.
- **Time Evolution:** The circuit is read from left to right, with each column representing a time step or layer of operations. This direction indicates the progression of quantum computation. In particular, the gates all the way on the left get applied first.
- **Measurements:** Measurement operations are typically represented by meter symbols and collapse the quantum state of a qubit into a classical bit. Measured wires are followed by classical output lines.
- **Output State:** Qubits that are not measured remain in a quantum state and collectively define the final output state $|\psi_{\text{out}}\rangle$. This state may be used for further quantum processing or interpreted as the result of the computation.

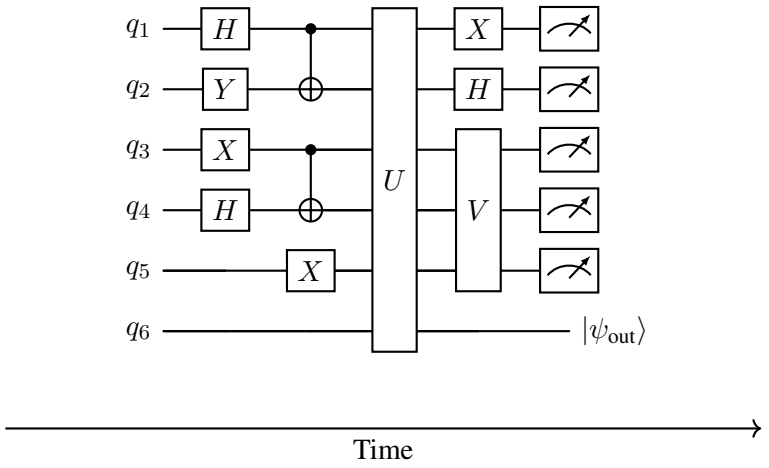


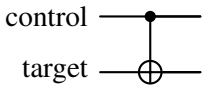
Figure 1: A quantum circuit which takes as input 6 qubits, denoted by $q_1, q_2, \ldots, q_6$. Here, the unitary U is a 6-qubit quantum gate and the unitary V is a 3-qubit quantum gate. The final output state of the computation is a single unmeasured qubit in the last wire, denoted by $|\psi_{\text{out}}\rangle$. The two-qubit gates are both CNOT gates.

9 Universal Quantum Gate Sets

Previously, we saw that the gate set $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$ is universal for classical computation. Therefore, one might wonder whether quantum computation also admits *universal quantum gate sets*; meaning, a set of gates that can approximately implement any quantum circuit with arbitrary precision.

It turns out that, in the quantum case, the answer is also **yes**! For the remainder of this section, we will take a closer look at some of these fundamental building blocks.

The CNOT gate. An important quantum gates is the controlled-NOT (CNOT); it is represented as



The CNOT gate flips the target qubit in the second register (represented by the second wire above) if and only if the control qubit in the first register (represented by the first wire) is in the state $|1\rangle$.

In matrix form, it is occasionally represented as

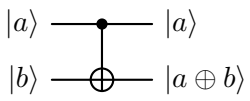
$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

However, we can also represent it by specifying how it acts on a given set of basis vectors. In the case of the product space $\mathbb{C}^2 \otimes \mathbb{C}^2$, we have the canonical basis given by

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$$

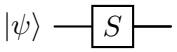
$$\begin{aligned}
\text{CNOT } |0\rangle \otimes |0\rangle &= |0\rangle \otimes |0\rangle \\
\text{CNOT } |0\rangle \otimes |1\rangle &= |0\rangle \otimes |1\rangle \\
\text{CNOT } |1\rangle \otimes |0\rangle &= |1\rangle \otimes |1\rangle \,. \\
\text{CNOT } |1\rangle \otimes |1\rangle &= |1\rangle \otimes |0\rangle \,.
\end{aligned}$$

Therefore, we can represent the action of the CNOT via the following quantum circuit:



In other words, the CNOT performs the operations $(a, b) \mapsto (a, a \oplus b)$ where $\oplus$ is XOR (or, binary addition modulo 2). Notice that this is a unitary matrix (you can verify that $\text{CNOT}^\dagger \cdot \text{CNOT} = I$), which means the operation is reversible—we can always undo it by applying the same gate again.

The S gate. The S gate is a single-qubit gate that applies a phase of $\frac{\pi}{2}$ to the $|1\rangle$ component of a qubit. It is represented as



In matrix form, the S gate is given by

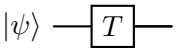
$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}.$$

This gate leaves the $|0\rangle$ state unchanged and multiplies the $|1\rangle$ state by the imaginary unit i. For example,

$$\begin{aligned}
S\ |0\rangle &= |0\rangle \,, \\
S\ |1\rangle &= i\ |1\rangle \,.
\end{aligned}$$

The S gate is also known as the phase gate, and satisfies $S^2 = Z$, where Z is the Pauli-Z gate. It is unitary and reversible, with $S^\dagger = S^{-1}$.

The T gate. The T gate is another single-qubit phase gate, applying a smaller phase of $\frac{\pi}{4}$ to the $|1\rangle$ component. It is represented as



In matrix form, the T gate is given by

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

This gate leaves the $|0\rangle$ state unchanged and multiplies the $|1\rangle$ state by $e^{i\pi/4}$. For example,

$$\begin{aligned}
T\ |0\rangle &= |0\rangle \,, \\
T\ |1\rangle &= e^{i\pi/4}\ |1\rangle \,.
\end{aligned}$$

The T gate is sometimes called the $\pi/8$ gate and satisfies $T^4 = Z$. It is a crucial component in universal quantum gate sets, especially in fault-tolerant quantum computing. Like the S gate, it is unitary and reversible, with $T^\dagger = T^{-1}$.

These aforementioned gates allow us to construct universal quantum get sets:

Universal quantum gates

The following sets of quantum gates are *universal* for quantum computation; in particular, they can approximate an arbitrary quantum circuit to any desired accuracy:

- $\{$`CNOT, H, S, T`$\}$ — This gate set is universal for quantum computation in the sense of approximating any unitary operation on qubits. The Hadamard (H) and phase gates (S, T) generate arbitrary single-qubit rotations, while the CNOT enables entanglement between qubits.
- $\{$`Toffoli, H`$\}$ — This set is universal for quantum computation when combined with quantum ancilla qubits. The Toffoli gate (also known as the controlled-controlled-NOT) is a reversible classical gate that, when paired with the Hadamard gate, allows for the simulation of quantum behavior. This gate set is particularly relevant in reversible computing and in contexts where classical logic is embedded within quantum circuits.