CMSC 657: Introduction to Quantum Information Processing
Lecture 5

Instructor: Daniel Gottesman

Fall 2024

1 Reversible Computation and Quantum Circuits

1.1 Gates and Circuits

The fundamental building block of a quantum circuit is the *quantum gate*. A quantum gate is an operation, often unitary, that counts as a single step of computation. Usually we insist that a single quantum gate only acts non-trivially on a small number of qubits, say two or three. We then like to study how to build *quantum circuits*, implementations of more complicated many-qubit unitary operations broken up into individual quantum gates. Note that quantum circuits also include non-unitary elements, particularly state preparation and measurement.

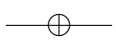
1.2 Reversible Classical Computation

Unitary gates are *reversible*: If $|\phi\rangle = U|\psi\rangle$, then $|\psi\rangle = U^\dagger|\phi\rangle$, so it is always possible to recover the input of a unitary quantum gate from the output (assuming you have the capability to perform arbitrary unitary operations). Let us therefore begin by studying *reversible* classical gates.

Some gates are naturally reversible, such as the NOT gate:

in	out
0	1
1	0

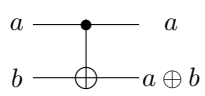
We often draw circuit diagrams, for instance:



In these circuit diagrams, time goes left to right. Each line represents a bit or later a qubit.

Another example of a reversible gate is the XOR gate:

in		out	
a	b	a	$a \oplus b$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



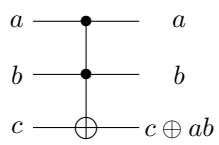
Note that if we only have one output $a \oplus b$, this is not reversible.

Other gates are harder to make reversible, for instance AND:

in		out	
a	b	a	ab
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

But we can make it reversible by keeping both input bits and adding another bit for the output, getting the Toffoli gate:

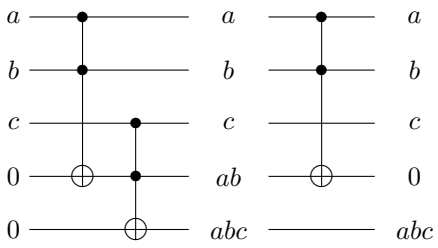
in			out		
a	b	c	a	b	$c \oplus ab$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0



When $c = 0$ for the input, the Toffoli gate computes the AND of a and b in the third register. When $c = 1$, it computes the NOT of the AND.

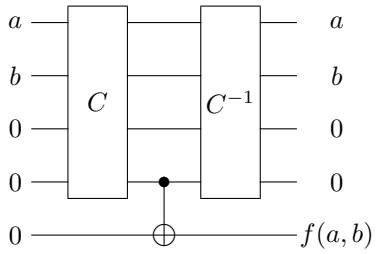
We can make any classical computation reversible in much the same way, by keeping all the input bits and simply adding additional bits to store the output of the circuit. Normally, however, we want to build up a large circuit as a sequence of reversible gates. If we keep the input bits at every stage, we will end up with a lot of left-over scratch bits at the end of the computation.

Example: (a AND b) AND $c = abc$.



The 4th output ab is a scratch bit — we don't care about it.

We can remove scratch bits in a long computation by uncomputing them. For instance, in the example, add another Toffoli gate (1,2 to 4) to uncompute the ab output. (Last part of figure above.) In general, given circuit C, do C, XOR output into another ancilla, then do C^{-1}, which is implemented as inverses of gates in C in reverse.



In a classical reversible computation, removing the scratch bits is a convenience. It saves space and is more aesthetically pleasing. But in a quantum computation, it is essential to remove the scratch bits, as we will see shortly.

1.3 Quantum Gates

We can immediately get some quantum gates by taking classical reversible gates and putting kets around everything. E.g.,

$$\text{CNOT}|a,b\rangle = |a, a \oplus b\rangle. \tag{1}$$

(CNOT = XOR, but more commonly called CNOT in the quantum context.) This only determines the action on basis states, but we can extend to the full Hilbert space by linearity:

$$\text{CNOT}(\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle) = \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle + \delta|10\rangle. \tag{2}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3}$$

Similarly, we have the quantum Toffoli gate

$$\text{Tof}|a,b,c\rangle = |a,b,c \oplus ab\rangle. \tag{4}$$

But if we only have classical gates, nothing quantumly interesting can happen. We must also add gates that modify superpositions by creating or destroying them, or by simply changing phases. For instance,

Hadamard transform —[H]— $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ $|0\rangle \mapsto |0\rangle + |1\rangle$, $|1\rangle \mapsto |0\rangle - |1\rangle$

Phase rotation —[R_θ]— $R_\theta = \begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix}$ $|0\rangle \mapsto e^{-i\theta}|0\rangle$, $|1\rangle \mapsto e^{i\theta}|1\rangle$

Note that $H^2 = I$:

$$H^2 = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5}$$

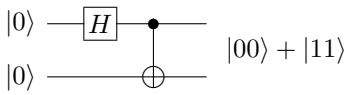
Also, note that $R_\theta = e^{-i\theta}\begin{pmatrix} 1 & 0 \\ 0 & e^{i2\theta} \end{pmatrix}$, and that the global phase $e^{-i\theta}$ has no physical significance, so it is equivalent to the gate $\begin{pmatrix} 1 & 0 \\ 0 & e^{i2\theta} \end{pmatrix}$.

We also need some non-unitary operations to make a complete circuit. We need some way of making the inputs to the circuit, and often we need extra helper qubits (called "ancillas"). At a minimum, we need a way of preparing $|0\rangle$ states.

We also need measurements to get the answer out at the end. At a minimum, we have measurement in the computational basis:

$$\alpha|0\rangle + \beta|1\rangle \mapsto 0 \text{ with prob. } |\alpha|^2, \ 1 \text{ with prob. } |\beta|^2. \tag{6}$$

We can put these circuit elements together in various ways to make more interesting quantum states. For instance, we can make an EPR pair entangled state $|00\rangle + |11\rangle$.



Note that getting rid of the scratch bits by uncomputing is essential to allow interference for quantum circuits. Compare the two versions of the example computing abc on a superposition, such as $|000\rangle + |110\rangle$.

$$\text{With scratch bit: } |000\rangle|00\rangle + |110\rangle|10\rangle \tag{7}$$

$$\text{Without scratch bit:} |000\rangle|00\rangle + |110\rangle|00\rangle. \tag{8}$$

Now trace out the fourth qubit, since we don't want it. In the first case, we have a *mixed state*:

$$|000\rangle|0\rangle\langle 000|\langle 0| + |110\rangle|0\rangle\langle 110|\langle 0|. \tag{9}$$

In the second case, we have a pure state:

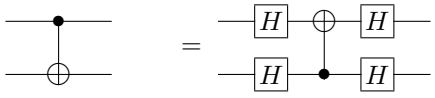
$$|000\rangle|0\rangle + |110\rangle|0\rangle. \tag{10}$$

We will return to this point next week when we talk about quantum algorithms, when using the mixed state version will cause the algorithm to fail.

1.3.1 Direction of CNOT Gates

The possibility of superposition means even classical gates behave a bit differently than one might expect. For instance, in a classical CNOT gate, the control bit does not change. But in a quantum CNOT gate, it can:

$$\text{CNOT}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = |00\rangle - |01\rangle + |11\rangle - |10\rangle = (|0\rangle - |1\rangle)(|0\rangle - |1\rangle). \tag{11}$$



Let us work the RHS out:

$$(H \otimes H)CNOT(2,1)(H \otimes H)|a\rangle|b\rangle = (H \otimes H)CNOT(2,1)(|0\rangle + (-1)^a|1\rangle) \otimes (|0\rangle + (-1)^b|1\rangle) \tag{12}$$

$$= (H \otimes H)(|00\rangle + (-1)^a|10\rangle + (-1)^b|11\rangle + (-1)^{a+b}|01\rangle) \tag{13}$$

$$= (H \otimes H)(|0\rangle + (-1)^a|1\rangle) \otimes (|0\rangle + (-1)^{a+b}|1\rangle). \tag{14}$$

Now,

$$H(|0\rangle + (-1)^a|1\rangle) = |0\rangle + |1\rangle + (-1)^a|0\rangle - (-1)^a|1\rangle \tag{15}$$

$$= (1 + (-1)^a)|0\rangle + (1 - (-1)^a)|1\rangle \tag{16}$$

$$= |a\rangle, \tag{17}$$

and similarly

$$H(|0\rangle + (-1)^{a+b}|1\rangle) = |a + b\rangle. \tag{18}$$

Therefore, the RHS is equal to $|a\rangle|a + b\rangle = CNOT(1,2)|a\rangle|b\rangle$.

The direction of the CNOT depends on what basis we look at it in!