CSE610 Week 3: Quantum Computations and Circuits

How to Get Entanglement

Here is a matrix whose understanding is key. Its standard name is $\mathbf{CNOT}$ for "controlled-NOT"; this is sometimes abbreviated to $\mathbf{CX}$.

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

Some observations:

1. This is a permutation matrix, and as-such is unitary.
2. It is also real symmetric, so it is Hermitian too.
3. It is not a tensor product of two 2×2 matrices. To prove this, represent the matrices by $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$. Then $A \otimes B = \begin{bmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{bmatrix}$. In order for this to equal $\mathbf{CNOT}$, we would have to solve $ag = 0$, $ae = 1$, and $dg = 1$. But these three are already impossible.
4. Hence, $\mathbf{CNOT}$ does not operate on each "tier" separately. As a 4×4 matrix, it acts on two qubits, because we have $N = 4 = d^n$ with $d = 2$ and $n = 2$. So each tier is a qubit, but $\mathbf{CNOT}$ has a joint action on the pair, not just on each qubit separately.

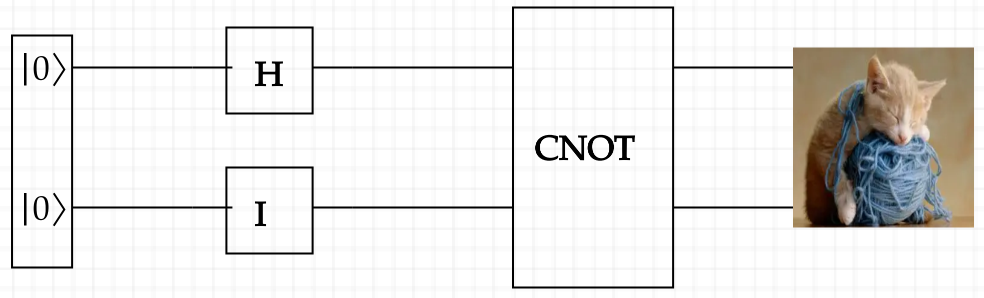
A simple example of a 4×4 matrix that is a tensor product is $\mathbf{H} \otimes \mathbf{I} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$. Well, it is really acting only on the first qubiot, because it is the identity on the second qubit. Now let us compose it with $\mathbf{CNOT}$, applying $\mathbf{H} \otimes \mathbf{I}$ first to whatever input $|x\rangle$ is given, say $x = 00$. To compose the matrices, we have to remember to put $\mathbf{H} \otimes \mathbf{I}$ on the right. So the computation

$$\mathbf{CNOT} \cdot (\mathbf{H} \otimes \mathbf{I}) \cdot |00\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

can be reduced by multiplying the matrices first or by multiplying $(\mathbf{H} \otimes \mathbf{I})|00\rangle$ first. The latter is more efficient and we get

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

This is the entangled state we saw before. We can diagram what happened as



We can't say we have separate values $|y\rangle$ and $|z\rangle$ on the wires anymore, because the state does not equal any tensor product of the form $|y\rangle \otimes |z\rangle$.

The entangled kitten image actually comes from a story in Britain's *New Scientist* magazine titled "Entangle Schrödinger's Cat to up its quantum weirdness." Come-on, it's just the vector $[1, 0, 0, 1]$, folks. It does, however, already warn us that whereas Boolean circuits always have a definite local bit value on every wire coming into or out of a gate, that won't be true in **quantum circuits**.

We could instead regard CNOT as an operator on our representation of playing card suits, where taking the order clubs-diamonds-hearts-spades used in bridge, we have:

$$|\clubsuit\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |\diamondsuit\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |\heartsuit\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |\spadesuit\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Then $\dfrac{[1,0,0,1]^T}{\sqrt{2}} = \dfrac{|\clubsuit\rangle + |\spadesuit\rangle}{\sqrt{2}}$, but this is not entangled---it's still a single-tier system. In order to have entanglement, there must be a representation of two separate physical systems that interact. Thus entanglement is not an innate mathematical property of the vector $[1, 0, 0, 1]$.

We could try to make it entangled by using a 2-bit binary code for the suits. The first bit could be 0 for minor suit, 1 for major suit. The second bit could be 0 for black suit, 1 for red suit. Then we'd have $|\clubsuit\rangle = |00\rangle$, $|\diamondsuit\rangle = |01\rangle$, thus far agreeing with the standard coding, but $|\heartsuit\rangle = |11\rangle$ rather than $|10\rangle$, and $|\spadesuit\rangle = |10\rangle$ under this code. So $\sqrt{0.5}[1, 0, 0, 1]^T$ would become the code for

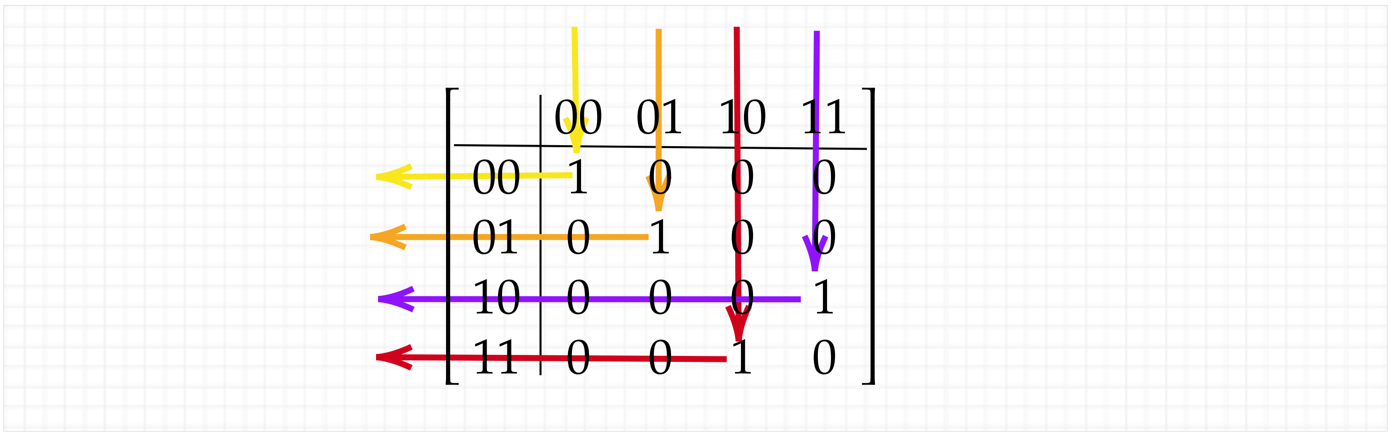
$$\frac{|\clubsuit\rangle + |\spadesuit\rangle}{\sqrt{2}} = \frac{|00\rangle + |10\rangle}{\sqrt{2}} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes |0\rangle,$$

which is separable---so not entangled. In order to minic the standard two-qubit code, we should define the second bit to be 1 if the suit symbol has a pointed top, 0 for a rounded top. Then we get $|\clubsuit\rangle = |00\rangle$, $|\lozenge\rangle = |01\rangle$, and now $|\heartsuit\rangle = |10\rangle$ and $|\spadesuit\rangle = |11\rangle$ as desired. If it really made sense for "minor/major" and "rounded/pointy" to be attributes of interacting physical entities, then we could say that $|\clubsuit\rangle + |\spadesuit\rangle$ is entangled. When they are, the effect is the same as if the deck has only black cards: you can never draw a card that is major but not pointy (hears) or a card that is pointy but minor (diamonds).

Note that under both of these codes for suits, CNOT has the effect of switching $|\heartsuit\rangle$ and $|\spadesuit\rangle$. So CNOT switches between these coding schemes. This is all the more a reason to stick with the standard basis and binary code order at first, avoiding such "relative" coding issues for the time being. So long as the standard code truly represents a real physical system, entanglement as modeled in the mathematical representation corresponds to real physical entanglement.

More about CNOT

Any linear operator is uniquely defined by its values on a particular basis, and on the standard basis, the values are: $\mathrm{CNOT}e_{00} = \mathrm{CNOT}|00\rangle = |00\rangle$, $\mathrm{CNOT}e_{01} = \mathrm{CNOT}|01\rangle = |01\rangle$, $\mathrm{CNOT}e_{10} = \mathrm{CNOT}|10\rangle = |11\rangle$, and $\mathrm{CNOT}e_{11} = \mathrm{CNOT}|11\rangle = |10\rangle$. We can get these from the respective columns of the CNOT matrix, and we can also label the quantum coordinates right on it:



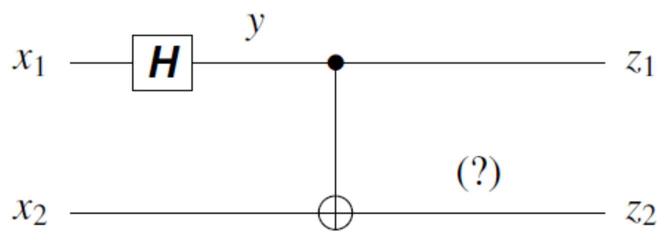
$$\left[\begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 00 & 1 & 0 & 0 & 0 \\ 01 & 0 & 1 & 0 & 0 \\ 10 & 0 & 0 & 0 & 1 \\ 11 & 0 & 0 & 1 & 0 \end{array}\right]$$

Because we multiply column vectors, the co-ordinates of the argument vector come in the top and go out to the left. If the first qubit is 0, then the whole gate acts as the identity. But if the first qubit is 1, then the basis value of the second qubit gets flipped---the same action as the **NOT** gate X. Hence the name Controlled-NOT, abbreviated CNOT: the **NOT** action is controlled by the first qubit. The action on a general 2-qubit quantum state $\phi = (a, b, c, d)$ is even easier to picture:

$$\text{CNOT}\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ d \\ c \end{pmatrix}.$$

All it does is switch the third and fourth components---of any 4-dim. state vector. Hence, CNOT is a **permutation gate** and is entirely deterministic. Permuting these two indices is exactly what we need to transform the separable state $\frac{1}{\sqrt{2}}(1,0,1,0)$ into the entangled state $\frac{1}{\sqrt{2}}(1,0,0,1)$.

The symbol for a **CNOT** gate is to use a black dot to represent the control on the *source qubit* and $\oplus$ (which I have used as a symbol for XOR) on the *target qubit*. This is more easily pictured by a quantum circuit diagram:



If $x_1 = |0\rangle$, then we can tell exactly what y is: it is the $|+\rangle$ state. And if $x_1 = |1\rangle$, then $y = |-\rangle$. If x_1 is any separate qubit state $(a,b) = a|0\rangle + b|1\rangle$, then by linearity we know that $y = a|+\rangle + b|-\rangle$. This expresses y over the transformed basis; in the standard basis it is

$$\frac{1}{\sqrt{2}}(a(1,1)+b(1,-1)) = \frac{1}{\sqrt{2}}(a+b,a-b) .$$

So we can say exactly what the input coming in to the first "wire" of the CNOT gate is. And the input to the second wire is just whatever x_2 is. But because that gate does entanglement, we cannot specify individual values for the wires coming *out*. The state is an inseparable 2-qubit state:

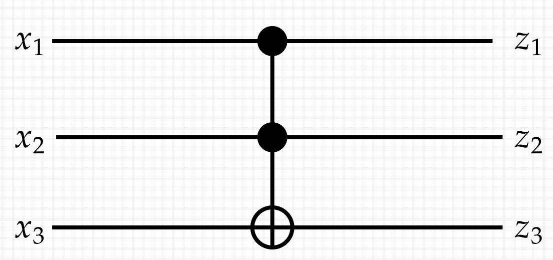
$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right).$$

If you measure either qubit individually, you get 0 or 1 with equal probability. This is the same as if you measured the state $|++\rangle = \frac{1}{2}[1,1,1,1]^T$. But that state is outwardly as well as inwardly different. When *both* qubits are to be measured, it allows 01 and 10 as possible outcomes, whereas measuring the entangled state does not. (We will cover measurements in more detail after circuits.)

Three Qubits and More

The **CNOT** gate by itself has the logical description $z_1 = x_1$ and $z_2 = x_1 \oplus x_2$. This means that if $x_1 = 0$ then $z_2 = x_2$, but if $x_1 = 1$ then $z_2 = \neg x_2$. Since this description is complete for all of the standard basis inputs $x = x_1 x_2 = 00, 01, 10, 11$, it extends by linearity to all quantum states. We can use this idea to specify the 3-qubit **Toffoli gate** (**Tof**). It has inputs x_1, x_2, x_3 and symbolic outputs z_1, z_2, z_3 (which, however, might not have individual values in non-basis cases owing to entanglement). Its spec in the basis quantum coordinates is:

$$z_1 = x_1,\; z_2 = x_2, z_3 = x_3 \oplus (x_1 \wedge x_2).$$



$$\left[\begin{array}{c|cccccccc}
 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
001 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
011 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
100 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
101 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
110 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
111 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}\right]$$

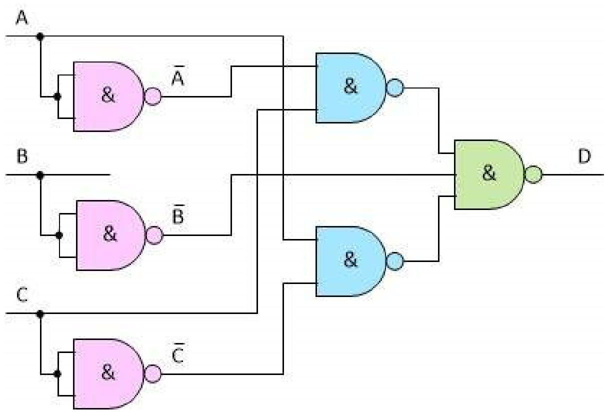
Of particular note is that if x_3 is fixed to be a constant-1 input, then

$$z_3 = \neg(x_1 \wedge x_2) = NAND(x_1, x_2).$$

Thus the Toffoli gate subsumes a classical NAND gate, except that you need an extra "helper wire" to put $x_3 = 1$ and you gate two extra output wires z_1, z_2 that only compute the identity on x_1, x_2 (in classical logic, that is---a non-basis quantum state can have knock-on effects even though all Toffoli does is switch the 7th and 8th components of the state vectors). If you have m Toffoli gates, then you get only order-of m wastage of wires, and you can use the good ones to simulate any Boolean circuit of m NAND gates. This already tells us a key fact:

Theorem: Classical Boolean circuits can be efficiently simulated by quantum circuits that don't even do any superposition or entanglement.

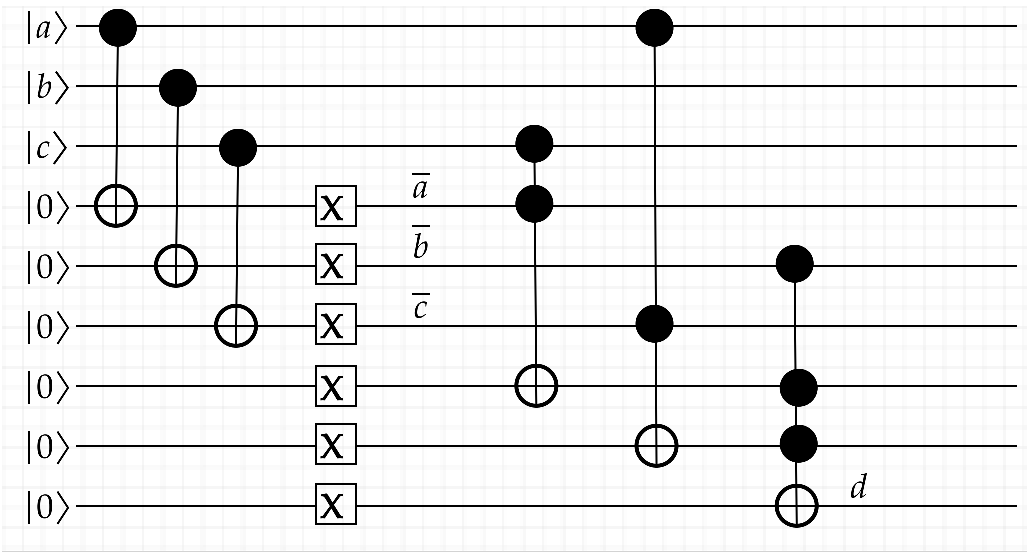
Here is a sizable example of this theorem. Consider the following circuit of NAND gates from the blog article "Implementing Logic Functions Using Only NAND or NOR Gates" by Max Maxfield:



There are six important gates at the very beginning that are not colored or emphasized. Those are the bit-splitters, which collectively make three copies of each of the inputs A, B, C. Doing so is trivial in classical circuits but needs attention in quantum circuits, where we shall see that copying is essentially allowed only on basis states and only with extra "helper" qubits initialized to zero (that is, to $|0\rangle$). Indeed, the CNOT gate can do the copies by itself, because for $b = 0, 1$ we have

$$\mathbf{CNOT} \cdot \big(|b\rangle \otimes |0\rangle\big) \;=\; |b\rangle \otimes |b\rangle \,.$$

Again, this is only true for the basis states $|b\rangle = |0\rangle$ and $|b\rangle = |1\rangle$, not for a general qubit value in place of $|b\rangle$. The pink gates use two copies fed into the same NAND gate only to make a unary NOT gate, so we can economically use the $\mathbf{X}$ gate for that. We can also use $\mathbf{X}$ to change a helper qubit from its $|0\rangle$ initialization to give $|1\rangle$ coming into the target of the Toffoli gate, which is what we need to make it simulate NAND. The green gate at right is a 3-way NAND, for which we fabricate a "4-way Toffoli" gate---or put another way, a triple-controlled bit-flip. Here is the circuit:



(We will later mirror the gates except the last one giving the function value d in order to reset the ancilla qubits 4--8 to $|0\rangle$, so that all qubits except the last keep their given basis values.)

The "quantum extra", beginning with using the Hadamard gate to create superpositions, is what promises to take us beyond classical computing.

Interlude: Larger Unitary Matrices

[This will cover Section 3.6. The purposes are:

- to introduce the notion of an (undirected) **graph**.
- to illustrate how the **adjacency matrix** A_G of a **regular** graph G is **doubly stochastic**, which is the classical-probability analogue of being unitary.
- to show a case where A_G can be converted into a 4×4 unitary matrix by making some entries negative (and then scaling every entry by the same constant factor).
- to show a different case of a regular graph---the six-node prism graph---where this cannot be done.
- to talk about larger unitary matrices---how they technically arise by tensor products with the identity matrix when you have n qubits, but why in practice we can break every circuit down into **gate matrices** of Toffoli size or smaller.

This also sets up some motivation for Chapter 16, the first "topics" chapter.]

Quantum Circuits on n Qubits

[This will skim over the initial part of chapter 4 with not so much emphasis on complexity and Turing machines, making a beeline for the examples in section 4.5.]