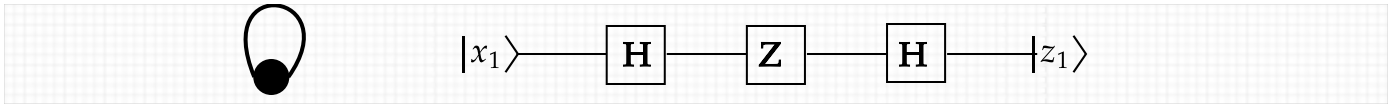
CSE610 Week 6: Visualizing Small Quantum Systems

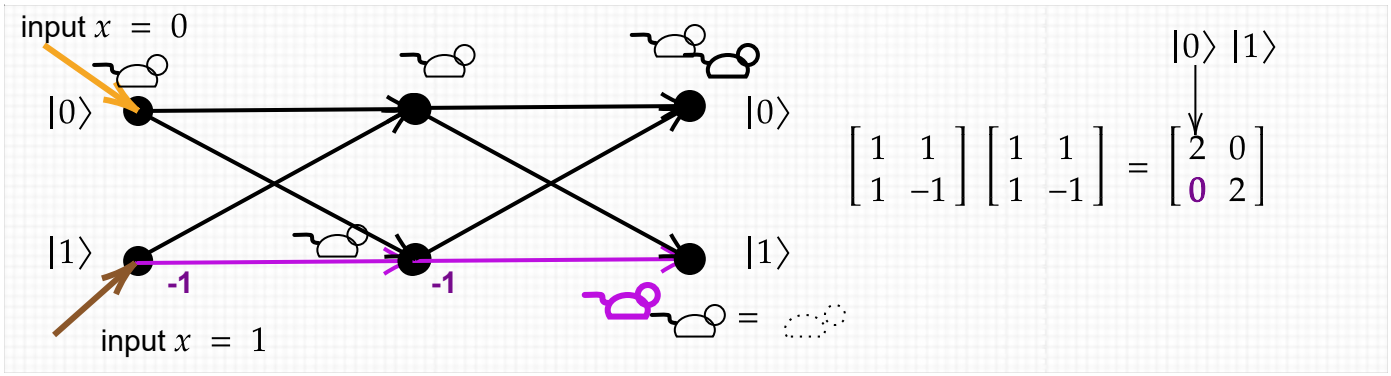
Can we build any interesting things with just a few qubits? Yes, in fact. The first example will be new but relatively simple.

Definition: A **graph state circuit** on n qubits consists of an n-qubit Hadamard transform (i.e., $\mathbf{H}^{\otimes n}$), then some number of $\mathbf{Z}$ and $\mathbf{CZ}$ gates, then a final $\mathbf{H}^{\otimes n}$.

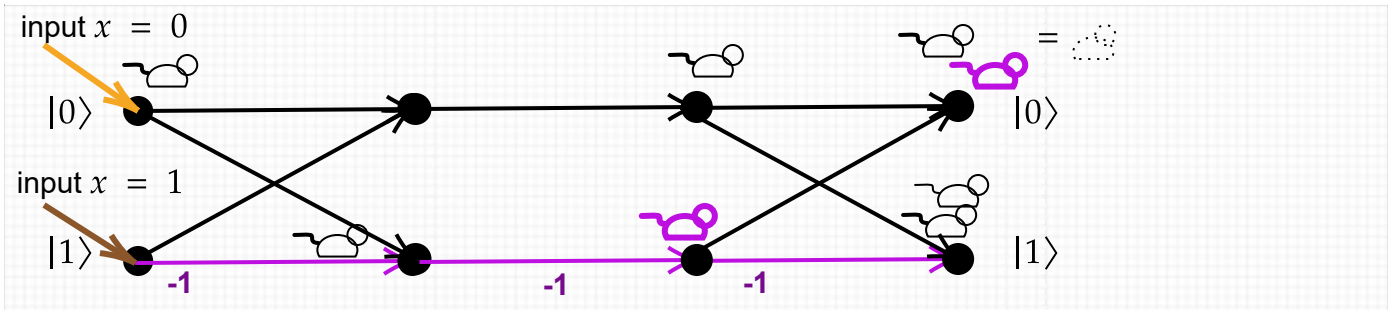
Each qubit is a node. A $\mathbf{CZ}$-gate connecting qubits i and j gives an undirected edge between nodes i and j. A $\mathbf{Z}$-gate on line i denotes a self-loop at node i. The simplest nonempty graph has just one node with a self-loop:



We have seen the equation $\mathbf{HZH} = \mathbf{X}$. How is this reflected when we visualize the quantum properties? There is onlyone change from the "maze" for two $\mathbf{H}$-gates canceling, which was:



The change is to insert a stage that again has a –1 on the $|1\rangle$ basis value but no "crossover":

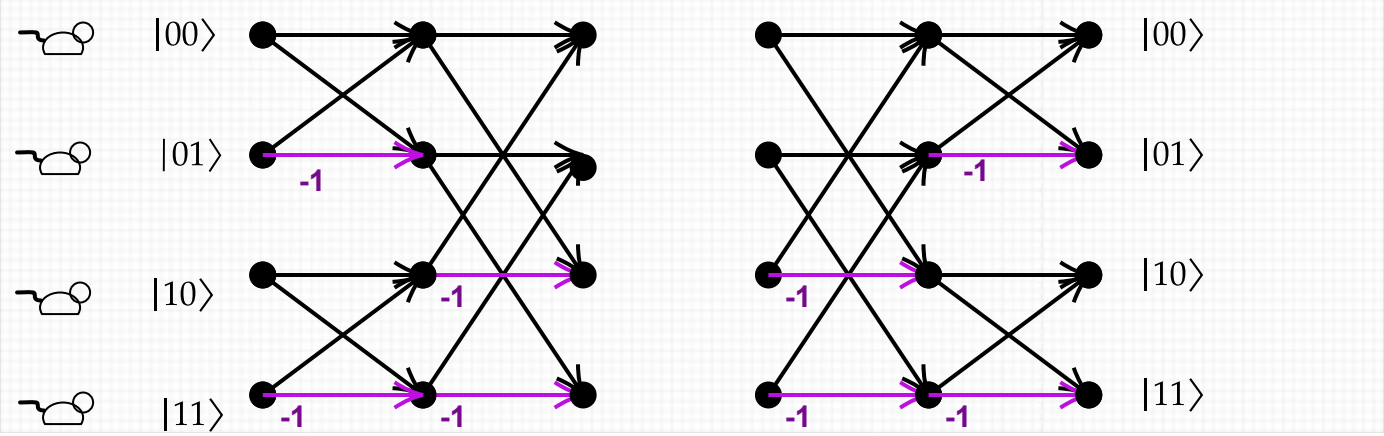


This time, when "Phil" starts running from $|0\rangle$ at left, the "mice" cancel at $z = |0\rangle$ and amplify at $z = |1\rangle$. And on input $x = |1\rangle$ they output the basis state $|0\rangle$. The result is Boolean NOT, i.e., $\mathbf{X}$.

[Footnote: A basic outcome $|z\rangle$ for the circuit C on input x has amplitude $\langle z|U_C|x\rangle$, not $\langle x|U_C|z\rangle$ as I've once been guilty of writing. Perhaps the diagrams should write the bra-form, $\langle 0|$ and $\langle 1|$ and so

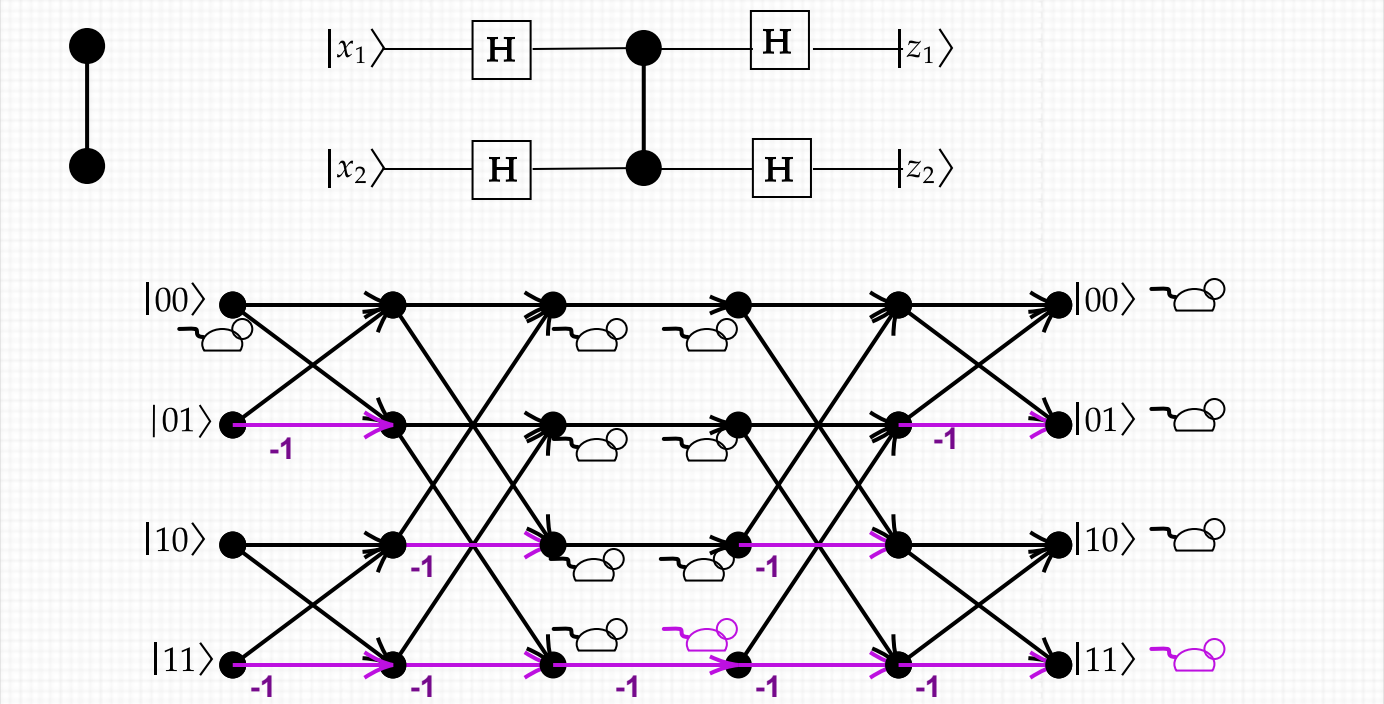
on, for z at right to emphasize this. But we've identified the ket-form with the notion of "outcome"; this is the form that would be given as input to a further piece of the circuit. This dilemma is another reason why Lipton and I first tried for a "handedness-free" approach.]

For graph state circuits of 2 nodes we need 2 qubits. The Hadamard transform of two qubits is diagrammed as at left and right. It does not matter what order the two $\mathbf{H}$ gates go in.



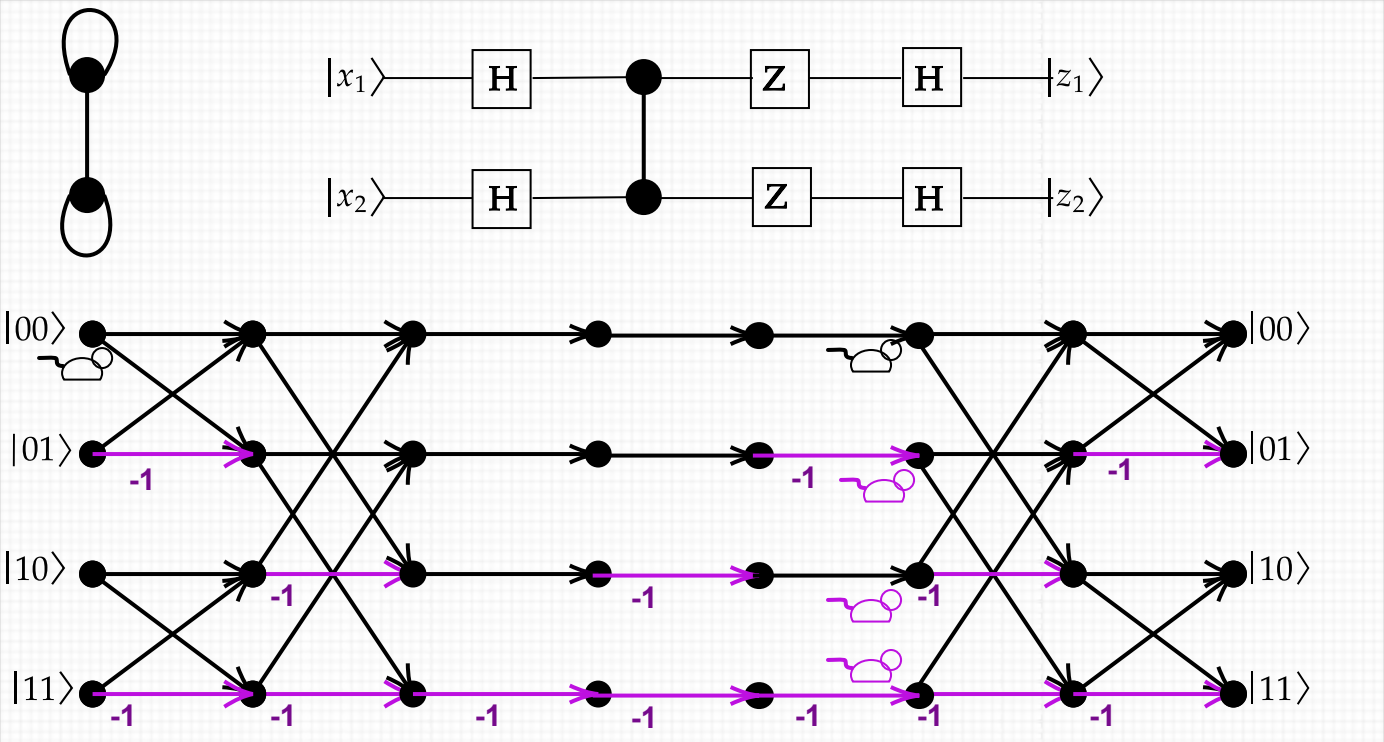
Note that the mouse running from $|00\rangle$ encounters no phase change, nor mice ending at $|00\rangle$ regardless of origin. This simply expresses that the Hadamard transform (and the QFT too) have every entry $+1$ (divided by the normalizing constant $R = \sqrt{2^n}$) in the row and column for $|00\rangle$. We will focus on the amplitude of getting $|00\rangle$ as output given $|00\rangle$ as input. If G is the graph, C_G the graph-state circuit, and U_G the unitary operator it computes, then the amplitude we want is $\langle 00|U_G|00\rangle$.

The simplest two-node G has a single edge connecting the two nodes. This introduces a single $\mathbf{CZ}$ gate between the qubits standing for the nodes.



If we take the two Hadamard gates away from line 1, then we have $\mathbf{H}\,2\,\mathbf{CZ}\,1\,2\,\mathbf{H}\,2$, which is equivalent to $\mathbf{CNOT}$. But with them, we get equal superpositions once again. Most in particular, the amplitude of $\langle 00|U_G|00\rangle$ is nonzero. [The lecture also noted how $\frac{1}{2}[1,1,1,-1]^T$ is a fixed point of $\mathbf{H}^{\otimes 2}$ and found some other fixed points of parts of the circuit, including one that was equal up to multiplication by the unit scalar -1.]

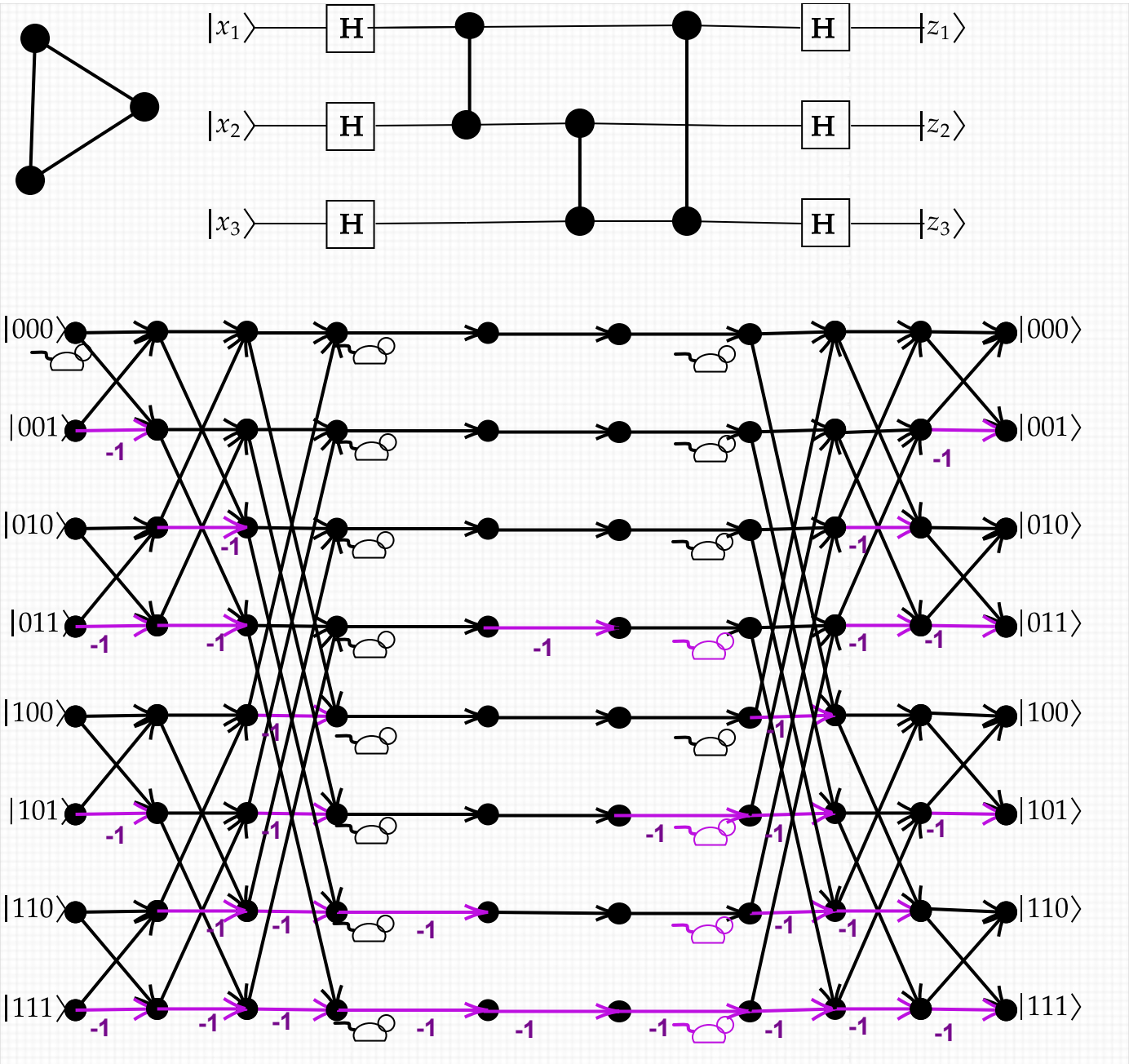
Now let's try a graph that adds a loop at each node. We can call it the "Q-Tip" graph:



The -1 phase shifts for the $\mathbf{Z}$ gates go on the basis states that have a 1 on line 1 or 2, respectively. Now the amplitude value $\langle 00|U_G|00\rangle$ is negative. Its sign does not affect the probability and the state still gives an equal superposition.

It does not matter whether we put the $\mathbf{Z}$ gates "before" or "after" the $\mathbf{CZ}$. The diagonal matrices all commute, and this is clear from how the paths go straight across without branching. We could simply make the whole graph into one diagonal gate with phase shifts that multiply the -1 factors along each row. A related thing to note is that if we repeat an edge or loop, then the two cancel completely. It's as if we have a graph with edges defined by even-odd parity rather than number.

Now let's try a three-node graph, the triangle:



For computing the amplitude $\langle 000|U_G|000\rangle$ it is not necessary to follow the "mice" through the Hadamard parts of the "maze". The mice entering the graph part from $x = |000\rangle$ are all positive, and the mice going to $z = |000\rangle$ will not change color once they leave the graph. So we need only track the middle portion and count how many mice are + and how many are −. For the triangle graph, the answer is: four of each. They **cancel**. So $\langle 000|U_G|000\rangle = 0$.

This leads us to more insight and a strategy for determining this amplitude for a general n-node graph $G = (V, E)$:

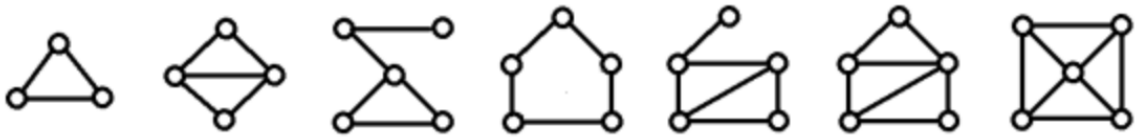
- Every basis state $|x\rangle$ with $x \in \{0,1\}^n$ corresponds to a **2-coloring** χ_x of the vertices. Say a node u is **black (B)** if $x_u = 1$, white (W) if $x_u = 0$. (The Greek letter χ (*chi*) looks like an X and

indeed X is its capital form, but the Greek letter that sounds like English X is ξ (*xi*) with capital Ξ. The χ gives the *ch* in *chromatic*. Well, we can say that the binary string x "is" the coloring χ.)

- For any edge $(u, v) \in E$, the edge contributes a -1 in its $\mathbf{CZ}$ gate if both u and v are colored **B**. Call it a **B-B** edge.
- Therefore, a coloring gives a -1 net contribution if it gives G an odd number of B-B edges.
- The amplitude value $\langle 0^n | U_G | 0^n \rangle$ is positive if fewer than 2^{n-1} (i.e., half) the colorings create an odd number of B-B edges, zero if exactly half do, negative if more.

Whether *one* amplitude is positive or negative does not matter so much in quantum up to equivalence under scalar multiplication. (My lecture demo'ed some examples.) But patterns of signs between different amplitudes $a_z|z\rangle$ of possible outcomes z may have further significance.

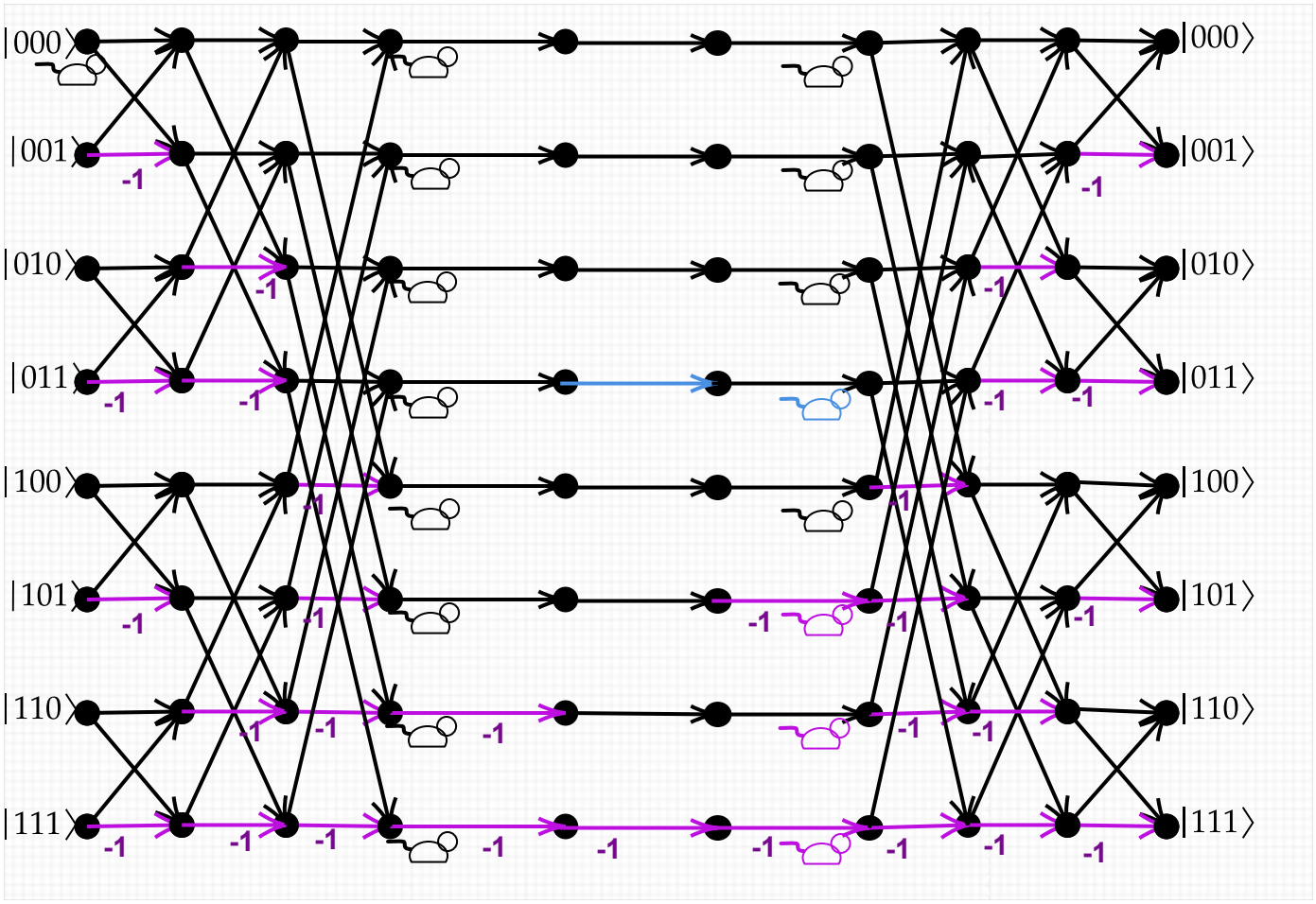
Whether the amplitude is *zero*, however, is absolute. I call a graph G "**net-zero**" if $\langle 0^n | U_G | 0^n \rangle = 0$. Above we first observed that the single-node loop graph is net-zero. The smallest simple undirected graph (meaning no loops or multiple edges) that is net-zero is the triangle. Here are all such graphs up to five vertices:



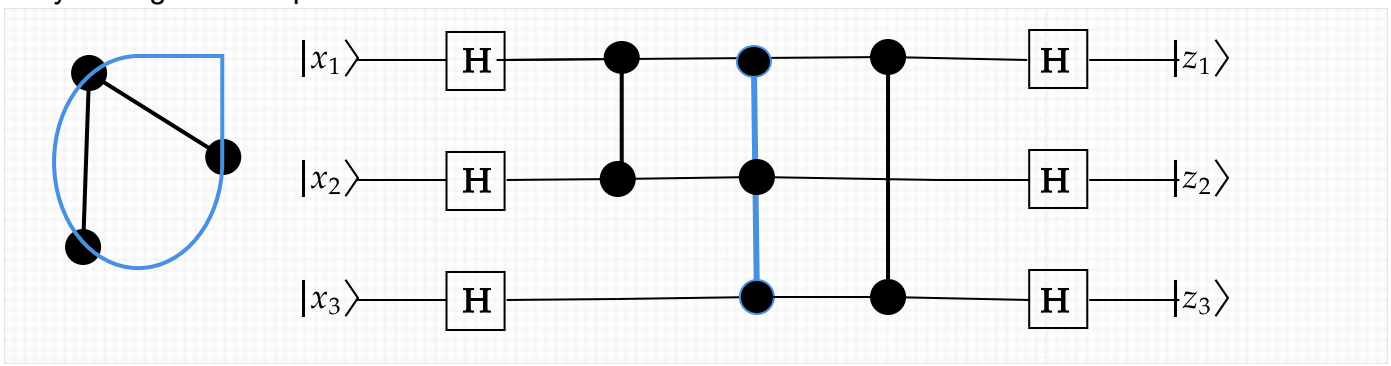
I do not see any simple way to tell "visually" whether a graph is net-zero. My recent PhD graduate Chaowen Guan and I improved the known running time to decide this algorithmically from $O(n^3)$ to whatever the time to multiply two $n \times n$ matrices is (currently $< O(n^{2.37286})$). The algorithm works by converting the graph-state circuit into a quadratic equation of a kind that converts into a linear equation in $O(n)$ variables, whose solutions can be counted in yea-much time. But a simple, more-direct criterion for a graph to be net-zero could give a practically much better algorithm. Guan and I wrote about this on the GLL blog at

https://rjlipton.wpcomstaging.com/2019/06/10/net-zero-graphs/

Some generalizations of graph-state circuits can be handled with equal efficiency. We can simulate $\mathbf{CNOT}$ gates since $\mathbf{CNOT}\ i\ j$ is equivalent to $\mathbf{H}\ j\ \mathbf{CZ}\ i\ j\ \mathbf{H}\ j$. The extra $\mathbf{H}$ gates take things outside the realm of graph-state circuits as strictly defined, but keeps them within the class of so-called **stabilizer circuits**, or equivalently, **Clifford circuits**, to which the same $< O(n^{2.37286})$ runtime applies (for getting any one amplitude, that is). The gates allowed in these circuits are $\mathbf{H}$, $\mathbf{CNOT}$, $\mathbf{S}$, $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{CZ}$, but notably not $\mathbf{Tof}$, $\mathbf{T}$, or $\mathbf{CS}$. But there are other tweaks that seem to be easy to bring within our framework, yet yield hard problems. Consider:



The only change was in the middle column, removing the -1 from the row for $|011\rangle$. The middle column now "fires" only when all 3 bits are 1, i.e., for the component of $|111\rangle$ in any state. This is the action of the double-controlled Z-gate, **CCZ** (which is really a triple control of a 180° phase shift). It is easy to diagram in a quantum circuit:



In graph-theoretic terms, this has replaced the edge $(2,3)$ by the **hyper-edge** $(1,2,3)$, thus creating a **hypergraph**. The effect of changing only the color of the mouse in row 4 (for $|011\rangle$) may seem small, but it has a wild effect on the state vector. Now $z = |000\rangle$ has 5 positive paths from $x = |000\rangle$ instead of 4, so its amplitude is $\frac{5-3}{8} = \frac{1}{4}$. Six other components have amplitude $\frac{1}{4}$, and they collectively have $\frac{7}{16}$ of the probability. The other has 7 positive paths to 1 negative, and so amplitude $\frac{7-1}{8} = \frac{3}{4}$ which squares to $\frac{9}{16}$. Note that the previous amplitude was $\frac{6-2}{8} = \frac{1}{2}$ which squares to just $\frac{1}{4}$, so flipping just

one path of eight made a $\frac{5}{16}$ difference to the probability, more than one might expect. The $\mathbf{CCZ}$ gate could likewise be in any order---the gates commute so there is no element of time sequencing until the final bank of $\mathbf{H}$ gates. The middle part is "instantaneous."

This little illustration of wildness sits over a more general point. The equation resulting from having the $\mathbf{CCZ}$ gate changes from quadratic to cubic. The trick to make it linear no longer applies. Counting solutions to this kind of cubic equation is NP-**hard**. In fact, sandwiching the $\mathbf{CCZ}$ gate between two $\mathbf{H}$ gates (on any one qubit line) gives the Toffoli gate (with target on that line). So $\mathbf{CCZ}$ goes outside the Clifford ambit and gives a universal gate set.

Deutsch's Algorithm

David Deutsch, drawing on two papers by Feynman and other sources, introduced quantum computing while he and I were graduate students at Oxford in the mid-1980s. At first, he claimed quantum computers could solve the Halting Problem in finite time. Fellows of Oxford's Mathematical Institute refuted the claim. But it was not crazy: a year ago it was proved that a binary quantum system of "interactive provers" ***can*** (kind-of-)solve the Halting Problem in finite time. (My review of the paper is at https://rjlipton.wordpress.com/2020/01/15/halting-is-poly-time-quantum-provable/) Per my memory of observing some meetings about it, the gap in Deutsch's argument had to do with properties of probability measures based on infinite binary sequences.

So Deutsch fell back on something less ambitious: demonstrating that there was a "very finite" task that quantum computers can do and classical ones cannot. (Well, unless the playing field is leveled for them...but before we argue about it, let's see the task.) The task is a **learning problem**, a kind of interaction we haven't covered until this last day. Instead of "input x, compute $y = f(x)$", a learning problem is to determine facts about an initially-unknown entity f that you can **query**.

1. **Oracle Turing machines** give a classic way to define this kind of problem. For oracle functions f or languages A drawn from a limited class---such as subclasses of the regular languages---can we design an OTM M that on input 0^n (for large enough n) can distinguish what A is in time (say) polynomial in n? The computation $M^A(0^n)$ can learn about A by making queries y on selected strings y and observing the answers $A(y)$.
2. One can also define **oracle circuits** that have special **oracle gates** with some number m of input wires and enough output wires to give the answer $f(y)$ on any $y \in \{0,1\}^m$.
3. An ordinary electrical test kit behaves that way. It is a circuit with a place(s) for you to insert one or more (possibly-defective) electrical components A. The test results should diagnose electrical facts about A.
4. Quantum circuits for all of the Deutsch, Deutsch-Jozsa, Simon, Shor, and Grover algorithms work this way. They involve an **oracle function** $f : \{0,1\}^n \to \{0,1\}^r$ given in **reversible form** as the function $F : \{0,1\}^{n+r} \to \{0,1\}^{n+r}$ defined by:

$$F(x,z) = (x, f(x) \oplus z).$$

Usually z is 0^r and the comma is just concatenation (i.e., tensor product) so the output is just $xf(x)$. In the simplest case $n = r = 1$, F is a two-(qu)bit function. Some examples:

- If f is the identity function, $f(x) = x$, then $F(x,z) = (x, x \oplus z) = \mathbf{CNOT}(x,z)$.
- If $f(x) = \neg x$, then $F(x,z) = (x, x \leftrightarrow z)$: $F(00) = 01$, $F(01) = 00$, $F(10) = 10$, $F(11) = 11$.
- If f is always false, i.e., $f(x) = 0$, then F is the identity function.
- If $f(x) = 1$, then $F(x,z) = (x, \neg z)$, so $F(00) = 01$, $F(01) = 00$, $F(10) = 11$, $F(11) = 10$.

These are all deterministic as functions of two-qubit basis states, so they permute the quantum coordinates $1 = 00$, $2 = 01$, $3 = 10$, and $4 = 11$. Recall that **CNOT** gives the permutation that swaps the coordinates 3 and 4, that is, $\mathbf{CNOT} = (3\ 4)$ in swap notation. In full, we have:

$$F_{id} = (3\ 4), \quad F_{\neg} = (1\ 2), \quad F_0 = (), \quad F_1 = (1\ 2)(3\ 4).$$

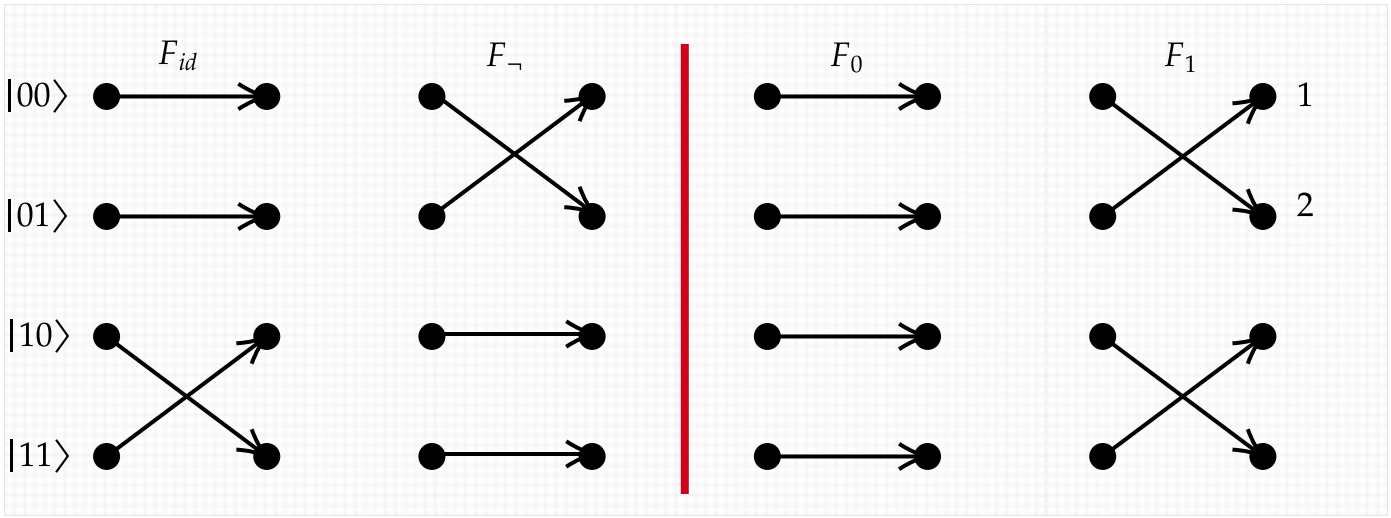
The functions $f(x) = 0$ and $f(x) = 1$ are *constant*. The identity and ¬ functions have one true and one false value each, so they *balance* values of 0 and 1. The question posed by Deutsch is:

How many queries are needed to tell whether f is constant from whether f is balanced?

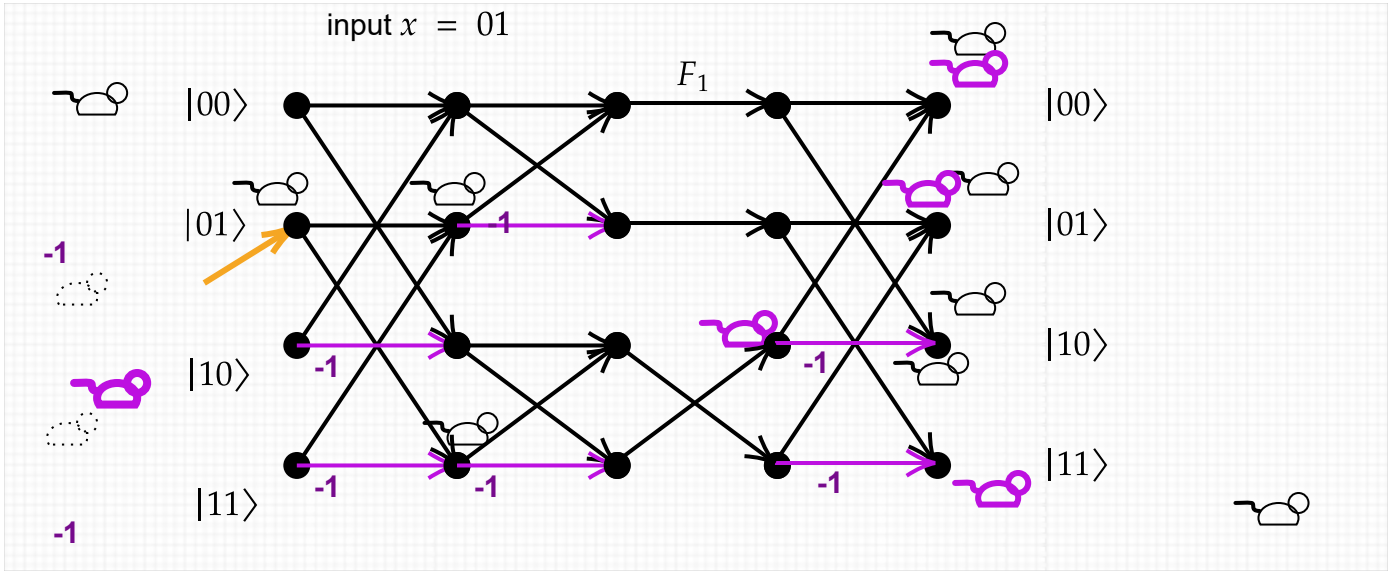
If we just think of f, suppose we try the query $y = 0$ and ask for $f(y)$. If we get the answer "$f(0) = 0$" then it f could be constant-false, but f could also be the balanced identity function. The answer $f(0) = 1$ would leave both constant-true and negation as possibilities. Likewise if we try $y = 1$. The first point is that this impossibility of hitting things with one query carries forward to the way we have to modify the problem for quantum:

How many queries are needed to tell (F_{id} *or* $F_{\neg}$) apart from (F_0 *or* F_1)?

It seems like we have more of a chance because now we can query two things: 00, 01, 10, or 11. Or in the permutation view, we can query $y = 1, 2, 3$, or 4. The problem is that the range of answers we can get is too limited for this to help. $F(1)$ and $F(2)$ can only be 1 ro 2; $F(3)$ and $F(4)$ can only be 3 or 4. So suppose you query $y = 3$ and get the answer 4. Then F could be F_{id} or F could be F_1. The basic problem for a classical algorithm is that every quadrant of the following diagram has both a straight and a cross:



A quantum circuit, however, can make one query to an oracle gate for any of these four functions, and can distinguish a member of the first pair from a member of the second pair by the answer to one qubit after a measurement. The input is not $|00\rangle$ but instead $|01\rangle$; that is, the ancilla is initialized to 1, not to 0. Here is the wavefront ("maze") diagram of how it works:



There is, IMHO, an "unfair" aspect of the comparison. The classical algorithm is being allowed to evaluate the oracle only at basis vectors. The quantum algorithm gets to evaluate it at a linear combination---indeed, it's the state

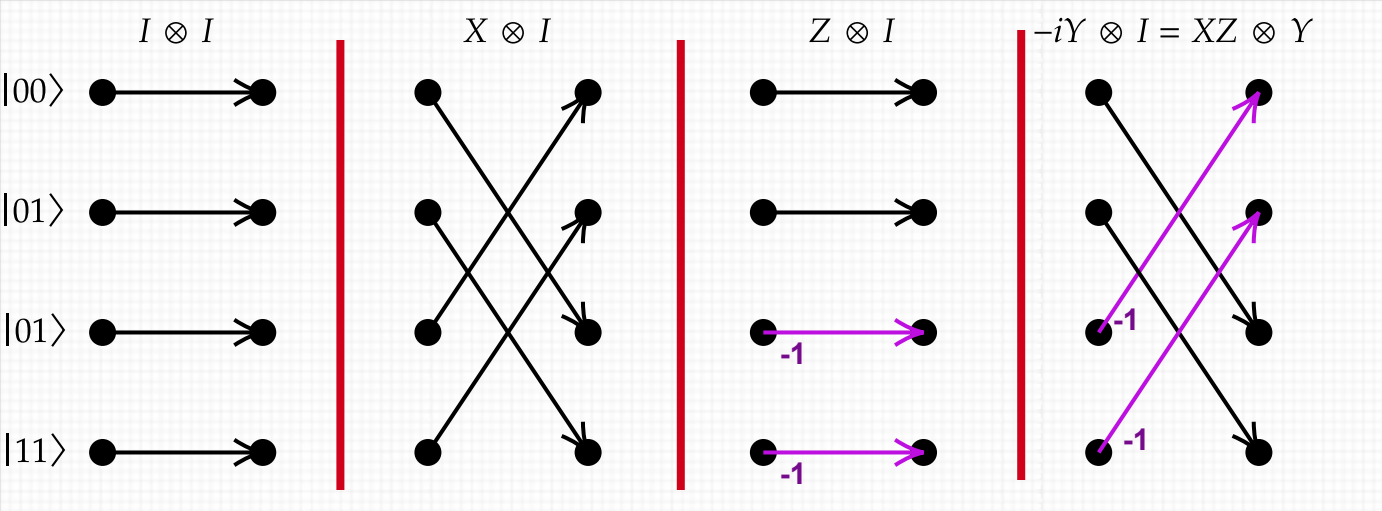
$$|+-\rangle = \frac{1}{2}\big(|00\rangle - |01\rangle + |10\rangle - |11\rangle\big)$$

from the Fri. 12/4 lecture. If we do the kind of linear extension of Boolean logic that was covered as the "Binary Linear Equations" presentation option, then we can solve the problem in one shot classically by evaluating at the point $(1,-1,1,-1)$ and seeing where the $-$ signs end up in the resulting vector. FYI: https://rjlipton.wordpress.com/2011/10/26/quantum-chocolate-boxes/

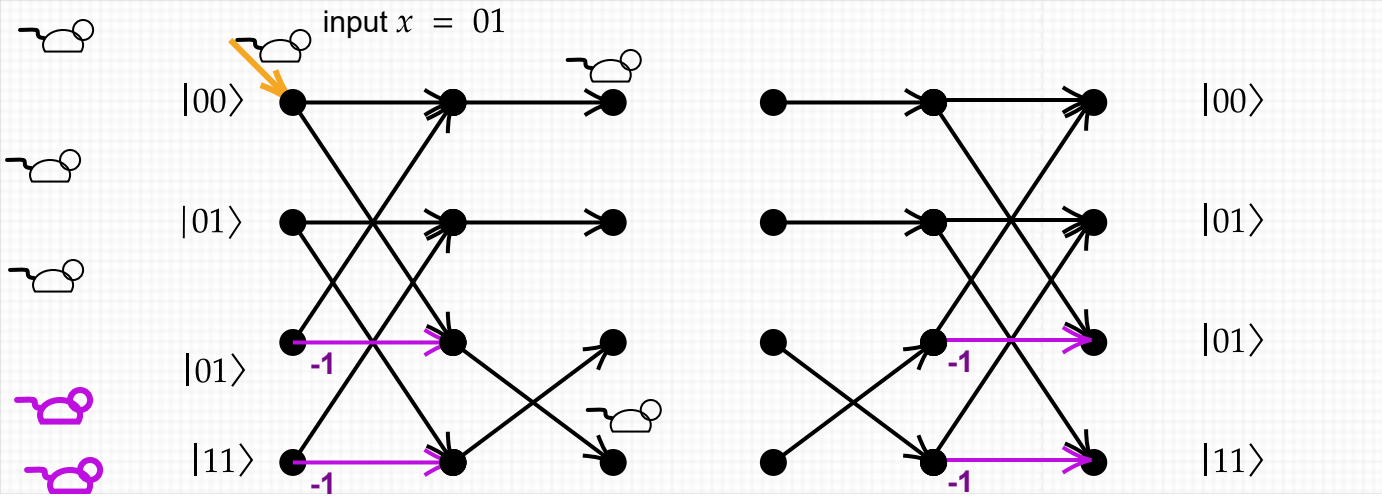
Example: Superdense Coding

It is easy to rig cases F_0, F_1, F_2, F_3 where you can distinguish them exactly by asking one query and measuring both qubits. Just define $F_i(00) = i$, for instance. "Superdense coding" is a case where the rigging has a bit of surprise because it appears to convey 2 bits of information with just 1 qubit of communication. This is impossible by **Holevo's Theorem** that n qubits can yield only n bits of classical information. (Another instance of this that you can input $\sim \frac{1}{2}n^2$ bits of information by choosing the **CZ** gates for edges of an undirected n-vertex graph G in a graph-state circuit C_G on n qubits, one for each vertex, but you can only get n bits of information out by measuring. Hence graph-state encoding is majorly *lossy*.) The rub is that the rigging involves the communicating parties "Alice" and "Bob" already having exchanged 1 bit of information in order to set them up with an entangled qubit pair.

We will regard it as a nice case of the learning problem because it uses the four Pauli matrices. We want to identify one of the following four possibilities **exactly** by the results of **two** qubits.

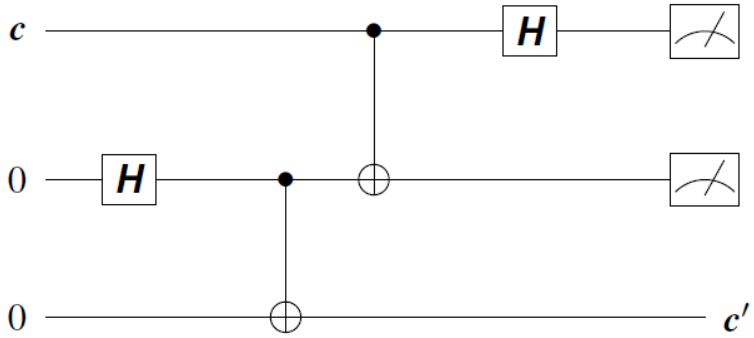


This time the input is $|00\rangle$. To work it out via wavefronts:



Quantum teleportation involves three qubits, two initially owned by Alice and one by Bob. Alice and Bob share entangled qubits as before, whereas Alice's other qubit is in an arbitrary (pure) state $\boldsymbol{c} = a\boldsymbol{e}_0 + b\boldsymbol{e}_1$. Alice has no knowledge of this state and hence cannot tell Bob how to prepare it, yet entirely by means of operations on her side of the lake she can ensure that Bob can possess a qubit in the identical state.

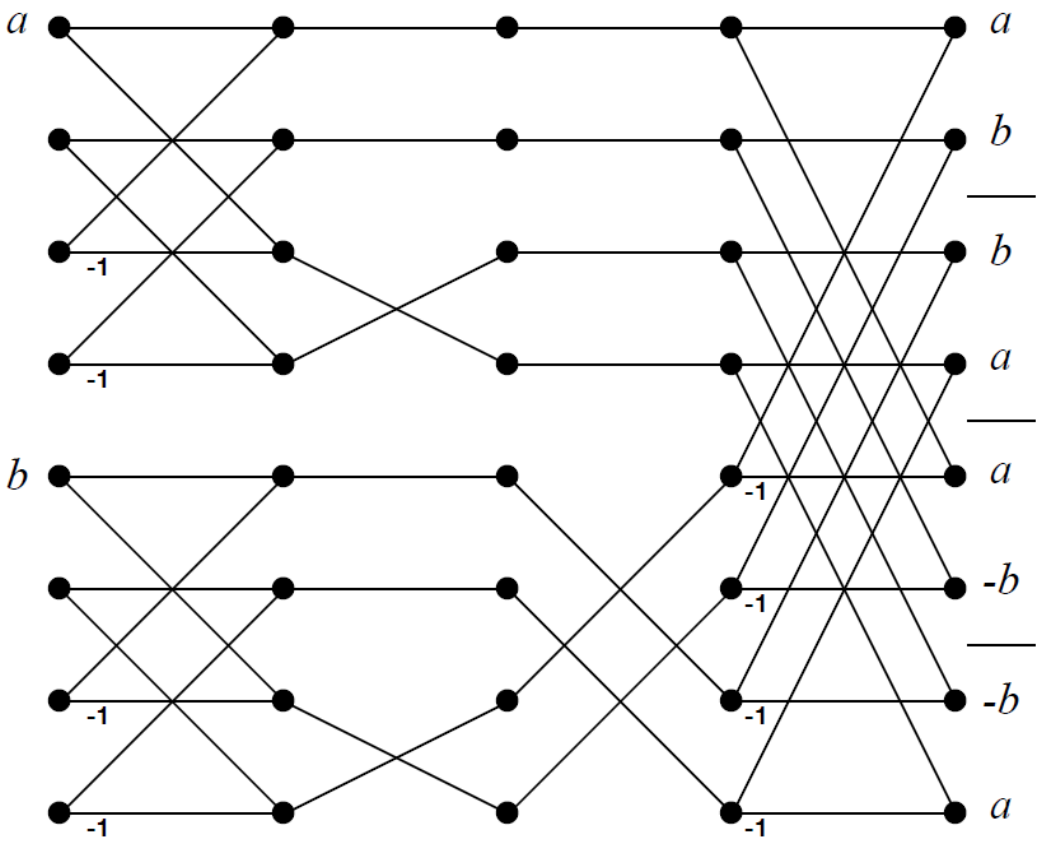
The following quantum circuit shows the operations, with $\boldsymbol{c}$ in the first quantum coordinate, Alice's entangled qubit second, and Bob's last. The circuit includes the Hadamard and ***CNOT*** gates used to entangle the latter two qubits.



With this indexing, the start state is $\boldsymbol{c} \otimes \boldsymbol{e}_{00}$, which equals $a\boldsymbol{e}_{000} + b\boldsymbol{e}_{100}$. After the first two gates, the state is

$$\boldsymbol{c} \otimes \frac{1}{\sqrt{2}} (\boldsymbol{e}_{00} + \boldsymbol{e}_{11}),$$

with Alice still in possession of the first coordinate of the entangled basis vectors. The point is that the rest of the circuit involves operations by Alice alone, including the measurements, all done on her side of the lake. This is different from using a two-qubit swap gate to switch the $\boldsymbol{c}$ part to Bob, which would cross the lake. No quantum interference is involved, so a maze diagram helps visualize the results even with "arbitrary-phase Phils" lined up at the first and fifth rows shown in figure 8.5, which are the entrances for $\boldsymbol{e}_{000}$ and $\boldsymbol{e}_{100}$.



Because Bob's qubit is the rightmost index, the measurement of Alice's two qubits selects one of the four pairs of values divided off by the bars at the right. Each pair superposes to yield the value of Bob's qubit *after* the two measurements "collapse" Alice's part of the system. The final step is that Alice sends two *classical* bits across the lake to tell Bob what results she got, that is, which quadrant was selected by nature. The rest is in some sense the inverse of Alice's step in the superdense coding: Bob uses the two bits to select one of the Pauli operations $\boldsymbol{I}, \boldsymbol{X}, \boldsymbol{Z}, i\boldsymbol{Y}$, respectively, and applies it to his qubit $\boldsymbol{c}'$ to restore it to Alice's original value $\boldsymbol{c}$.

Deutsch-Jozsa Extension

Getting back to Deutsch's Problem, Richard Jozsa added that if you only care about distinguishing ***constant*** functions $f : \{0,1\}^n \rightarrow \{0,1\}$ from ***balanced*** ones, then you can make the classical algorithms require $2^{n-1} + 1$ queries, while the quantum ones can still do it on one query to a completely separable superposed state. This is a conditional problem, called a **promise problem**, in that it only applies when f is in one of those two cases. If f is neither balanced nor constant, then "all bets are off"---any answer is fine, even $|\ \text{¯\_(ツ)\_/¯}\ \rangle$.

The maze diagrams would get exponentially big, but we can track the computations via linear algebra. It is like Deutsch's setup except with $\mathbf{H}^{\otimes n}$ in place of the first $\mathbf{H}$, input $|0^n 1\rangle$ in place of $|01\rangle$, and targets (ignoring the $\sqrt{2}$ normalizers):

- constant $\mapsto |0^n\rangle\big(|0\rangle + |1\rangle\big)$ (instead of $\big(|00\rangle + |01\rangle\big)$, so that 0^n is certainly measured.
- balanced $\mapsto |?\,\rangle$ (instead of $\big(|10\rangle + |11\rangle\big)$, such that 0^n is certainly *not* measured.

The key observation is that for any f, any argument $x \in \{0,1\}^n$, and $b \in \{0,1\}$, the amplitude in the component xb of the final quantum state ϕ is

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{t \in \{0,1\}^n} (-1)^{x \bullet t} (-1)^{f(t) \oplus b}.$$

Here $x \bullet t$ means taking the dot-products $x_i \cdot t_i$ (which is the same as $x_i \wedge t_i$) and adding them up modulo 2 (which is the same as XOR-ing them). Well, when $x = 0^n$ this is always just zero, so the first term is $(-1)^0$ and just drops out, leaving

$$\phi\big(0^n b\big) = \frac{1}{\sqrt{2^{n+1}}} (-1)^b \sum_{t \in \{0,1\}^n} (-1)^{f(t)}.$$

Note that the $(-1)^b$ term is independent of the sum over t, so it comes out of the sum---and this is why we get two equal possibilities in the original Deutsch's algorithm as well. Ths final point is that:

- When f is ***constant***, these terms are all the same, so they ***amplify***---giving $\frac{1}{\sqrt{2}}$ for the constant-false function and $\frac{-1}{\sqrt{2}}$ for constant-true. Both of these amplitudes square to $\frac{1}{2}$ and so together soak up all the output probability, so that 0^n is measured with certainty.
- When f is ***balanced***, the big sum has an equal number of $+1$ and -1 terms, so they all ***interfere*** and *cancel*. Hence 0^n will certainly not be measured.

Added: A *randomized* classical algorithm can efficiently tell with high probability whether f is constant by querying some random strings. If it ever gets different answers $f(y) \neq f(y')$ then definitely f is not constant. (So, under the condition of the "promised problem," it must be balanced.) If it always gets the same answer, then since any balanced function gives 50-50 probability on random strings, it can quickly figure that f is constant. But it is still the case that a deterministic algorithm needs exponentially many queries and hence exponential time.