CSE491/596 Last Lecture (Fri 12/09/22): Signs of Quantum Advantage

The term "quantum advantage", which is IBM's preferred alternative to saying "quantum supremacy", IMHO has a slightly different meaning. It means any ability of quantum hardware or software to out-perform any classical machine on a certain task. The three signature applications regarded as having quantum advantage, though not proven by all means, are:

1. **Shor's Algorithm** for factoring and related problems: exponential time advantage.
2. **Grover's Algorithm** for searching for a witness string: quadratic time advantage.
3. Simulating natural (quantum) processes in real time: exponential time advantage.

The third was Richard Feynman's motivation for quantum computing in contrast to "It From Bit" and the wholly-lexical view of Nature. It expresses "quantum supremacy." Although the quadratic time advantage of Grover's Algorithm is usually written as time $O\left(\sqrt{s(n)}\right)$ to search $s(n)$ locations, it really arises in applications where $s(n) = 2^{r(n)}$ for some linear or polynomial function $r(n)$, so the savings is "only" $O\left(2^{r(n)/2}\right)$ in place of $O\left(2^{r(n)}\right)$. Even with this clarification, I remain a "Grover skeptic"---I think that a true measure of the total "effort" of the search remains $O\left(2^{r(n)}\right)$. I am a "Shor-believer", but both Richard Lipton and I believe the problems solved by Shor's Algorithm belong to classical polynomial time anyway. So for me, wherther there is truly an exponential advantage in quantum devolves upon the question I began with last week.

These algorithms, covered respectively in chapters 11-12, 13, and 18, are beyond our scope, but we can give a taste of where quantum advantage comes from in examples using 2 or 3 qubits, as covered in chapter 8. These examples began with David Deutsch, after he abandoned his initial claim at Oxford in 1985 that quantum computers could decide the Halting Problem of classical computers. Whether the classical setting they allow is commensurate with advantages given to the quantum setting is a matter for debate about these small examples.

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 $\mathbf{CNOT}$ 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 $\neg$ functions have one true and one false value each, so they *balance* values of 0 and 1. The question posed by Deutsch is:

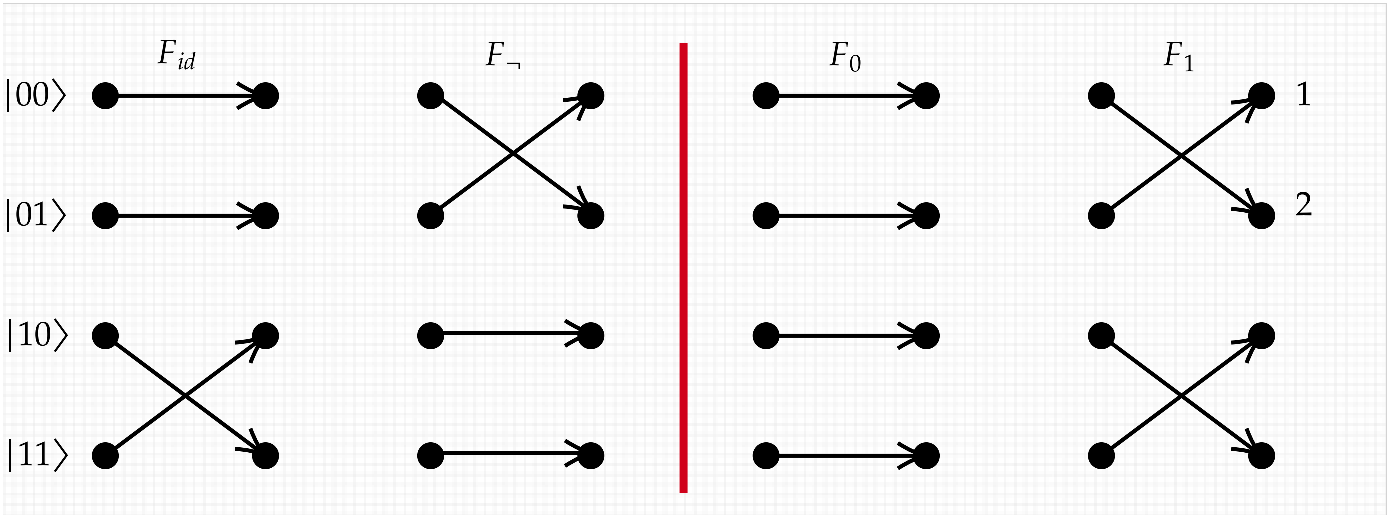
How many queries are needed to tell whether f is constant from whether f is balanced?

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

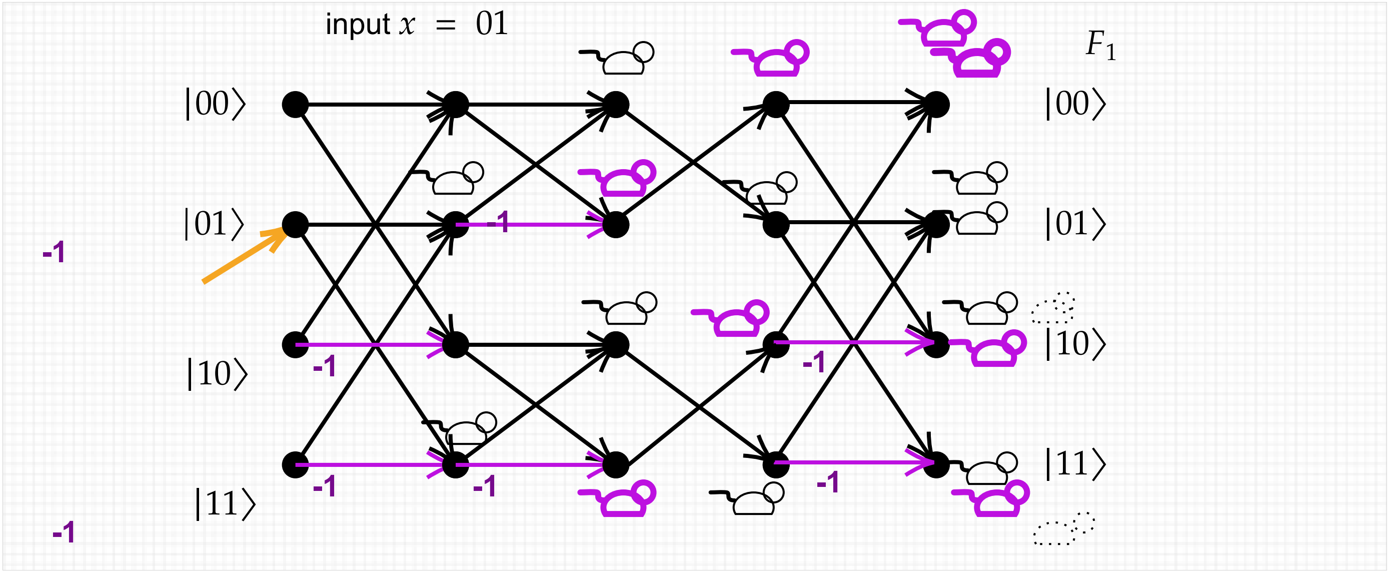
$y = 1$. The first point is that this impossibility of hitting things with one query carries forward to the way we have to modify the problem for quantum:

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

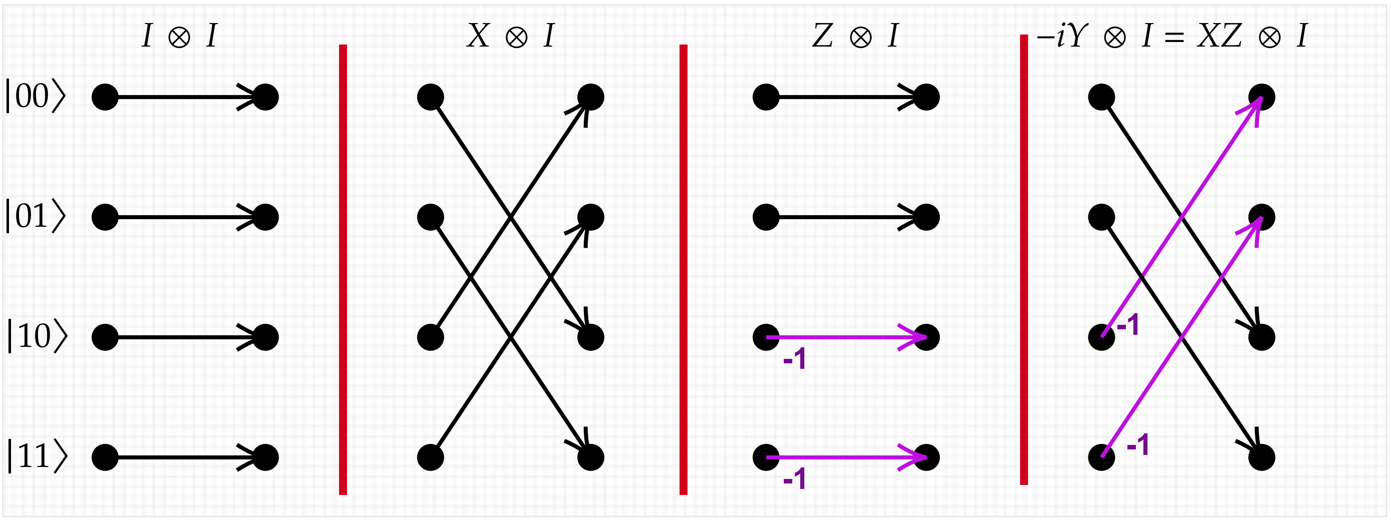
$$|+-\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

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/

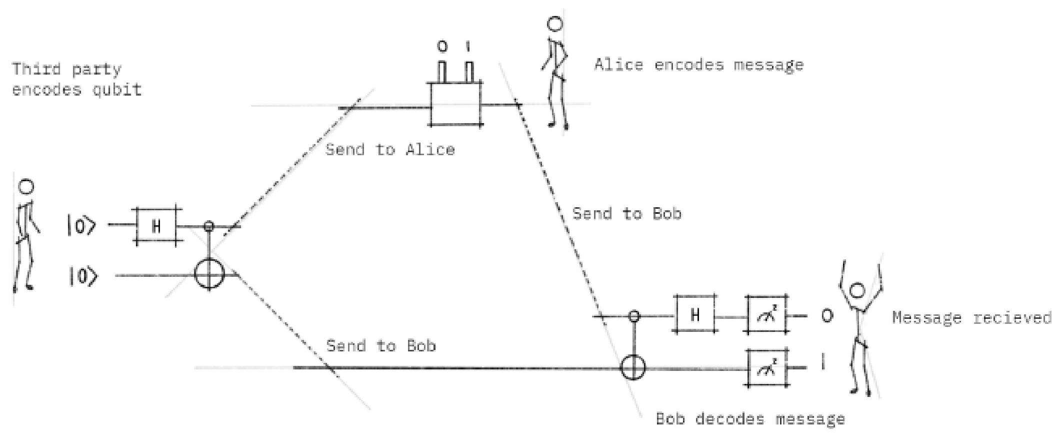
Aside: Superdense Coding

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 $\mathbf{CZ}$ 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.

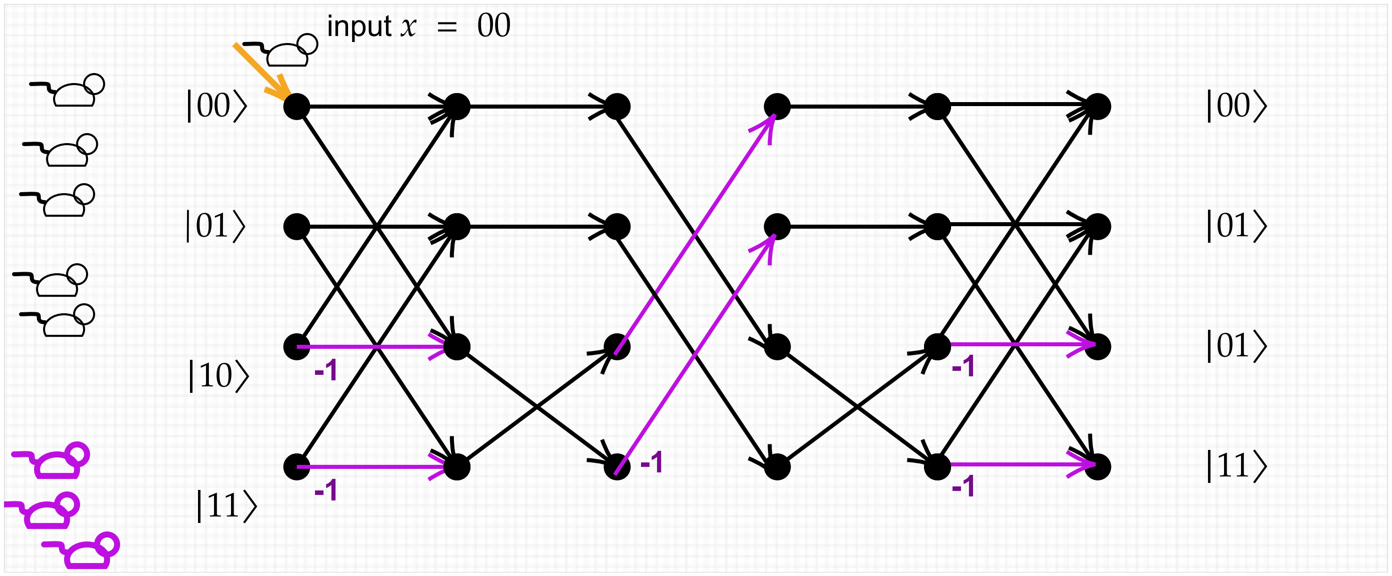


Here is the setting as drawn by IBM's **Qiskit** webpage:



This time the input to the whole circuit is $|00\rangle$. Alice encodes a 2-bit message by applying one of the four Pauli matrices to her qubit, which is the top line of the quantum circuit. She sends the result to Bob, sending him *one qubit*. From Bob's ability to tell exactly which of the four matrices she used, he seems to be getting 2 bits of classical information from the send of *one qubit*. But there is a catch, so that this does not violate the information law that one qubit can carry only one classical bit in any measurement: Bob already got a prior qubit from Charlie that, by virtue of entanglement, counts as a prior conduit from Alice. So he got 2 qubits of information from Alice in total after all. One was in the past, before Alice made her elective choice among 4 options, but the conduit stayed in effect after Alice committed her choice. So the channel carried two qubits of information after all.

What we care about is that this circuit works as advertized. To work it out via wavefronts:



Deutsch-Jozsa Extension

Getting back to Deutsch's Problem, Richard Jozsa added that if you only care about distinguishing ***constant*** functions $f:\{0,1\}^n \to \{0,1\}$ 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.

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 $|0^n1\rangle$ in place of $|01\rangle$, and targets (ignoring the root-2 normalizers):

- constant $\mapsto |0^n\rangle(|0\rangle+|1\rangle)$ (instead of $(|00\rangle+|01\rangle)$, so that 0^n is certainly measured.
- balanced $\mapsto |?\rangle$ (instead of $(|10\rangle+|11\rangle)$, 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

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

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:

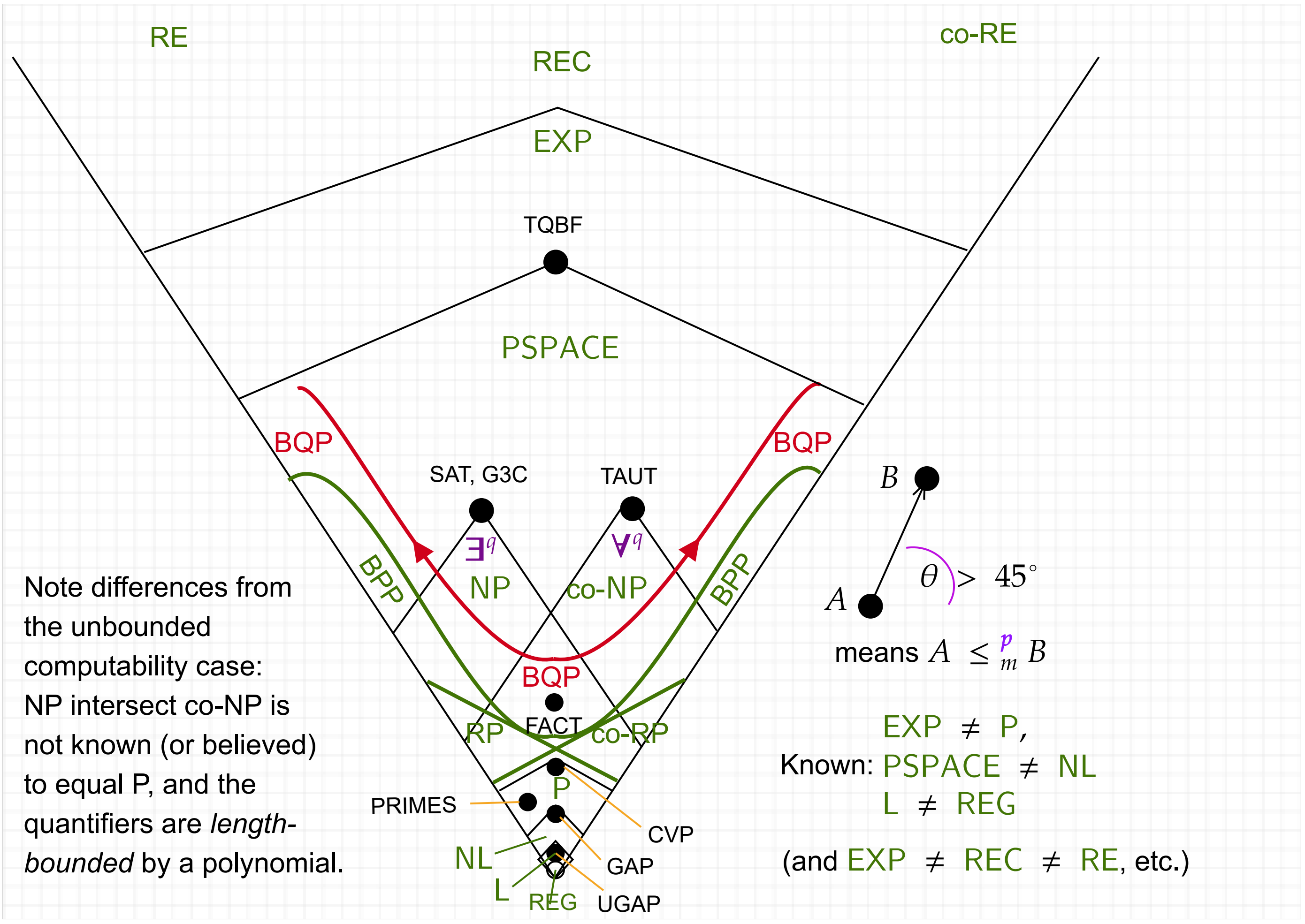
- When f is ***constant***, these terms are all the same, so they ***amplify***---for the constant-false function they give $\frac{1}{\sqrt{2}}$ whereas $\frac{-1}{\sqrt{2}}$ for constant-true. Both of these amplitudes square to $\frac{1}{2}$ and so together soak up all the output probability, so that 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.

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. 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.

Simon's Algorithm

Covering this in full would need a 6th lecture in the last 2 weeks. The big point is that Daniel Simon designed a task that classical randomized algorithms provably require exponential time for---in their restricted classical setting---but where quantum algorithms can do it in polynomial time *using some advantages of their setting*.   FYI, see https://rjlipton.wordpress.com/2011/11/14/more-quantum-chocolate-boxes/ for my attempt to find a real-valued analogue of Simon's problem that a similarly-liberated, but still classical, randomized algorithm can do after all in polynomial time.

Fair or not, and the oracle-dependent nature of the problem notwithstanding, Simon's algorithm is what inspired Peter Shor to realize that if he substituted $\mathbf{QFT}_n$ for Deutsch-Jozsa and Simon's use of $\mathbf{H}^{\otimes n}$, then he could make it impact the (group-theoretic) periodicity in the powering function modulo a number $M = pq$, in a way that allows finding p and q with high probability in roughly-quadratic quantum time. That is, he classified **FACTORING** as belonging to BQP. This plus the **de-randomization** of **PRIMES** from RP $\cap$ co-RP to P in 2002 and **UGAP** (which is the graph-accessibility problem for undirected graphs) from randomized logspace to DLOG in 2004 completes our current top-level knowledge:



Note differences from the unbounded computability case: NP intersect co-NP is not known (or believed) to equal P, and the quantifiers are *length-bounded* by a polynomial.

means $A \leq_m^p B$

Known: EXP $\neq$ P, PSPACE $\neq$ NL, L $\neq$ REG

(and EXP $\neq$ REC $\neq$ RE, etc.)