CSE 599d - Quantum Computing
The Quantum Circuit Model and Universal Quantum Computation

Dave Bacon
Department of Computer Science & Engineering, University of Washington

So far we have talked about quantum computations involving only a few qubits. In this lecture I'd like to begin to discuss how we might scale this up to more than a few qubits and to make something resembling a valid model of computation. To do proper justice to this task, we should probably review the history of the classical theory of computation, and the struggles which the early pioneers in quantum computing went though in order to define a valid model of quantum computing. But we are lucky because a lot of the pitfalls and results have already been overcome, so we won't need to dwell on this work too much but instead get to the more important pragmatic question of what is needed for a universal quantum computer.

So where to begin. The natural place to begin is probably back in 1936 with Alan Turing's seminal paper "On Computable Numbers, with an Application to the Entscheidungsproblem." In this paper Turing defined a model of carrying out a computation which is now called a Turing machine. What is a Turing machine? A Turing machine is a machine which is designed (thought-experiment-like) to mimic a person (computer) who can change the contents of a unlimited paper tape which is divided up into cells that contain symbols based on the local information about the symbols on the tape and an internal state which the person keeps track of. To be more concrete, we can imagine a machine which consists of four main components. The first is an infinite tape which has been subdivided into cells into which symbols from some alphabet can be written and erased. Usually cells that have not been written on are assumed to be filled with no symbol. The second component of the Turing machine is a head. This is a device for reading and writing symbols onto the tape. This head will occupy only one cell of the Turing tape at a time. The head can also move one cell up or down the Turing tape. Third the Turing machine contains a little local memory cell called the state register of the Turing machine. This register will be in one of a finite set of different configurations and represents the "state" of the Turing machine. There are usually special states for the Turing machine, like a halt state which halts the action of the Turing machine. Finally there is the brains of the Turing machine, the controller. The controller of a Turing machine is specified by an action table. This table tells the Turing machine how to act. In particular it gives instructions for, given that the Turing machine state register has a particular configuration, and the symbol read by the head of the Turing machine, what symbol to write on the cell underneath the head, what direction to move the head of the Turing machine, and how to change the state of the Turing machine. It is easy to imagine ourselves as a Turing machine. Indeed many office workers have often pondered if they are nothing more than Turing machines in the great machine known as the corporation! But intuitively at least we can imagine that we can specify instructions to a Turing machine which allow it to carry out important algorithmic tasks.

One of the most important points that Turing made in his 1936 paper was that it is possible to design a Turing machine which is a *universal* Turing machine. Every Turing machine computes some (partial) computable function from possible input strings specified on the symbol tape. So specific Turing machines compute specific partial computable functions. But the action table which describes how a specific Turing machine works could also be written as a series of symbols on the Turing tape. What Turing showed in 1936 was that it was possible to design a Turing machine which could take the symbols on the Turing tape which specify the action table along with the symbols specifying some input on the tape and that this Turing machine would compute the same computable function as the Turing machine specified by the symbols on the Turing tape. This, of course, is just our modern notion of a programmable computer. But what Turing did that was so great was that he formalized a model of computation, the Turing machine, and then showed that the notation of programming could, in effect, be made rigorous.

At this point it is probably useful to introduce what is usually known as the Church-Turing thesis. The question that this "thesis" address is whether the model of what can be computed by a Turing machine is indeed the most general notion of what can be computed in the real world. Over the years there are have been numerous attempts to construct other models of computers. For example the Church in the title of this thesis is not the Church of some religion but the name of a guy who studied another model of computers called the lambda calculus. One might wonder that this model is in any sense more powerful than the Turing machine: that is are there computable functions in this other model which are not computable functions on the Turing machine. For the labmda calculus it was quickly shown that the notion of computable functions for a Turing machine and for the lambda calculus are identical. The Church-Turing thesis is the hypothesis that this will always be true: that every reasonable model of computation is equivalent to the Church-Turing thesis. Now what the heck does reasonable mean here? Well certainly many have argued that this has got something to do with physics. What has been discovered (so far) is that every model which seems reasonable (including quantum computers) satisfies the Church-Turing thesis. Now this brings up two issues. The first is whether there are models that are unreasonable which do not satisfy the Church-Turing thesis. Indeed

there are. For example there are computer models called hypercomputation which explicitly compute non-Turing-computable functions. But we call these models not reasonable because I know of no way to physically construct these devices (although certainly some have tried.) The lesson, of course, is that you should never name something "hyper"-such-and-such. The second issue is whether, since reasonable has something to do with physics (i.e. how the real world works) whether it is possible to "derive" the Church-Turing thesis from the laws of physics. This was one of the original motivations of David Deutsch to consider quantum computers. So far I would say that this task has not been achieved and indeed it is a fun problem to think about.

Now onward to the quantum world! If we are going to begin to contemplate quantum computers, perhaps the best place to begin is to begin by constructing quantum Turing machines: i.e. Turing machines where the Turing tape stores quantum information, the state of Turing machine is also quantum mechanical, the action table is now a description of a valid quantum evolution of the quantum Turing machine. This is indeed one way towards quantum computers and this can be done. However we will not choose this route. Why? Well because just like we don't usually talk about Turing machines when we talk about our everyday classical algorithms because it is too cumbersome, we will use a slightly less formal language to talk about quantum computers. In particular we will work with what is called the quantum circuit model of quantum computation. This model has, in some respects, drawbacks (in particular we will need to talk about uniform quantum circuits), but is mostly a very nice, if not totally transparent, formalism.

I. THE QUANTUM CIRCUIT MODEL

OK, so what is a quantum circuit. A quantum circuit is a prescription for quantum operations we will perform on some set of quantum data. So the first part of a quantum circuit is a description of what the quantum data is. In most cases we will talk about n-qubit quantum circuits. In this case the quantum data are just n quantum bits. But in full generality a quantum circuit model can be used to describe different quantum information, like three level quantum systems, or qutrits. Having defined what the quantum circuit operates on, we can now describe a quantum circuit. A quantum circuit is a prescription for carrying out the following procedure

1. (Preparation) Prepare an input quantum state $|i\rangle$,
$$|i\rangle = |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle \tag{1}$$
where $i_j \in \{0, 1\}$, i.e. we prepare a quantum state in the computational basis. This input is the classical information which describes, for example the specific input to our quantum computation. Often times it also contains padded state which are initiallized in some fiducial manner like all $|0\rangle$ in the computational basis.

2. (Evolution) Apply a unitary evolution due to some set of quantum gates, $U_1, U_2, \ldots, U_r$ For right now just think of a quantum gate as a unitary evolution on k of the n qubits. We will fix k (i.e. make it not depend on n.) Then the full unitary evolution of the quantum circuit is $U = U_r U_{r-1} \cdots U_2 U_1$.

3. (Output) Measure the state of the n qubits in the computational basis . The output of our quantum computation is then the measured state $|k\rangle$. The probability of output $|k\rangle$ given the first two steps is $|\langle k|U|i\rangle|^2$.

Now one thing to be clear about from the onset is that this is not the most general quantum circuit we can construct, but it turns out that every quantum circuit in more general settings can be turned into a quantum circuit of the above form. One particular thing to notice is that often the classical input is sometimes used only to decide what unitaries U_i are implemented. In this case we often think about this input as classical bits which are used as control bits for these unitaries. Often we even get rid of this control notation in the circuit and it is implicit. A second thing to realize is that at this point we haven't really defined a quantum computer: we've just defined a machine which manipulates quantum information (Think about the classical case. We input some information and get some output. At the current stage in our development of circuits, any possible output is attainable. But nothing has been said about a mechanical recipe for producing this manipulation.)

The notion of a quantum algorithm in the quantum circuit model is thus not just the above notion of a quantum circuit, but is instead really embedded in the concept of a "consistent uniform quantum circuit family". Lets take this concept apart word by word. First what is a quantum circuit family. A quantum circuit family is a set of quantum circuits. The elements of this set are labelled by an integer n. n is the number of qubits in the quantum circuit on n qubits of which the circuit labelled by n is a member of. Thus a quantum circuit family is a set of circuits, $\{C_1, C_2, C_3, \ldots\}$ where C_i is a quantum circuit on i qubits.

Next what is a consistent quantum circuit family? A quantum circuit family is consistent if the circuit on n qubits, C_n acting on an m qubit input, $m < n$ padded by $|0\rangle$'s gives the same output as C_m does if we pad this output with $|0\rangle$ as appropriate. Thus if we have a circuit family $\{C_1, C_2, C_3, \ldots\}$, then this family is called consistent if
$$C_n(|i\rangle_m \otimes |0\rangle^{\otimes n-m}) = (C_m|i\rangle_m) \otimes |0\rangle^{\otimes n-m} \tag{2}$$

Now often times our input does not grow in chunks of a single bit. Therefore we often apply consistent to mean growing with our input size (for example ancilla bits may be needed for each new input bit, so we would only require consistency across a growth of two bits in circuit size.)

Finally we come to the most subtle point in our definition of a quantum circuit family. That is the notion of a uniform quantum circuit family. We call a quantum circuit family uniform if there is a classical algorithm for constructing the quantum circuit C_n given classical input n, the size of the quantum circuit. In other words we require that there be some sort of Turing machine which will tell us how to construct each element in the circuit family. This is an important quality to keep track of, because with out it we would be able to bury uncomputable computations in the gate construction itself. This would be very bad, so we require that our circuit families be uniform.

So now we have a notion of what a quantum algorithm is in the quantum circuit model, it is a consistent uniform quantum circuit family. If this was the end of the story, we would be in a wicked state of affairs: for every quantum algorithm we would have to redesign our hardware (the quantum gates) each time we wanted to run a different quantum algorithm. But luckily there is a way out of this pickle of a situation. We do this by defining a *fully universal quantum gate set*:

> A set of quantum gates $\mathcal{G}$ is a *fully universal quantum gate set* if for every $\epsilon > 0$, a sequence of gates from $\mathcal{G}$ can be used to produce a quantum circuit on n qubits to an accuracy of ϵ.

A. Accuracy of Quantum Gates

Well here we've gone ahead and used a concept, the accuracy of a unitary evolution, that we haven't even defined. So let's fix that. Suppose that we desire to implement a unitary U but actually implement the unitary V. Then the accuracy we will use is given by

$$E(U,V) = \max_{|\psi\rangle} ||(U-V)|\psi\rangle|| \tag{3}$$

where $|||v\rangle|| = \sqrt{\langle v|v\rangle}$ and the maximization is over all normalized states. Now why do we use this definition of accuracy? Because if we are given the state $U|\psi\rangle$ and then make a measurement as opposed to being given the state $V|\psi\rangle$ and then make the same measurement, then the above quantity is a good measure of how different the probabilities for the measurement can be. Let's prove this!

Suppose that we measure in a basis $\{|\phi_i\rangle\}$. The probability that we get outcome i if we have performed U is $Pr(i|U) = |\langle\phi_i|U|\psi\rangle|^2$ and if we have performed V is $Pr(i|V) = |\langle\phi_i|V|\psi\rangle|^2$. Thus the difference in probabilities is

$$\begin{aligned} |Pr(i|U) - Pr(i|V)| &= ||\langle\phi_i|U|\psi\rangle|^2 - |\langle\phi_i|V|\psi\rangle|^2| \\ &= |\langle\psi|U^\dagger|\phi_i\rangle\langle\phi_i|U|\psi\rangle - \langle\psi|V^\dagger|\phi_i\rangle\langle\phi_i|V|\psi\rangle| \end{aligned} \tag{4}$$

To simplify notation denote $P = |\phi_i\rangle\langle\phi_i|$. Then

$$\begin{aligned} |Pr(i|U) - Pr(i|V)| &= |\langle\psi|U^\dagger PU|\psi\rangle - \langle\psi|V^\dagger PV|\psi\rangle| \\ &= |\langle\psi|U^\dagger P|\delta\rangle + \langle\delta|PV|\psi\rangle| \end{aligned} \tag{5}$$

where $|\delta\rangle = (U-V)|\psi\rangle$. Next use the triangle inequality, $|x+y| < |x| + |y|$:

$$|Pr(i|U) - Pr(i|V)| \leq |\langle\psi|U^\dagger P|\delta\rangle| + |\langle\psi|PV|\delta\rangle| \tag{6}$$

Recall that the Cauchy-Schwarz inequality is given by $|\langle v|w\rangle|^2 \leq \langle v|v\rangle\langle w|w\rangle$ (A quick proof: $\langle v|v\rangle\langle w|w\rangle = \sum_i \langle v|i\rangle\langle i|v\rangle\langle w|w\rangle$. Assume one of the i points along the $|w\rangle$ direction. Then $\sum_i \langle v|i\rangle\langle i|v\rangle\langle w|w\rangle \geq \frac{\langle v|w\rangle\langle w|v\rangle}{\langle w|w\rangle}\langle w|w\rangle = \langle v|w\rangle\langle w|v\rangle = |\langle v|w\rangle|^2$) Applying the square root of the Cauchy-Schwarz inequality to our different of probabilities yields:

$$|Pr(i|U) - Pr(i|V)| \leq |||\delta\rangle|| \cdot ||PU|\psi\rangle|| + |||\delta\rangle|| \cdot ||VP|\psi\rangle|| \tag{7}$$

But $||PU|\psi\rangle||$ and $||VP|\psi\rangle||$ are both less than unity (because they represent a projector onto a normalized state.) Thus

$$|Pr(i|U) - Pr(i|V)| \leq 2|||\delta\rangle|| \tag{8}$$

from which we can derive

$$|Pr(i|U) - Pr(i|V)| \leq 2 \max_{|\psi\rangle} ||(U-V)|\psi\rangle|| \tag{9}$$

or

$$|Pr(i|U) - Pr(i|V)| \leq 2E(U,V) \tag{10}$$

Thus we see that $E(U,V)$ bounds the maximum difference in probabilities for an outcome of measurement probabilities given that we have implemented V instead of U. Thus it is a good measurement of "accuracy" for unitary evolutions.

One thing you might worry about in quantum computation is that if you build up a quantum circuit from a series of quantum gates, then if these gates are not implemented with perfect accuracy, then the error in our circuit will grow worse than the sum of the errors in implementing individual gates. Here we show that this is not true. Which is nice because if it were not true we would be in trouble! Suppose that we start in the state $|\psi\rangle$. We wish to implement the evolution U_2U_1, but instead we implement V_2V_1. Then the error is given by $E(U_2U_1, V_2V_1)$. To bound this we add a new term

$$E(U_2U_1, V_2V_1) = \max_{|\psi\rangle} ||(U_2U_1 - V_2V_1)|\psi\rangle|| = \max_{|\psi\rangle} ||(U_2U_1 - V_2U_1 + V_2U_1 - V_2V_1)|\psi\rangle|| \tag{11}$$

where we can reexpress as

$$E(U_2U_1, V_2V_1) = \max_{|\psi\rangle} ||(U_2 - V_2)U_1|\psi\rangle + V_2(U_1 - V_1)|\psi\rangle|| \tag{12}$$

Next use the triangle inequality $|||x\rangle + |y\rangle|| \leq |||x\rangle|| + |||y\rangle||$ so

$$E(U_2U_1, V_2V_1) \leq \max_{|\psi\rangle} (||(U_2 - V_2)U_1|\psi\rangle|| + ||V_2(U_1 - V_1)|\psi\rangle||) \tag{13}$$

Since we are maximizing over $|\psi\rangle$ and U_1 and V_1 just represent unitary rotations (and hence could only decrease these two terms), we find that

$$E(U_2U_1, V_2V_1) \leq E(U_2, V_2) + E(U_1, V_1) \tag{14}$$

OK, so what does this tell us? This tells us that the errors in accuracy for a series of two unitary gates *add*. Induction then shows that the errors in accuracy for a series of n unitary gates also add. This is good because it tells us that if we want a unitary to implemented to a precision ϵ, then if this is done with N gates, then we need only implement each gate to a precision $\frac{\epsilon}{N}$. What could this have been? It could have been that one minus the accuracy multiplies. This would mean that the success probability scales exponentially poorly in the errors of the individual gates. But luckily this is not true, so we are safe!

B. Universal Set of Quantum Gates

Now that we have taken the important sidestep of defining the accuracy of a unitary transform, we can return to our definition of a universal set of quantum gates

> A set of quantum gates $\mathcal{G}$ is a *fully universal quantum gate set* if for every $\epsilon > 0$, a sequence of gates from $\mathcal{G}$ can be used to produce a quantum circuit on n qubits to an accuracy of ϵ.

Okay, so do universal sets of quantum gates exist? Let's show one!

Suppose that our gate set consists of the single qubit gates $T = \exp(i\frac{\pi}{8}Z)$ and $M = \exp(i\frac{\pi}{4}Y)$ and the two qubit gate the controlled-NOT C_X. Note that this means that for our quantum circuit of n qubits we can implement H or M on any of the n qubits and the controlled-NOT between any two qubits (actually we only need nearest neighbors to be able to interact by a C_X acting in either direction.) So now the question is, by applying gates from this set of quantum gates on n qubits , can we approximate any unitary evolution on these n qubits.

Lets begin to answer this question by working with just a single qubit. First note that by powering T we can implement any $\exp(i\frac{\pi}{8}kZ)$ where $k \in \mathbb{Z}_8$ and by powering M we can implement any $\exp(i\frac{\pi}{4}Yj)$ where $j \in \mathbb{Z}_4$. Using the fact that $MZM^3 = -X$ this implies that we can implement $\exp(i\frac{\pi}{8}kX)$ for any $k \in \mathbb{Z}_8$ by $MT^{8-k}M^3$. Further, by using this gate for $k = 2$, we can similarly construct $\exp(i\frac{\pi}{8}kZ)$, $k \in \mathbb{Z}_8$. In particular we can implement

$$\exp(i\frac{\pi}{8}Y)\exp(i\frac{\pi}{8}X) = \left(c_{\frac{\pi}{8}}I + is_{\frac{\pi}{8}}Y\right)\left(c_{\frac{\pi}{8}}I + is_{\frac{\pi}{8}}X\right) = c_{\frac{\pi}{8}}^2 I + i(s_{\frac{\pi}{8}}c_{\frac{\pi}{8}}(X+Y) + s_{\frac{\pi}{8}}^2 Z) \tag{15}$$

where $c_{\frac{\pi}{8}} = \cos\left(\frac{\pi}{8}\right)$ and $s_{\frac{\pi}{8}} = \sin\left(\frac{\pi}{8}\right)$. What does this represent? This represents a rotation in the Bloch sphere by an angle θ given by $\cos\left(\frac{\theta}{2}\right) = c_{\frac{\pi}{8}}^2$. This is a rather strange angle this θ. In fact this angle is an irrational multiple of 2π. The proof of this fact is given in Appendix A. Now if we are rotating about some axis $\hat{n}$ by an irrational multiple of 2π, then we can approximate any angle ϕ by a sequence repeated rotations about this angle. Don't worry about the efficiency of this right now! Thus to any accuracy we desire we can approximate gates which rotate around the axis

$$\hat{n}_1 = \frac{1}{\sqrt{c_{\frac{\pi}{8}}^2 + 1}}(c_{\frac{\pi}{8}}, c_{\frac{\pi}{8}}, s_{\frac{\pi}{8}}) \tag{16}$$

Similarly we can perform

$$\exp(i\frac{\pi}{8}X)\exp(i\frac{\pi}{8}Z) = \left(c_{\frac{\pi}{8}}I + is_{\frac{\pi}{8}}X\right)\left(c_{\frac{\pi}{8}}I + is_{\frac{\pi}{8}}Z\right) = c_{\frac{\pi}{8}}^2 I + i(s_{\frac{\pi}{8}}c_{\frac{\pi}{8}}(X+Z) + s_{\frac{\pi}{8}}^2 Y) \tag{17}$$

which is a rotation about the axis

$$\hat{n}_2 = \frac{1}{\sqrt{c_{\frac{\pi}{8}}^2 + 1}}(c_{\frac{\pi}{8}}, s_{\frac{\pi}{8}}, c_{\frac{\pi}{8}}). \tag{18}$$

by the same irrational θ. Thus by powering this operation we can again approximate any rotation about this axis. Finally via a similar construction we can rotate about the axis

$$\hat{n}_3 = \frac{1}{\sqrt{c_{\frac{\pi}{8}}^2 + 1}}(s_{\frac{\pi}{8}}, c_{\frac{\pi}{8}}, c_{\frac{\pi}{8}}). \tag{19}$$

Note that $\hat{n}_1, \hat{n}_2, \hat{n}_3$ are linearly independent. Finally we use the Trotter formula:

$$\left[\exp\left(i\frac{A}{N}\right)\exp\left(i\frac{B}{N}\right)\right]^N = \exp\left(i(A+B)\right) + O\left(\frac{1}{N}\right) \tag{20}$$

Thus to any accuracy we desire we can enact the evolution

$$\exp(i(v_1\hat{n}_1 + v_2\hat{n}_2 + v_3\hat{n}_3)\cdot\sigma) \tag{21}$$

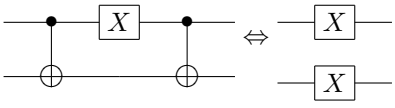
But since the $\hat{n}_i$ are linearly independent we can use this to construct any rotation in $SU(2)$. So what we have shown is that by using only T and M gates we can approximate to any accuracy we desire any rotation in $SU(2)$.

At this point it would be useful to point out that this fact, that we can use two rotations to approximate any element of $SU(2)$, is really not that surprising. Why? Because there are very few finite subgroups of $SU(2)$! In fact the finite subgroups of $SU(2)$ are limited to cyclic groups, dihedral groups, the tetrahedral group, the octahedral group, and the icosahedral group. If you are interested there is an amazing correspondence between these groups and Dynkin diagrams for semisimple Lie groups known as the McKay correspondence. It is hard to do anything to qubits that doesn't turn into everything on qubits!

Okay, so what we have just shown is that using T and M we can approximate any single qubit gate. The next step in showing that we have universal quantum computation is to show how we can use single qubit gates plus the controlled-NOT to approximate any gate on two qubits, i.e. on the $SU(4)$ of these two qubits. In analogy with a generic element of $SU(2)$, every element of $SU(4)$ can be written as

$$\exp\left(i\sum_{\alpha,\beta=0|\alpha=\beta\neq 0}^{3}\theta_{\alpha\beta}\sigma_\alpha\otimes\sigma_\beta\right) \tag{22}$$

Now a very useful identity is the circuit identity



(23)

Using $U\exp(iH)U^\dagger = \exp(iUHU^\dagger)$ this allows us to conjugate a controlled-NOT about a single qubit gate as

$$C_X\exp(i\theta X\otimes I)C_X = \exp(i\theta X\otimes X) \tag{24}$$

Now using single qubit rotations it is possible to conjugate these and to change this two qubit term to any two qubit Pauli operation:

$$\exp(i\theta\sigma_\alpha \otimes \sigma_\beta) \tag{25}$$

where $\alpha, \beta \in \{1, 2, 3\}$. The single qubit rotations we have already seen how to achieve. Thus we can construct any $\exp(i\theta\sigma_\alpha \otimes \sigma_\beta)$ except $\alpha = \beta = 0$. The next step is to use the Trotter formula. This allows us to construct to any desired accuracy a gate in the $SU(4)$ of two qubits.

Finally we can now induct up to obtain any desired $SU(2^n)$ on n qubits. We can do this using a similar construction to the conjugation by the controlled-NOT. If we can implement $SU(2^k)$ on any k qubits, then to show we can implement $SU(2^{k+1})$ we just show how to implement a rotation like $X^{\otimes k+1}$ using the controlled-NOT trick and then again use the local rotations to get any rotation with a $k+1$ non-trivial Pauli terms. Again we complete with the Trotter approximation.

To bring things back together, what have we achieved? We have shown how, using a sequence M and T on every qubit, along with C_X between all qubits, we can implement any possible unitary transform to any desired accuracy ϵ.

But lets return now to the efficiency of our circuit constructions. One thing we might worry about is that if we use different gate sets for achieving universal quantum computation, then these gate sets will lead to very different numbers of gates in order to approximate a particular quantum computation. In particular let's review that original point in our construction, where we took a rotation by an irrational multiple of 2π, $\exp(i\theta\hat{n}\cdot\vec{\sigma})^N = \exp(i\theta N\hat{n}\cdot\vec{\sigma})$. Diagramatically we think about this going around the circle, never repeating, and densely filling the circle. If we assume this filling is rather uniform, then we expect the maximum distance of any point from one of the other elements to go like $O\left(\frac{1}{N}\right)$. Thus the first step in our construction produces gates to an accuracy $\epsilon \approx \frac{1}{N}$.

This is bad! To see why this is bad, suppose that we have a circuit family which requires $f(n)$ gates from a gate set $\mathcal{G}$. To produce this circuit family with a universal gate set one approximates each of these gates $f(n)$ gates with a set of gates from the universal gate set. To produce an accuracy ϵ each gates should be simulated to an accuracy $\frac{\epsilon}{f(n)}$ (remember that the accuracies add.) Now suppose that if, instead of using gates from $\mathcal{G}$, we use gates from a different gate set. Since our construction for simulating the old gate sets goes like $\frac{1}{N}$, this implies that we need to use $N \approx \frac{f(n)}{\epsilon}$ gates to approximate each gate in the old gate set by ones from the new gate set. Thus our circuit will take $O\left(\frac{f(n)^2}{\epsilon}\right)$ gates to simulate with this new gate set. Now this is bad. Why? Well the square is not horrible, although it is not good. But that $\frac{1}{\epsilon}$ is not very good. It means that for every increase in the precision we desire, we have to increase the size of our circuit by a multiplicative amount corresponding to this precision. This is really not a good thing!

But luckily there is a way out of this. And this is a beautiful result due to Kitaev and Solovay:

> (Kitaev-Solovay Theorem): Let the unitary operators $U_1, \ldots, U_p$ generate a dense subset of $SU(d)$. Then any matrix $U \in SU(d)$ can be approximated to within ϵ by $O\left(\log^c\left(\frac{1}{\epsilon}\right)\right)$ elements of $U_1, \ldots, U_p$ and their inverses $U_1^{-1}, \cdots, U_p^{-1}$. ($c$ is a fixed constant.)

Now this is very nice: it means, essentially, that all gate sets we come up with, for fixed size gates, are equivalent. Note that this fixed size requirement is important. This is because we haven't explicitly noted how the number of gates needed scales with the dimension $SU(d)$. In fact it scales poorly with d. But in universal quantum gate sets we are dealing with gates of fixed size (like single qubit and two qubit gates) so this doesn't matter.

Now we won't prove the Kitaev-Solovay theorem (for a long time the actual proof of this theorem was quantum computing folklore: with some unpublished notes floating around, but no explicit write up.) But we can at least get some intuition about why this works. Suppose that we have two non-commuting operators U_a and U_b. When we were just powering a single unitary, the order of this unitary didn't matter. But now that we have two non-commuting operators, the order does matter. Thus for two applications, there are four possible gate combinations, U_a^2, U_aU_b, U_bU_a, and U_b^2. In general there will be 2^n such combinations. If we assume that these gate uniformly fill $SU(d)$, then if the gates don't overlap too much we then expect that we can obtain accuracy $\epsilon = O\left(\frac{1}{2^n}\right)$ for n applications of these gates.

Finally it is nice to note that for a circuit family with $f(n)$ gates from one universal gate set, then the Kitaev-Solovay theorem implies that a universal gate set can execute this circuit family with $O\left(f(n)\log^c\left(\frac{f(n)}{\epsilon}\right)\right)$ elements of a different gate set to accuracy ϵ.

Let's take stock of what we've done. We've showed that there are discrete sets of quantum gates which we can apply and which we can then use to construct to any accuracy we desire any unitary quantum circuit on n qubits. Thus there is a sense in which this set of gates is universal. Further we have seen that the Kitaev-Solovay theorem, guarantees, with some technical conditions, that no matter what gate set we choose, every universal set of quantum gates is equivalent to every other set of quantum gates.

One important thing to realize is that we have used a discrete set of quantum gates. Sometimes in the literature you will see people consider continuous sets of quantum gates. This is, in one sense, kind of crazy because it implies that we can implement each of these gates to any desired accuracy. I'm a theoretician and even I don't believe this is experimentally possible. In fact when we get to the issue of fault-tolerant quantum computation, we will see that the fact that we consider a discrete set of quantum gates is very important. Now in another sense, it is okay to consider continuous sets of quantum gates: certainly what we really mean is that we have a discrete set, to a certain accuracy. Then we can use the results of discrete sets of universal quantum gates to proceed. So this shouldn't bother you too much: unless you want to build a full scale fault-tolerant quantum computer!

APPENDIX A: IRRATIONALITY OF θ

How do we show that the angle defined by $\cos\left(\frac{\theta}{2}\right) = \cos\left(\frac{\pi}{8}\right)^2$ is an irrational multiple of 2π? The argument below is taken from P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "A new universal and fault-tolerant quantum basis", Information Processing Letters, vol. 75, pp. 101-107, 2000 available at `http://lanl.arxiv.org/abs/quant-ph/9906054`. Instead of working with θ lets work with $\alpha = e^{i\theta}$. Now we need to remember some results from algebra. A minimal (monic) polynomial of an algebraic number is polynomial $p(x)$ of smallest degree with leading coefficient unity and coefficients from all coming from the rational numbers $\mathbb{Q}$. $c^2_{\frac{\pi}{8}} = \frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right)$, so

$$\alpha = \frac{1}{2\sqrt{2}}\left[(\sqrt{2}+1) + i(\sqrt{2}-1)\right] \tag{A1}$$

Now clearly the minimal polynomial for α cannot be a polynomial of degree one, because then it would be a rational number. If it is polynomial of degree two, then from the fundamental theorem of algebra, it must be that the polynomial is $(x-\alpha)(x-\alpha^*) = x^2 - (\alpha+\alpha^*)x + 1$. But this polynomial has irrational coefficients in the order one term. Similarly using the fundamental theorem of algebra, a third order polynomial must have its third root real, call it r, so it would be of the form $(x-r)(x-\alpha)(x-\alpha) = x^3 - (\alpha+\alpha^*-r)x^2 + (1+r(\alpha+\alpha^*))x + r$. Since $\alpha\alpha^*$ is rational, this implies that r must also be rational (from the zeroth order term), but then this contradicts the first order term (because $\alpha+\alpha^*$ is irrational). Thus the minimial polynomial for α must be of at least forth order. Indeed it is easy to find this, because we know that it must be of the form

$$p(x) = (x-\alpha)(x-\alpha^*)(x-\beta)(x-\beta^*) = (x^2 - (\alpha+\alpha^*)x + 1)(x^2 - (\beta+\beta^*)x + \beta\beta^*) \tag{A2}$$

which, requires that $\alpha + \alpha^* + \beta + \beta^*$ must be rational. So $\beta = a - \frac{1}{\sqrt{2}} + b$, for some rational a and imaginary b. But $(\alpha+\alpha^*)(\beta+\beta^*)$ must also be rational, and this requires $(1+\frac{1}{\sqrt{2}})(a-\frac{1}{\sqrt{2}})$ is rational which is only true if $a = 1$. Further requiring that $\beta\beta^*$ is rational implies that b must be such that β is on the complex unit circle. There are, of course two solutions for this, and choosing one arbitrarily leads to $\frac{1}{2\sqrt{2}}\left[(\sqrt{2}-1) + i(\sqrt{2}+1)\right]$. This leads to the forth order polynomial

$$p(x) = x^4 - 2x^3 + 2x^2 - \frac{3}{2}x + \frac{9}{16} \tag{A3}$$

This the minimal polynomial for α. For some reason that I cannot figure out the paper of Boykin *et al.* contains a different minimal polynomial.

Now there is a theorem from algebra which states that the minimal polynomial for $e^{i2\pi c}$ exists and is cyclotomic iff c is rational.

A cyclotomic polynomial is a polynomial of the form $\Phi_n(x) = \sum_{k=1}^{\phi(n)}(x - z_k)$ where z_k is the kth primitive n root of unity, $e^{\frac{2\pi ij}{n}}$ with j coprime to n. The first few cyclotomic polynomials are $\Phi_1 = x-1$, $\Phi_2 = x+1$, $\Phi_3 = x^2+x+1$... When n is prime, the cyclotomic polynomial is just $\Phi_p(x) = x^{p-1} + x^{p-2} + \cdots + x + 1$. It is easy to see that there is a recursive formula for cyclotonic polynomials which is given by

$$\Phi_n(x) = \prod_{d|n} \Phi_d(x) \tag{A4}$$

From this and the expression for the cyclotomic polynomials for a prime number we see that a cyclotomic polynomial always has integer coefficients. And this is where we get our contradiction. The theorem says that if c is rational iff a minimal polynomial for $e^{i2\pi c}$ exists and is cyclotomic. But we have seen above that our minimal polynomial has rational coefficients. Hence we have shown that c is not rational!

For completeness it is nice to see a proof of the theorem that if c is rational then a minimal polynomial for $e^{i2\pi c}$ exists and it is cyclotomic. First note that if c is ration, we can express it as p/q where $p, q \in \mathbb{Z}^+$ and $q \neq 0$. Then $\beta = e^{i2\pi c}$ satisfies $\beta^q - 1 = 0$. Thus we have a polynomial $x^q - 1$ which exists and is monic. But it might not be the minimal polynomial. Further we want this minimal polynomial to be cyclotomic. We claim that $x^n - 1 = \prod_{d|n} \Phi_d(x)$. Why is this true? Well every nth root of unity is a primitive dth root of unity for exactly one positive divisor d of n. So we have shown that $x^q - 1 = \prod_{d|n} \Phi_d(x)$. Thus the minimal polynomial will be one of the cyclotomic functions $\Phi_d(x)$. This completes the proof.

ACKNOWLEDGMENTS

The diagrams in these notes were typeset using Q-circuit a latex utility written by graduate student extraordinaires Steve Flammia and Bryan Eastin.