UCSD Summer school notes

The quantum circuit model

1 Introduction

In the previous lecture you saw the basics of quantum information, including the definition of quantum states and unitary evolution. From the point of view of computation there are multiple questions that are raised by the physical model:

- Quantum states and Hamiltonians are described by complex parameters. Can the description of a quantum state, and a quantum evolution, be efficiently discretized in a way that errors do not accumulate too much?
- How do we make sense of the "complexity" of a given Hamiltonian, or the associated unitary evolution, in terms of the computations it can implement? How do we even associate a computational problem to a Hamiltonian?
- Is it possible to define a universal notion of "efficient quantum computation", similar to the classical Turing machine, or is it necessary to take into account the details of the physical system that is used to implement the evolution? Is there a "universal Hamiltonian" whose evolution can emulate the evolution of any other Hamiltonian, as a function of the initial conditions?
- Quantum evolution is unitary, and in particular reversible. Classical computation is not reversible (think of the AND gate). Is reversibility a significant restriction? Can any classical circuit be made reversible? If so, with what overhead?

One of the messages that we aim to carry across during these few days is that there is a unique notion of *universal* quantum computation (analogous to the universal Turing machine), but there are many *restricted* models of quantum computation. Some of these models may be universal, but many are not. These models are often motivated by specific physical implementations, the hope being that they might be easier to realize than a full-blown universal quantum machine. While not being universal, restricted models may still allow for efficient quantum algorithms for classes of problems for which we have no classical algorithms. The diversity of these models makes quantum computation, and quantum algorithms, a much more rich area than you might think!

Deutsch is the first to have considered the notion of a quantum Turing machine (QTM) [Deu85]. Deutsch's paper is a great read, and reminds us of the shock that the introduction of quantum computation might have brought to computer science — in particular, Deutsch discusses the significance of quantum computers for the, at the time widely accepted, Church-Turing thesis.

Deutsch proved that there exists a universal QTM: a fixed machine that has the ability to simulate any other QTM. Deutsch even showed the machine could in principle simulate the evolution of any physical process (as long as it follows the laws of quantum mechanics!), thus laying the basis for his "physical Church-Turing thesis". However, Deutsch's simulation was not efficient. The first to show efficient simulation were Bernstein and Vazirani, in a paper that lay the foundations of quantum complexity theory [BV97]. One of the most striking results in that paper is an oracle separation between BPP and BQP (a class introduced in that paper). This is called the Bernstein-Vazirani algorithm, and we'll discuss it later.

If you think that classical TMs are unwieldy, well, I'll let you imagine what a quantum TM could look like... The model plays an important foundational role, but in practice the most convenient model to work with is the quantum analogue of the circuit model, with gates acting on qubits, etc.

As quantum Turing machines, quantum circuits were also first introduced by Deutsch, who called them "Quantum computational networks" [Deu89]. Deutsch introduced the notion of a quantum gate; in particular, he saw the importance of reversibility and showed that any classical circuits could be made into a quantum one. Yao [Yao93] later showed that quantum circuits are just as powerful as QTMs: for any QTM, input size n, and bound t on the running time of the QTM for inputs of size n, there is a circuit of size polynomial in n and t that computes the same function. (In the same paper Yao initiated the study of quantum communication complexity, a wonderful area on which we'll unfortunately say very little.)

The goal of this lecture is to introduce quantum gates and quantum circuits, discuss universality of gate sets, and define the class BQP.

Readings. There are many excellent introductions to the topic, so we won't be exhaustive in the notes; a good starting point is the book [KSV02]. A good source for definitions of quantum complexity classes, such as BQP and many others, is [Wat09].

2 Quantum circuits

2.1 n qubits

The first thing we need is an appropriate state space. We're all familiar with the notion that a classical circuit operates on n bits: the state space for the circuit is $\{0,1\}^n$. Similarly, a quantum circuit will operate on n qubits. As we saw in the previous lecture, the state space for two qubits is the tensor product of two copies of the state space for a single qubit, $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$. If we repeat this operation n times we get $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$. Thus a quantum circuit operates on unit vectors in $\mathbb{C}^{2^n}$, a space of exponential dimension! Note that by itself this is not such an extraordinary fact, as the state space associated with n bits also has exponential cardinality. The main difference of course is that n qubits can be in the state $|x\rangle$ for any $x \in \{0,1\}^n$, but also $\frac{1}{\sqrt{2}}(|x\rangle - |y\rangle)$, etc. (Note the "$-$" sign: superpositions are not distributions!)

We'll usually take the initial state of a quantum circuit to be the state $|0^n\rangle = |0\rangle \otimes \cdots \otimes |0\rangle$. Sometimes we'll also assume the circuit has an input $|x\rangle$ written on some of the qubits, so the input state may look something like $|x\rangle|0\rangle$, where x is on n qubits, and there are m zero qubits, that we call "ancilla" qubits. Note the omission of the "$\otimes$" symbol, which is regularly omitted for ease of notation.

2.2 Reversible computation

Now that we have a state space we want to act on it! A first step might be to check that we understand how to apply a classical circuit using the quantum notation.

For example, if the circuit just has the simplest gate, a NOT gate, this is just the operation NOT: $|0\rangle \mapsto |1\rangle, |1\rangle \mapsto |0\rangle$. Do you recognize this? It is the Pauli σ_X operation. This operation is unitary, and so we can apply it to a qubit. Now, we might want to apply this to a qubit that is part of a bigger state space of n qubits. It will look like this: $\text{NOT}_i : |x\rangle \mapsto |x'\rangle$, where x' is like x but with the i-th bit flipped. We can write this gate $\sigma_x(i) = \text{Id} \otimes \sigma_x \otimes \text{Id}$, where the first Id is in fact a tensor product of $(i-1)$ 2×2 identities on $(i-1)$ qubits, and the second is on $(n-i)$ qubits. Make sure you understand how to write this as a matrix! Start with $n = 2$.

So far so good. Now what about the AND gate. Well, this should act as

$$\text{AND} : \ |00\rangle \to |0\rangle, \ |01\rangle \to |0\rangle, \ |10\rangle \to |0\rangle, \ |11\rangle \to |1\rangle \ . \tag{1}$$

Do you notice any problem? There are tons of problems! As you now know all quantum operations are unitary. A necessary condition for being a unitary is to be invertible, but the map above is not injective. So (1) doesn't make much sense as a quantum operation. Unfortunately most classical gates, such as AND or OR, are *not* reversible. Any idea how to deal with this?

Here's a trick: any classical gate that maps two bits to one bit can be made reversible, by leaving the input unchanged, but instead writing the output in a third, ancilla register:

$$\text{AND} : \ |x\rangle|y\rangle|c\rangle \mapsto |x\rangle|y\rangle|(x \wedge y) \oplus c\rangle \qquad \forall x, y, c \in \{0,1\}. \tag{2}$$

Can you prove that this is reversible? (Hint: apply it twice.) Note it is important that we XOR-ed the output in the third register, instead of just "writing it" there. We wouldn't be allowed to overwite c, because then information would be lost and the map wouldn't be reversible.

Note how our reversible AND (2) now is a 3-qubit gate. In fact you can show that this is necessary.

Exercise 1. Show that two-bit reversible gates are not universal for classical computation. (Hint: show that any n-bit classical circuit made of two-bit reversible gates implements a linear transformations over $\mathbb{F}_2^n$.)

But three-bit gates are enough. For example, the Toffoli gate (also called CCNOT) is a specific gate that is universal for classical computation. This gate simply XOR's a 1 in the third input if and only if the first two inputs equal 1: it is precisely our reversible AND in (2)! (Exercise: write the Toffoli gate as an 8×8 unitary matrix.)

2.3 Quantum gates

We now understand how to implement any n-bit classical circuit as a sequence of three-qubit unitary (in fact, permutation) gates acting on n qubits. But unitaries can do more things than permutations. What is a good example? The Hadamard transformation! As a single-qubit gate, this implements

$$H : \ |0\rangle \mapsto \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big), \quad |1\rangle \mapsto \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \ .$$

More succinctly, we can write

$$H : \ |x\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} (-1)^{xy} |y\rangle \ , \qquad \forall x \in \{0,1\} \ .$$

Does this formula look anything familiar? Fourier transform! So that's already a funny thing that a quantum circuit can do: if you act on every single qubit using a H gate, then you have a circuit of depth 1 that implements the Fourier transform over $\mathbb{F}_2^n$,

$$H^{\otimes n} : \ |x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \ , \qquad \forall x \in \{0,1\}^n \ .$$

Note how this map does something really weird, which is to compute a global function of the whole input — the inner product with any given string y — and then stores it in the phase. Just because it depends on all bits of the input, the best classical circuit for computing inner products would have depth $\log n$. It may not be immediately obvious if doing this in the phase of a superposition is going to be any useful — but believe me, it is, and we'll see how to take advantage of the quantum Fourier transform for algorithms in the next lecture.

Generalizing from this example, we can define a notion of single-qubit and two-qubit gates: respectively, a 2×2 and a 4×4 unitary matrix U, that can be applied on any one or two out of n qubits by considering the larger unitary $U \otimes \mathrm{Id}$. Note for the case of two-qubit U this is a bit tricky, as the two qubits may not be right next to each other. Make sure you understand what $U \otimes \mathrm{Id}$ looks like as a matrix if, for example, U is a CNOT gate acting on qubits 1 and 3 out of a 3-qubit system. In practice, we generally won't have to explicitly compute such matrices. Instead, we will work directly with circuit representations, with a wire for each qubit — exactly the same as for classical circuits.

Here is an example of a two-qubit gate: the CNOT gate, that flips the second qubit if and only if the first qubit is 1. More precisely, CNOT: $|x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle$. Note that this is in fact a classical gate. Make sure you know how to write it as a 4×4 matrix.

2.4 Quantum circuits

Given a finite set of allowed gates $\{U_1, \ldots, U_k\}$ we can now define a notion of a circuit C over this gate set. By definition a circuit implements a global unitary transformation. Do all unitaries have a circuit representation? Obviously this will depend on the choice of gate set: for example, if there are only single-qubit gates the circuit maps product states to product states, which is not fully general. Moreover, it's clear that for any finite gate set, there are many unitaries that cannot be expressed as a product of gates taken from that set: while there is a continuous number of unitaries, the set of unitaries that we can express using our finite gate set is only countable.

So what if we allow approximation? To answer this we first need to select a proper notion of approximation. It is natural to bound $\|U - U'\| = \sup_{|\psi\rangle} \|(U - U')|\psi\rangle\|$: this is just the operator norm. If $\|U - U'\|$ is small then the two unitaries have approximately the same effect on any state. This is true even when the unitary acts only on a subset of the qubits, as $\|U \otimes \mathrm{Id} - U' \otimes \mathrm{Id}\| = \|U - U'\|$.

Unfortunately, by a simple counting argument we can see that whatever set of elementary gates that we allow, as long as it is finite some unitaries will, at best, require a lot of gates to approximate: the space of n-qubit unitaries has dimension roughly $(2^n)^2$ (a bit less), and Schroedinger's equations allows any one of them to be counted as a possible quantum evolution, at least in principle. However, by simple volume estimation the size of an ε-net over that space, for any reasonable measure of distance, will be of order $(1/\varepsilon)^{2^{\Theta(n)}}$. But the number of circuits involving at most k gates, each acting on at most 2 qubits, and taken from some finite gate set of size C, is only of order $C^{\binom{k}{2}}$ (at each step, choose a gate and the two qubits for it to act on). This means that to reach any n-qubit unitary, even up to constant accuracy ε, will require circuits of exponential size, $k = 2^{\Omega(n)}$.

But all this is a necessary condition. We care about sufficiency. Is it the case that any n-qubit unitary be approximated to within ε by a finite circuit made of 1 and 2-qubit gates? The answer is yes, but we won't show it explicitly; you can find a proof in the book [KSV02]. It is not difficult to show; the idea is just to decompose any big unitary into a product of simpler operations — in a similar flavor as one finds decompositions such as LU by Gaussian elimination.[1] In fact, it is known that all single-qubit gates, plus CNOT, are sufficient, so these gates form what is called a *universal gate set*.

Remark 2. How is an abstract circuit implemented physically? One way to think about this is that the device has a giant quantum memory, that holds the n qubits in a coherent state. Then whenever a gate U is to be applied to a pair of qubits, the two qubits are copied outside of the box and into a special "chamber" in charge of applying the gate. Then in that chamber one can "turn on" a Hamiltonian H, such as a specific magnetic field, for a fixed amount of time t, such that $e^{iHt} \approx U$. Then the two qubits are copied back into memory, and the process can continue with the next gate. Although this description is probably not entirely accurate, it gives a good idea why it is so hard to construct a quantum computer: it is the interaction between the quantum memory and the device performing the quantum evolution that is particularly delicate.

2.5 The Solovay-Kitaev theorem

What we have seen so far is that there is a well-defined notion of universality: as long as we consider a set of single- and two-qubit gates from which one can construct an arbitrarily good approximation to any single-qubit gate, and that includes the CNOT gate, then the gate set is universal, i.e. it can approximate arbitrarily well any n-qubit unitary (as long as the error per gate is made much smaller than the total number of gates in the circuit, so that errors don't add up to too much).

Unfortunately there is a potential catch in this argument, because the number of gates required still depends on how efficiently the set allows to approximate a single unitary, and this could vary wildly from one set to another. If, for example, getting ε-close required $2^{1/\varepsilon}$ elementary gates, given that we need ε to scale with the size of the final circuit this wouldn't be at all efficient! The Solovay-Kitaev theorem shows that can never be the case: any finite set of gates that generates a dense subset of $U(2)$, the group of all 2×2 unitary matrices, does so efficiently. More precisely:

Theorem 3 (Solovay-Kitaev '97)**.** *There is a constant c such that for any gate set G such that $\langle G \rangle$ is dense in $SU(2)$ and G is closed under inverse, for any $\varepsilon > 0$ there is an $\ell = O(\log^c(1/\varepsilon))$ such that G^ℓ is an ε-net in $SU(2)$.*

The best exponent currently known to hold for all gate sets is $c = 3.97$. For specific gate sets it is known how to get the exponent down to 1. (Information-theoretically, any inverse-closed dense set gets exponent 1.) This is relevant for quantum compilers, that aim to decompose an arbitrary unitary in as small a sequence of gates as possible.

Note the requirement that the gate set is closed under inverses. This seems to be required for the proof, but it is an open question if the theorem still holds without that condition. (See [BO17] for recent progress.)

Note also the theorem makes a statement about $SU(2)$ only. $SU(2)$ is the group of unitaries with determinant 1; equivalently, it is $U(2)$ modulo phase. There is a good reason for this, as the theorem is false for $U(2)$. The problem is that a global phase cannot be approximated efficiently enough. Try it: the best you can do is use continued fractions, and that may require a number of gates that scales like $1/\sqrt{\varepsilon}$, not $\log(1/\varepsilon)$. But this is ok, because quantum operations are only defined up to a global phase.

[1] Beware though that it is in fact really quite different: tensor products of matrices behave very differently from direct sums, so one has to be very careful in trying to apply any intuition one might have about the latter to the former.

While the theorem states existence, we really care about an efficient algorithms. This also exists, and comes out from the proof. See the book [KSV02], or [DN05] for a self-contained proof, or [Ozo09] for a more modern exposition.

2.6 Universal gate sets

A gate set is universal if it is closed under inverse and generates a dense subset of $SU(2)$. This allows us to apply Theorem 3, and adding in the CNOT gate gives a universal set. What gates one chooses depends on the "hardware", or on what operations one finds convenient to have as elementary building blocks, depending on the application. A popular set is the Hadamard and the $\pi/8$ gate (also called T gate), given by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

It is possible to have a universal set made of just a single gate: Deutsch's 3-qubit CTL-CTL-R gate, where R is a rotation along a particular angle θ such that θ/π is irrational.

A useful classification of unitary operations in terms of their power for computation is as follows. At the bottom level we have the Pauli operations, which are all tensor products of single-qubit unitaries taken from $\{\mathrm{Id}, \sigma_X, \sigma_Y, \sigma_Z\}$. These form a group called the Pauli group $\mathcal{P}$. At the first level we have what are called *Clifford gates*: these are all unitaries U such that $UPU^\dagger \in \mathcal{P}$ for all $P \in \mathcal{P}$. This is again a group, called the Clifford group $\mathcal{C}$. Then at the second level we have unitaries that conjugate Cliffords to Cliffords, etc.

It is known that any circuit made only of Clifford gates can be efficiently simulated classically, so it is not universal. This is known as the *Gottesman-Knill theorem*. It is an interesting result because the reason the circuits can be simulated efficiently is *not* because they do not create entanglement; indeed, a Hadamard and a CNOT, which are both Clifford gates, can generate very complex kinds of entangled states, such as any number of EPR pairs between any pairs of circuits. Instead, the Gottesman-Knill theorem relies on representing quantum states using the *Stabilizer formalism*, a topic we will return to when we discuss quantum error correction.

As soon as one non-Clifford gate is added to the 2-qubit Clifford group, we get a universal gate set.

Exercise 4. Verify that the H and CNOT gates are Clifford gates, but the T gate is not.

2.7 Principle of deferred measurement

We have not discussed measurements yet. Our convention has been that a circuit is made of a sequence of single-qubit and two-qubit *unitary* gates. The circuit acts on $n+m$ qubits, the first n of which contain the input $|x\rangle$ and the remaining m are initialized to $|0\rangle$. The output of the circuit is determined by measuring one (or more) specially designated *output qubits* in the computational basis.

What if we would like to make intermediate measurements? Or introduce ancilla qubits in the middle of the computation? Neither of these operations can be modeled as a unitary gate. Rather, both measurements and introduction of ancilla can be represented as *trace-preserving completely positive* (TPCP) maps. The exact definition of a TPCP map would take us too far; suffice it to say that it captures the most general operation allowed by quantum mechanics — it allows to express unitary evolution, measurements, and creation of qubits in the same formalism.

To keep the discussion simple let's consider the case of a measurement in the computational basis: if $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, this returns 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$. Here is an alternate

description of the same operation: initialize an ancilla to $|0\rangle$, apply a CNOT from the computational qubit to the ancilla, and measure the ancilla. The CNOT accomplishes

$$(\alpha|0\rangle + \beta|1\rangle)|0\rangle \;\mapsto\; \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle \;.$$

So measuring the ancilla gives the same outcome distribution as measuring the computational qubit. Moreover, the computational qubit is automatically projected in the same state as it would have had it been measured. Since no operation is performed on the ancilla, it can be measured at the end of the circuit, to the same effect. And the computation can be continued on the computational qubit, right after the CNOT.

This argument shows that any intermediate measurement can be simulated by adjoining a single ancilla qubit, and replacing the measurement gate by a CNOT from the computational qubit to the ancilla. Nothing needs to be done to the ancilla! It can be measured at the end of the circuit, or not, since we don't do anything with the outcome.

Thus, at the expense of increasing the space complexity of the circuit we can always assume a circuit is unitary. But if we consider space-bounded computation?

Open Question 5. Characterize the power of log-space quantum computation. If intermediate measurements are *not* allowed then [FL16] gives complete problems for "unitary quantum logspace". Currently we don't know any problems that can be solved in quantum logspace with intermediate measurements, and not without. Note that Watrous [Wat03] shows that quantum $\log(n)$-space (including intermediate measurements) is included in classical randomized $\log^2(n)$-space.

3 The class BQP

Given a quantum circuit Q acting on $n+m$ qubits, we say that "Q accepts input x with probability p" if the probability of obtaining the outcome 1 after a measurement in the computational basis of the output qubit of Q on input $|x\rangle$ is p.

Definition 6. We say that a language $L = (L_{yes}, L_{no})$ is in BQP if there exists a family of polynomial-time generated quantum circuits $\{Q_n\}_{n\in\mathbb{N}}$ such that for all integer n and $x \in \{0,1\}^n$, if $x \in L_{yes}$ then Q_n accepts x with probability at least $\frac{2}{3}$, and if $x \in L_{no}$ then Q_n accepts x with probability at most $\frac{1}{3}$.

Note the requirement that the family $\{Q_n\}$ is polynomial-time generated. This means that there exists a classical Turing Machine that on input 1^n runs in time $\mathrm{poly}(n)$ and returns a description of Q_n as a sequence of gates taken from a fixed universal set.

The definition of BQP sets arbitrary values $2/3$ and $1/3$ for the completeness and soundness parameters. Error amplification works just as for the case of BPP, by repeating the circuit sequentially. This requires intermediate measurements, but we have seen that these could be postponed till the end of the computation. This requires ancilla qubits, so the state space increases. In fact, it is also known how to perform "in place" amplification, without increasing the number of ancilla. As a result, any choice of a, b such that $a - b > \mathrm{poly}^{-1}(n)$ gives the same definition: for any such a, b, and for any fixed polynomial q, $\mathrm{BQP}(a,b) = \mathrm{BQP} = \mathrm{BQP}(1-2^{-q}, 2^{-q})$.

Exercise 7. Show that BQP is included in PP, the class of languages for which there exists a probabilistic Turing machine that accepts YES inputs with probability $> 1/2$, and rejects NO inputs with probability $> 1/2$. (Hint: first show inclusion in PSPACE by giving space-efficient implementations of basic linear algebra operations. Inclusion in PP follows from similar arguments, but is a bit more delicate.)

The class PP lies outside of the polynomial hierarchy. The most commonly-held belief is that the intersection of BQP and PH is non-trivial: it is neither BPP, nor PH itself. There are candidate oracle problems that are in BQP, but are not believed to be in PH: see [Aar10, FU15].

References

[Aar10] Scott Aaronson. Bqp and the polynomial hierarchy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 141–150. ACM, 2010.

[BO17] Adam Bouland and Maris Ozols. Trading inverses for an irrep in the solovay-kitaev theorem. *arXiv preprint arXiv:1712.09798*, 2017.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[Deu85] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. In *Proc. R. Soc. Lond. A*, volume 400, pages 97–117. The Royal Society, 1985.

[Deu89] David Deutsch. Quantum computational networks. In *Proc. R. Soc. Lond. A*, volume 425, pages 73–90. The Royal Society, 1989.

[DN05] Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030*, 2005.

[FL16] Bill Fefferman and Cedric Yen-Yu Lin. A complete characterization of unitary quantum space. *arXiv preprint arXiv:1604.01384*, 2016.

[FU15] Bill Fefferman and Chris Umans. The power of quantum fourier sampling. *arXiv preprint arXiv:1507.05592*, 2015.

[KSV02] Alexei Yu Kitaev, Alexander Shen, and Mikhail N Vyalyi. *Classical and quantum computation*. Number 47. American Mathematical Soc., 2002.

[Ozo09] Maris Ozols. The solovay-kitaev theorem. *Essay at University of Waterloo*, 2009.

[Wat03] John Watrous. On the complexity of simulating space-bounded quantum computations. *Computational Complexity*, 12(1-2):48–84, 2003.

[Wat09] John Watrous. Quantum computational complexity. In *Encyclopedia of complexity and systems science*, pages 7174–7201. Springer, 2009.

[Yao93] A Chi-Chih Yao. Quantum circuit complexity. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 352–361. IEEE, 1993.