6.896 Quantum Complexity Theory September 16, 2008

Lecture 4

Lecturer: Scott Aaronson

1 Review of the last lecture

1.1 BQP

BQP is a class of languages $L \subseteq (0,1)^*$, decidable with bounded error probability (say 1/3) by a uniform family of polynomial-size quantum circuit over some universal family of gate. In today's lecture, we will see where this BQP sits in inclusion diagram of complexity classes.

1.2 Solovay-Kitaev Theorem

With a finite set of gates, we can approximate any n-qubit unitary within L2 accuracy ϵ using $2^n(n + polylog(1/\epsilon))$ gates (For example, Hadamard and Toffoli gates). In fact, with CNOT-gate and arbitrary 1-qubit gates, we can apply any n-qubit unitary exactly.

2 Basic properties of BQP

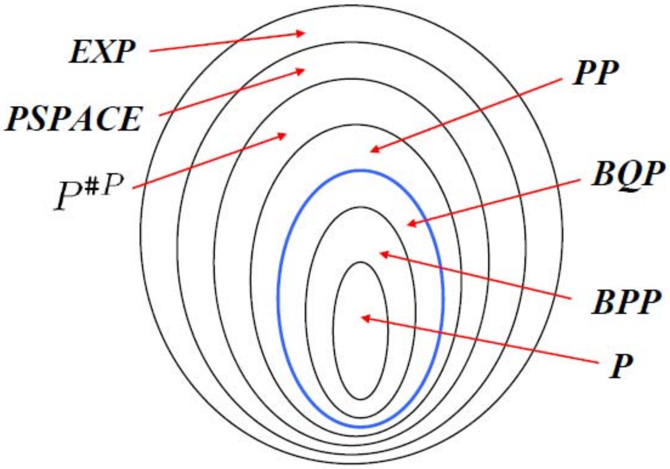

Figure 1: Inclusion diagram of complexity classes.

2.1 $P \subseteq BQP$

We can easily see that quantum circuit can simulate classical circuit.

2.2 $BPP \subseteq BQP$

Quantum computer can solve anything classical probabilistic computer can solve, since quantum property gives us randomness. For example, applying Hadamard gate to $|0\rangle$ gives us a random source of $|0\rangle$ and $|1\rangle$.

2.3 $BPP \subseteq EXP$

Since quantum state is written as $|\psi\rangle = \sum \alpha_x |x\rangle$, we can simulate the whole evolution of all the state vectors with classical computer, within exponential time at most.

2.4 $BQP \subseteq PSPACE$

In terms of computational complexity, the schrodinger picture ($\sum \alpha_x |x\rangle$) and Heisenberg's density matrix (ρ) both lead to an exponential-space simulation since we need to calculate whole evolution of state vectors. On the other hand, the Feynman's path integral, summing up all the histories, leads to a polynomial-space simulation. By writing each final amplitude as a sum of contributions from all the possible paths, we can calculate the sum in $PSPACE$.

For example, the calculation of $H \otimes H|0\rangle$ can be viewed as follows in Feynmann's path integral. We calculate amplitude for each path separately which needs polynomial space only.

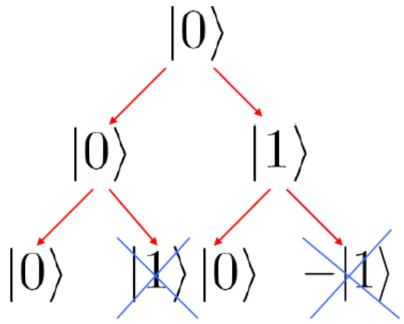

Figure 2: Path integral interpretation of $H \otimes H|0\rangle$. We calculate amplitudes for each four path.

2.5 $BQP \subseteq P^{\#P} \subseteq PSPACE$

$\#P$ is the class of counting problems. To get a class of decision problem, we consider P with $\#P$ oracle, or $P^{\#P}$. Since we can do counting in polynomial space, $P^P \subseteq PSPACE$. Also, $\#P$ can follow all the possible paths non-deterministically in Feynmann's path integral. We can determine that $BQP \subseteq P^{\#P}$.

2.6 $BQP \subseteq PP$

PP stands for probabilistic polynomial time. It is defined as the class of languages L for which there exists a polynomial-time randomized Turing machine M such that for all inputs x:

- if $x \in L$, then $M(x)$ accepts w.p $\geq 1/2$
- if $x \notin L$, then $M(x)$ accepts w.p $< 1/2$.

Note that there is no probability gap, so $1/2$ appears instead of $1/3$ and $2/3$. This class is physical not realistic for we cannot know whether the probability is $1/2$ or $1/2 - 1/2^{|x|}$ without running algorithm exponential time. However, in terms of complexity theory, we can prove that $BQP \subseteq PP$.

PP is the decision version of $\#P$, which means we cannot count the number of accepting paths in the nondeterministic Turing machine, but we can ask whether the number of accepting paths is greater than or less than the number of rejecting paths.

For PP, the threshold is 1/2, but for BQP, the threshold is 1/3. However, we can set the threshold which is less than 1/2 as we like for PP.

At first, we nondeterministially guess x, i, j. Then if $\alpha_{x,i}\alpha^*_{x,j} > 0$, create a number of accepting paths proportional to $|\alpha_{x,i}\alpha^*_{x,j}|$. If $\alpha_{x,i}\alpha^*_{x,j} < 0$, create a number of accepting paths proportional to $|\alpha_{x,i}\alpha^*_{x,j}|$. If $\alpha_{x,i}\alpha^*_{x,j} = 0$, we have the accepting and rejecting paths perfectly balanced each other. Therefore, we know that $BQP \subseteq PP$.

Note that once we get the ability to set the threshold as any number we like, we can determine the exact number by binary search. This fact implies that $P^{\#P} = P^{PP}$.

3 Inclusion diagram

3.1 $BPP \neq BQP$

Can we prove that quantum computer exceeds classical computer? The answer is no since it would imply $P \neq PSPACE$, which is a great challenge as proving $P \neq NP$.

3.2 Where is NP?

At first, we still don't know where NP sits in the diagram and how NP relates to BQP. We conjecture that $NP \not\subseteq BQP$, which means that quantum computer cannot solve NP complete problems in polynomial time. However, we have no idea as for whether $BQP \not\subseteq NP$ or not.

Another interesting question is that if $P = NP$, then $P = BQP$. Also, if $P = PP$, then $P = BQP$.

4 Structural properties of BQP

4.1 $BQP^{BQP} = BQP$?

In classical computer science, we assume that when we write some algorithm, then we can use it as a subroutine in other algorithm. Does the same thing exist in quantum computer?

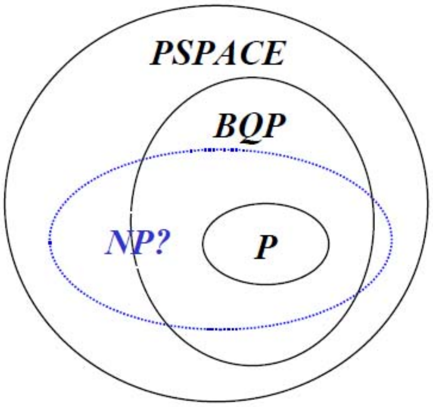

Figure 3: NP and BQP.

For initial input $|0\rangle$, we get $|work(0)\rangle|output(0)\rangle$. For initial input $|1\rangle$, we get $|work(1)\rangle|output(1)\rangle$. Here, $|work(i)\rangle$ represents subroutine and $|output(i)\rangle$ represent the answer to measure. Usually, we throw away unnecessary qubits other than outputs when we proceed to further calculations.

However, if our input state is $|0\rangle+|1\rangle$, then we will have $|work(0)\rangle|output(0)\rangle+|work(1)\rangle|output(1)\rangle$. This state is an entangled state over work space and output space. Naturally, the states in subroutine space(or work space) affect the result of further operation on output space.

4.2 Uncomputing

This smart trick was introduced by Charlie Bennett. At first, we run the subroutine (unitary operation) and get the answer. Then we apply CNOT-gate to the answer and keep it in some safe location that won't be touched again. Then we run the entire subroutine backwards to erase everything but the answer.

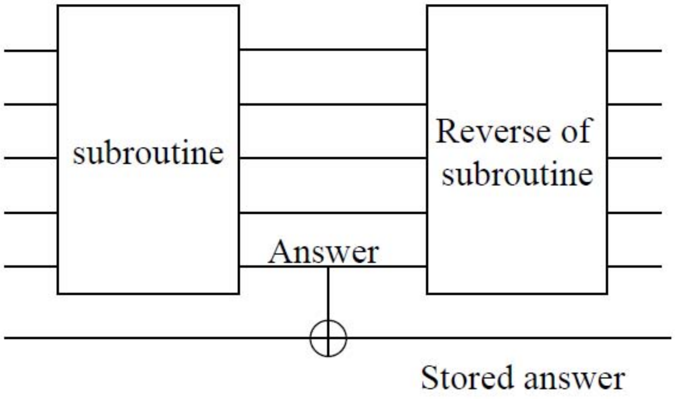

Figure 4: The idea of uncomputing.

The uncomputing step will partly erase the unnecessary residues from the subroutine space, but not completely erase them. However, we can deal with it by amplifying the subroutine part so that error becomes exponentially small. For example, by taking majority vote after many parallel subroutines, we can decrease the error exponentially. The process of taking majority vote can be done in polynomial time, so this amplification process can cope with the error from this uncomputation.

In summary, we know that $BQP^{BQP} = BQP$, in other words, BQP with a BQP oracle is no more powerful than ordinary BQP.