1 Readings

Benenti, Casati, and Strini:

Quantum Gates Ch. 3.2-3.4

Universality Ch. 3.5-3.6

Kaye:

Ch. 1.2, 9.4

Complexity Ch. 9

2 Universality of Gate Sets

2.1 Classical

The *NAND* gate is universal for classical computation. The *NAND* gate is the result of applying *NOT* to $aANDb = a \wedge b = a \uparrow b$. See Figure 7.

For any boolean function $\{0,1\}^n \longrightarrow \{0,1\}$, there is a circuit built of *NAND* gates (possibly with *FANOUT*= copy) for that function. Note that neither of these gates are reversible.

2.2 Quantum

A set G of quantum gates is called universal if for any $\varepsilon > 0$ and any unitary matrix U on n qubits, there is a sequence of gates $g_1, \ldots, g_l$ from G such that $\|U - U_{g_l} \cdots U_{g_2} U_{g_1}\| \leq \varepsilon$.

Here U_g is $V \otimes I$, where V is the unitary transformation on k qubits operated on by the quantum gate g, and I is the identity acting on the remaining $n-k$ qubits. The operator norm is defined by $\|U - U'\| = \max_{|v\rangle \text{unit vector}} \|(U-U')|v\rangle\|$. (Recall that for a vector w, $\|w\| = \sqrt{\langle w|w\rangle}$.)

Examples of universal gate sets include

- *CNOT* and all single qubit gates (continuous gates)
- *CNOT*, Hadamard, and suitable phase flips (continuous gates)
- *CNOT*, Hadamard, X and T ($\pi/8$) (discrete gates)
- Toffoli and Hadamard (discrete gates)

3 Approximating Unitary Operators

Last time we defined a universal set of quantum gates. Now we consider the question of just how many gates are needed to effect an arbitrary quantum operation, or circuit?

An n-qubit gate U (a $2^n \times 2^n$ unitary matrix) has exponentially many parameters. So typically in general we need $\exp(n)$ many gates to even approximate U.

The Solovay-Kitaev theorem says that, as a function of ε, the complexity of an approximation is only $\log^2 \frac{1}{\varepsilon}$. This is rather efficient – the complexity as a function of n is the problem.

In general, the circuit may require an exponential number 2^n of gates. Functions which can be efficiently evaluated require only a polynomial number n^c gates. Quantum computation may be regarded as the study of those unitary transformations on n qubits that can be described by a sequence of polynomial in n quantum gates from a universal family of gates. U is "easy" (implementable) if $U \approx U_{g_k} \cdots U_{g_1}$ for $k = O(\text{poly}(n))$. This definition doesn't depend on our choice of a (finite) universal gate family, since any particular gate in one gate family can be well-approximated with a constant number of gates from another universal gate family. The constant factor does not affect the distinction between polynomial- and exponential-size circuits.

4 Complexity Theory

Complexity theory deals with the scaling of a computation with the resources, in particular, with the number of bits (qubits) specifying the input (the size of the input).

An algorithm is defined as a set of instructions for solving a given problem, e.g., $a + b =?$.

Read the section on Turing machines in Benenti et al., Ch. 1.1. The Church-Turing thesis states that any function computable by an algorithm can be computed on class of a Turing machine, the paradigm of a computational device. Furthermore, there exists a universal Turing machine that given a descriptor of a given computation from a specific Turing machine, can perform that computation with at most polynomial slowdown. This is related to the strong version of the Church-Turing thesis, which states that any model of computation can be simulated by a probabilistic Turing machine with at most polynomial increase in the number of gates. Since gates are equivalent to time, the converse of this implies that if a computational problem cannot be solved in polynomial time on a probabilistic Turing machine, then it is not solvable. Shor's algorithm challenges this since there is a polynomial quantum algorithm, but no known polynomial time classical algorithm. Whether one may exist or not is still an open problem.

Note that the Kitaev-Solovay theorem tells us that provided the gate set is universal, the distinction between functions which require exponentially large circuits and those which can be computed with polynomial-size circuits does not depend on the chosen set of gates.

Now consider how computations scale with number of bits. E.g., finding the square of an integer, x. We need $L = log_2 x$ bits to represent the number and require $s \propto L^2$ steps to compute x^2. Thus finding the square of a numbers is in P, i.e., $s \propto L^k$ where k is an integer. We say that problems with polynomial complexity are 'easy', while problems with superpolynomial complexity are 'hard'. These include scalings $s \propto exp(L), s \propto 2^n, s \propto n!$. Why this distinction? Several reasons:

- usually if one has an algorithm scaling algorithmically one has taken advantage of some mathematical insight into the problem - polynomial algorithms are generally low order ($k = 1, 2, 3$) and one rarely finds $k > 10$
- suppose that today you solve a problem of size n and are asked tomorrow to solve the same problem for a number of size $n+1$. If your algorithm scales as $O(2^n)$ you will need twice as much computation time tomorrow, but if your algorithm scales as $O(n^2)$ then you will need only a small fraction more than today (consider $2n+1$ versus n^2).

Examples of complexity classifications:

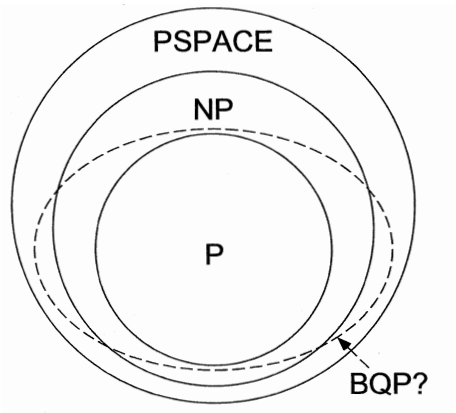

Figure 1: Schematic of possible hierarchy of complexity classes.

- matrix mutliply $O(n^3)$ (now lower order but greater than 2 is possible)
- sorting n items $O(nlogn)$
- factorization of an integer N by number field sieve $expO(n^{1/3}(logn)^{2/3})$ (here $n = logN$ is the input size). For a number N with 250 digits, this requires about 10^6 years on a 200 MIPS machine.

Complexity classes

P = solve algorithm in time (gates) polynomial in the number of bits

NP = can verify a solution in polynomial time, e.g., whether numbers a and b factor N (just multiply them together)

$P \subset NP$ but it is an open questions as to whether NP is actually larger than P...

NPC = any NP problem can be reduced to it ($NP - complete$), e.g., the traveling salesman problem.

$PSPACE$ = polynomial in space resources but no limit on time

$P \subset PSPACE$ but it is also an open question whether $PSPACE$ is actually bigger than P...

BPP = bounded probabilistic polynomial algorithm

BQP = quantum probabilistic algorithm with bounded error and running in polynomial time

What is known is that

$P \subseteq BPP \subseteq BQP \subseteq PSPACE$. The extent of BQP is not well understood.

For an introductory overview to complexity theory, see S. Mertens, cond-mat/0012185.