Quantum information theory (20110401)

Lecturer: Jens Eisert
Chapter 8: Elements of quantum computing

Contents

8 Elements of quantum computing **5**
8.1 Why quantum computing? 5
8.1.1 Quantum computers reduce the complexity of certain computational tasks 5
8.1.2 Quantum systems can simulate other quantum systems 6
8.1.3 Moore's law has physical limits 6
8.1.4 Even small quantum circuits may be useful 7
8.2 From classical to quantum computing 7
8.2.1 Qubits and quantum parallelism 7
8.2.2 Read out and probabilistic nature of quantum computers . . . 8
8.2.3 The circuit model and universal quantum gates 8
8.3 Gottesman-Knill and Solovay-Kitaev theorems 11
8.3.1 Clifford operations 11
8.3.2 The Solovay-Kitaev-theorem 12
8.3.3 How to program a quantum computer? 12

Chapter 8

Elements of quantum computing

8.1 Why quantum computing?

8.1.1 Quantum computers reduce the complexity of certain computational tasks

One reason for aiming at building quantum computers is that they will solve certain types of problems faster than any (present or future) classical computer – it seems that the border between *easy* and *hard* problems is different for quantum computers than it is for their classical counterparts. Here easy means that the time for solving the problem grows polynomially with the length of the input data (like for the problem of multiplying two numbers), whereas hard problems are those for which the required time grows exponentially. Prominent examples for hard problems are the travelling salesman problem, the graph isomorphism problem, and the problem of factoring a number into primes[1]. For the latter it was, to the surprise of all, shown by Peter Shor in 1994 that it could efficiently be solved by a quantum computer in polynomial time. Hence, a problem which is hard for any classical computer becomes easy for quantum computers[2]. Shor's result gets even more brisance from the fact that the security of public key encryption, i.e., the security of home banking and any other information transfer via the internet, is heavily based on the fact that factoring is a hard problem.

One might think that the cost for the gained exponential speedup in quantum computers is an exponential increase of the required accuracy for all the involved operations. This would then be reminiscent of the drawback of analogue computers. Fortunately, this is not the case and a constant accuracy is sufficient. However, achieving this "constant" is without doubt experimentally highly challenging[3]

---

[1] These problems are strongly believed to be hard (the same is by the way true for a special instance of the computer game "minesweeper"). However, in all cases there is no proof that a polynomial-time algorithm can not exist. The question whether there exists such an algorithm (for the travelling salesman or the minesweeper problem) is in fact the notorious $P \overset{?}{=} NP$ question for whose solution there is even a prize of one million dollar.

[2] In fact Shor's algorithm strikes the *strong Church-Turing thesis*, which states that every reasonable physical computing device can be simulated on a probabilistic Turing machine with at most polynomial overhead.

8.1.2 Quantum systems can simulate other quantum systems

Nature provides many fascinating collective quantum phenomena like superconductivity, magnetism and Bose-Einstein condensation. Although all properties of matter are described by and can in principle be determined from the laws of quantum mechanics, physicists have very often serious difficulties to understand them in detail and to predict them by starting from fundamental rules and first principles. One reason for these difficulties is the fact that the number of parameters needed to describe a many-particle quantum system grows exponentially with the number of particles. Hence, comparing a theoretical model for the behavior of more than, say, thirty particles with experimental reality is not possible by simulating the theoretical model numerically on a classical computer without making serious simplifications. When thinking about this problem of simulating quantum systems on classical computers Richard Feynman came in the early eighties to the conclusion that such a classical simulation typically suffers from an exponential slowdown, whereas another quantum system could in principle do the simulation efficiently with bearable overhead

In this way, a quantum computer operated as a *quantum simulator* could be used as a link between theoretical models which are formulated on a fundamental level and experimental observations. Similar to Shor's algorithm a quantum simulator would yield an exponential speedup compared to a classical computer. An important difference between these two applications is, however, that a useful Shor-algorithm quantum computer requires thousands of qubits whereas a few tens of qubits could already be useful for the simulation of quantum systems.

8.1.3 Moore's law has physical limits

Apart from the computational power of a quantum computer there is a much more banal argument for incorporating quantum mechanics into computer science: *Moore's law*. In 1965 Intel co-founder Gordon Moore observed an exponential growth in the number of transistors per square inch on integrated circuit and he predicted that this trend would continue. In fact, since then this density has doubled approximately every 18 months[4]. If this trend continues then around the year 2020 the components of computers are at the atomic scale where quantum effects are dominant. We have thus to inevitably cope with these effects, and we can either try to circumvent and eliminate them as long as this is possible and keep on doing classical computing or we can at some point try to make use of them and start doing quantum computing.

---

[3] Note that the beginning of this chapter is largely taken from a review article of mine on quantum computing.

[4] Actually, not every prediction of the pioneers in computer business was that foresighted: In 1943 Thomas Watson, chairman of IBM, for instance predicted a world market for five computers and in 1977 Digital Equipment Corp. founder Ken Olson stated that "there is no reason anyone would want a computer in their home".

8.1.4 Even small quantum circuits may be useful

Besides the quantum computer with its mentioned applications quantum information science yields a couple of other useful applications which might be easier to realize. The best example is quantum cryptography which enables one to transmit information with “the security of nature’s laws”. However, small building blocks of a quantum computer, i.e., small quantum circuits may be useful as well. One potential application is for instance in precision measurements like in atomic clocks. The latter are important in global positioning systems as well as in synchronizing networks and distant telescopes. By generating quantum correlations between the N relevant atoms in the atomic clock, a quantum circuit could in principle reduce the uncertainty of the clock by a factor $\sqrt{N}$. Another application of small quantum circuits is *entanglement distillation*: in order to distribute entangled states over large distances we have to send them through inevitably noisy channels, thereby loosing some of the entanglement. Fortunately, however, we can in many cases *distill* a few highly entangled states out of many weakly entangled ones.

8.2 From classical to quantum computing

Let us now have a closer look at the way a quantum computer works. We will do so by comparing the concepts of classical computing with the basics of quantum computing. In fact, many classical concepts have very similar quantum counterparts, like bits become qubits and still the logic is often best explained within a circuit model. However, there are also crucial differences, which we will describe on the following pages.

8.2.1 Qubits and quantum parallelism

We have already see that the elementary information carriers in a quantum computer are the *qubits* – quantum bits. In contrast to classical bits which take on either the value zero or one, qubits can be in every *superposition* of the state vectors $|0\rangle$ and $|1\rangle$. This means that the vector $|\psi\rangle$ describing the state vector of the qubit can be any linear combination

$$|\psi\rangle \;=\; \alpha|0\rangle + \beta|1\rangle \tag{8.1}$$

of the vectors $|0\rangle$ and $|1\rangle$ with complex coefficients α and β. In the same way a system of many qubits can be in a superposition of *all* classically possible states

$$|0,0,\ldots,0\rangle + |1,0,\ldots,0\rangle + \ldots + |1,1,\ldots,1\rangle \;. \tag{8.2}$$

The basis $\{|0,0,\ldots,0\rangle, |0,1,\ldots,0\rangle, \ldots, |1,1,\ldots,1\rangle\}$ that corresponds to the binary words of length n in a quantum system of n qubits is called the *computational basis*[5].

Using the superposition of Eq. (8.2) as an input for an algorithm means somehow to run the computation on all classically possible input states at the same time. This possibility is called *quantum parallelism* and it is certainly one of the reasons for the computational power of a quantum computer. The mathematical structure behind the composition of quantum systems is the one of the tensor product. This implies that the dimension of the space characterizing the system grows exponentially with the number of qubits. A Physically, qubits correspond to effective two-level systems like ground state and excited state of an atom, the polarization degree of freedom of light or up-and down orientation of a spin-1/2 particle.

---

[5] In finite-dimensional quantum systems as we encounter here the computational basis spans the *Hilbert space* associated with the physical system.

8.2.2 Read out and probabilistic nature of quantum computers

An important difference between classical and quantum computers lies in the read-out process. In the classical case there is not much to say: the output is a bit-string which is obtained in a deterministic manner, i.e., repeating the computation will lead to the same output again[6]. However, due to the probabilistic nature of quantum mechanics, this is different for a quantum computer. If the output of the computation is for instance the state vector $|\psi\rangle$ in Eq. (8.1), α and β cannot be determined by a single measurement on a single specimen. In fact, $|\alpha|^2$ and $|\beta|^2$ are the probabilities for the system to be found in $|0\rangle$ and $|1\rangle$ respectively. Hence, the absolute values of these coefficients can be determined by repeating the computation, measuring in the basis $|0\rangle,\ |1\rangle$ and then counting the relative frequencies. The actual outcome of every single measurement is thereby completely indetermined. In the same manner, the state of a quantum system consisting of n qubits can be measured in the computational basis, which means that the outcome corresponding to some binary word occurs with the probability given by the square of the absolut value of the respective coefficient. So in effect, the probabilistic nature of the read out process on the one hand and the possibility of exploiting quantum parallelism on the other hand are competing aspects when it comes to comparing the computational power of quantum and classical computers.

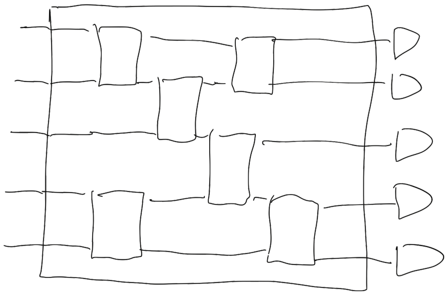
8.2.3 The circuit model and universal quantum gates

A classical digital computer operates on a string of input bits and returns a string of output bits. The function in between can be described as a logical circuit build up out of many elementary logic operations. That is, the whole computation can be decomposed into an array of smaller operations – gates – acting only on one or two bits like the AND, OR and NOT operation. In fact, these three gates together with the COPY (or FANOUT) operation form a *universal* set of gates into which every well-defined input-output function can be decomposed. The complexity of an algorithm is then essentially the number of required elementary gates, resp. its asymptotic growth with the size of the input.

The circuit model for the quantum computer is actually very reminiscent of the classical circuit model: of course, we have to replace the input-output function by a quantum operation mapping quantum states onto quantum states. It is sufficient to consider operations only that have the property to be unitary, which means that the computation is taken to be logically reversible. In turn, any unitary operation can be decomposed into elementary gates acting only on one or two qubits. A set of elementary gates that allows for a realization of any unitary to arbitrary approximation is again referred to as being *universal*.

---

[6] Within the circuit model described above this is a trivial observation since all the elementary gates are deterministic operations. Note that even *probabilistic* classical algorithms run essentially on deterministic grounds.



Universal gate set: A set of unitaries $\mathcal{S}$ is called a universal gate set on n qubits, if for an arbitrary $\epsilon > 0$ there exists a composition of gates $\mathcal{S}$ supported on a finite subset of qubits each so that the circuit V generated in this way satisfies

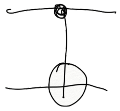
$$\|U - V\| < \epsilon, \tag{8.3}$$

where $\|.\|$ is the operator norm.

The operator norm is nothing but the largest singular value. An important example of a set of universal gates is in this case any randomly chosen one-qubit rotation together with the *CNOT (Controlled NOT)* operation, which acts as

$$|x, y\rangle \mapsto |x, y \oplus x\rangle \ , \tag{8.4}$$

where $\oplus$ means addition modulo 2.



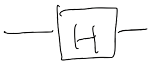
Like in the classical case there are infinitely many sets of universal gates. Notably, also any generic (i.e., randomly chosen) two-qubit gate (together with the possibility of switching the leads in order to swap qubits) is itself a universal set, very much like the NAND gate is for classical computing[7]. Notably, any quantum circuit that makes use of a certain universal set of quantum gates can be simulated by a different quantum circuit based on another universal set of gates with only polylogarithmic overhead.

---

[7] Any such generic quantum gate has so-called *entangling power*, in that it may transform a product state vector into one that can no longer be written as a tensor product. Such quantum mechanical pure states are called *entangled*. In intermediate steps of a quantum algorithm the physical state of the system is in general highly multi-particle entangled. In turn, the implementation of quantum gates in distributed quantum computation requires entanglement as a resource.

A particularly useful single-qubit gate is the *Hadamard gate*, acting as

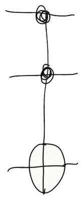
$$|0\rangle \quad \mapsto \quad H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}, \quad |1\rangle \mapsto H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}. \tag{8.5}$$



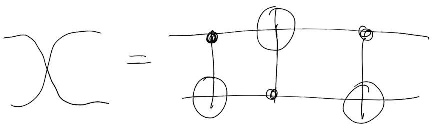
Another quantum gate often considered is the *quantum Toffoli gate*. It is a three qubit analog of the CNOT gate, acting as

$$|a, b, c\rangle \mapsto |a, b, c \oplus ab\rangle \; . \tag{8.6}$$

Remarkable, the Toffoli gate and the Hadamard gate together constitute a universal gate set. That is to say, one can approximate an abitrary single-qubit rotation by means of a suitable circuit consisting of Hadamard and Toffoli gates only, which seems highly counter-intuitive.



The *SWAP* gate exchanges two qubits in their quantum state. Interestingly, it has maximum entangling power, if one things about it. It can be written in terms of three CNOT gates.

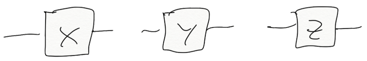


A *phase gate* does nothing but multiplying one of the basis vectors with a phase,

$$|0\rangle \quad \mapsto \quad |0\rangle, \quad |1\rangle \mapsto i|1\rangle, \tag{8.7}$$

and a *Pauli gate* corresponds to one of the three unitary Pauli matrices,

$$|\psi\rangle \mapsto X|\psi\rangle, \tag{8.8}$$
$$|\psi\rangle \mapsto Y|\psi\rangle, \tag{8.9}$$
$$|\psi\rangle \mapsto Z|\psi\rangle. \tag{8.10}$$



8.3 Gottesman-Knill and Solovay-Kitaev theorems

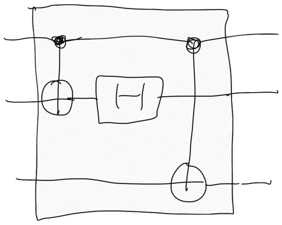
8.3.1 Clifford operations

The CNOT, the Hadamard, the phase gate, and the Pauli gate are quantum gates of utmost importance. Given their key status in many quantum algorithms, one might be tempted to think that with these ingredients alone (together with measurements of Pauli operators, see below), powerful quantum algorithms may be constructed that outperform the best known classical algorithm to a problem. This intuition is yet not correct: it is the content of the *Gottesman-Knill theorem* that any quantum circuit consisting of only these ingredients can be simulated efficiently on a classical computer. The proof of the Gottesman-Knill-theorem is deeply rooted in the stabilizer formalism that we will encounter later in the context of quantum error correction.

Pauli group: The Pauli group $\mathcal{G}_1$ on one qubit is the 16 element group defined by the Pauli operators together with prefactors ± 1 and $\pm i$, i.e.,

$$\mathcal{G}_1 = \{\pm\mathbb{I}, \pm i\mathbb{I}, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}, \tag{8.11}$$

The Pauli group on n qubits $\mathcal{G}_n$ is the group generated by the operators described above applied to each of qubits in $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$.



Closely related to the Pauli group is the normalizer of the Pauli group, the so-called *Clifford group*.

Clifford group: The n-qubit Clifford group $\mathcal{C}_n$ is the normalizer (up to complex phases) of the Pauli group $\mathcal{G}_n$, i.e.,

$$\mathcal{C}_n = \{U : U\mathcal{G}_n U^\dagger \subset \mathcal{G}_n\}/U(1). \tag{8.12}$$

Clifford operations, so the operations associated with Clifford unitaries, are extremely important in quantum information theory. Since they map Pauli operators onto Pauli operators, they can be efficiently described, even though they can be used to generate multipartite entanglement.

Gottesman-Knill theorem: Clifford circuits can be classically efficiently simulated, i.e., in polynomial time.

8.3.2 The Solovay-Kitaev-theorem

An important theorem in this context is the *Solovay-Kitaev-theorem*. It basically states that if a set of single-qubit quantum gates generates a dense subset of $SU(2)$ then that set is guaranteed to fill $SU(2)$ quickly, which means any desired gate can be approximated by a fairly short sequence of gates from the generating set.

Solovay-Kitaev-theorem: Let $\mathcal{G}$ be a finite set of elements in $SU(2)$ containing its own inverses (so that $g \in \mathcal{G}$ implies that $g^{-1} \in \mathcal{G}$ and such that the group $\langle \mathcal{G} \rangle$ is dense in $SU(2)$. Consider some $\epsilon > 0$. Then there is a constant c such that for any $U \in SU(2)$, there is a sequence S of quantum gates from $\mathcal{G}$ of length $O(\log c(1/\epsilon))$ such that $\|S - U\| \leq \epsilon$. That is, S approximates U in operator norm error.

8.3.3 How to program a quantum computer?

The good thing about the classical computer on which this chapter has been written is that it is programmable. It is a single device capable of performing different operations depending on the program it is given: word processing, algebraic transformations, displaying movies, etc.. To put it in more abstract words a classical computer is a *universal gate array*: we can program *every* possible function with n input and n output bits by specifying a program of length $n2^n$. That is, a fixed circuit with $n(1 + 2^n)$ input bits can be used in order to compute any function on the first n bits in the register. Is the same true for quantum computers? Or will these devices typically be made-to-measure with respect to a single task? Nielsen and Chuang have shown that quantum computers cannot be universal gate arrays. Even if the program is itself given in form of a quantum state it would require a program register of infinite length in order to perform an arbitrary (unitary) operation on a finite number of qubits – universality was shown to be only possible in a probabilistic manner. In this sense, quantum computers will not be the kind of all purpose devices which classical computers are. In practice, however, any finite set of quantum programs can run on a quantum computer with a finite program register. This issue applies, however, to the programming of a quantum computer with a fixed hardware, which is, needless to say, still in the remote future as a physical device.