Manuel Kauers
RISC-Linz

Hagenberg, May 7, 2003

Factoring Integers in Polynomial Time
An Introduction to Quantum Computing

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Notation . . . 2

2 Quantum Basics **2**
2.1 Physical Background . . . 2
2.2 Quantum Information . . . 3
2.3 Quantum Gates . . . 4

3 Quantum Algorithms **5**
3.1 Deutsch-Jozsa Algorithm . . . 5
3.2 Quantum Fourier Transform . . . 7
3.3 Factoring Integers . . . 9

4 Conclusion **11**

1 Introduction

1.1 Motivation

Many algorithms in Computer Algebra suffer from a very high complexity. A famous challange problem of high complexity is the factorisation of integers. The best known *classical* algorithm for integer factorisation runs in time $\exp(\mathrm{O}(1)n^{1/3}\log^{2/3} n)$ where n is the length of the integer to be factored, i.e., the number of bits needed to encode it in binary.

However, by deviating from the classical concept of computation, and by allowing *quantum* steps in the algorithm, algorithms can be developed which are a great deal faster than their classical counter parts. Such an algorithm for integer factorisation has been discovered by Peter Shor and the main purpose of this paper is to outline the basic ideas of his algorithm together with a description of the necessary amount of underlying quantum computing concepts.

It is hardly possible to give an introduction to quantum computing within a 90 minutes talk or in a paper of corresponding length — a one semester course would be much more appropriate. As a result, we will have to leave out a lot of details. More rigorous introductions into quantum computing, including also an explanation of Shor's algorithm, can be found in [3, 4, 5, 6]. This paper was based on these sources.

1.2 Notation

Quantum computing has to do with quantum mechanics which in turn has to do with physics. Physicists are known to like strange notation, and the physicists' notation has become standard in the field of quantum computing. We will stick to this notation for the sake of consistency to the literature.

It will turn out later that the vector space $\mathbb{C}^2$ plays an important role in quantum computing. Once and for all, we single out an orthonormal basis $B = \{b_0, b_1\}$ of $\mathbb{C}^2$. Using *ket notation*, we will write $|0\rangle := b_0$ and $|1\rangle := b_1$. More general, we will write $|x\rangle = a|0\rangle + b|1\rangle$ for any vector in $\mathbb{C}^2$, and $\langle x|$ for the transpose of $|x\rangle$. $\langle x|y\rangle := \langle x| \cdot |y\rangle$ consequently denotes the standard skalar product of $|x\rangle$ with $|y\rangle$.

The ket notation is used also in the spaces $V = \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ where $A \otimes B$ denotes the *tensor product* of A with B. If $\{a_1, \ldots, a_n\}$ is a basis of A and $\{b_1, \ldots, b_m\}$ is a basis of B, then $A \otimes B$ can be viewed as the vector space to the basis $\{a_i \otimes b_j : i = 1, \ldots, n; j = 1, \ldots, m\}$ where $a_i \otimes b_j$ is simply a formal product. In ket notation, we will often use notations like

$$|00\ldots0\rangle := |0\rangle|0\rangle\ldots|0\rangle := |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle.$$

for the basis vectors in $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$.

2 Quantum Basics

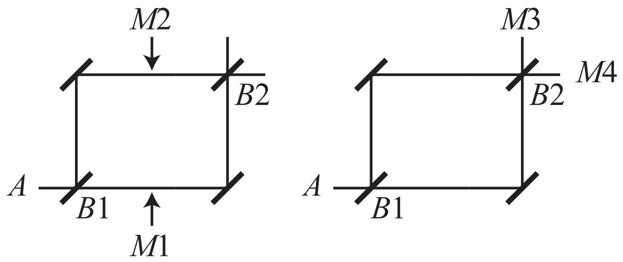
2.1 Physical Background

Wolfang Pauli said that it is an inherent property of quantum mechanics that it cannot be understood; it can only be accepted. Following this "paradigm" we will not try to understand physical underpinnings of what is going on in the still hypothetical device of a quantum computer, especially we will not say anything about how a quantum computer can be implemented in real world. Rather, we want to remind the reader of some spectacular effects in quantum physics which (s)he might have heard about in high school, mainly to develop some kind of intuition.

What is the behaviour of an electron in the hull of an atom? First physical models believed that electrons are flying around the nucleus on well defined orbits just as planets deterministically revolve around their star. Quantum physics tells us that this is an oversimplified model. Rather, the electron is smoothly distributed around the nucleus. Like with photons, it appears in both wave and particle form. When we leave it in peace, it will at the same time be somehow everywhere and somehow nowhere in the hull, but when we make a *measurement,* it "flips" to the particle state and can be located.

If the life of a cat in a black box is made dependent of some effects from quantum mechanics, we will not be able to predict for a certain time if the cat is dead or alive. (Schrödinger's cat.) We only know when we open the box to check, i.e., again when we do a *measurement.* Before the measurement, the cat is actually *both* dead and alive.

To see that the state of the cat is indeed a *superposition* of dead and alive rather than "either dead or alive, we just don't know", consider the Mach-Zehnder interferometer depicted in the following diagram.



Two beam splitters B_1 and B_2 leave half of the light through without disturbing, and half of the light is perfectly reflected. Now, if a photon is emitted from source A it may be found with equal probability $p = \frac{1}{2}$ at one of the positions M_1 and M_2 if a measurement is done there. However, if we measure only after the second beam splitter B_2, we will find the electron with probability $p = 1$ in M_3 and with probability $p = 0$ in M_4. It is therefore better to imagine that the photon takes both ways the same time. It does not decide at B_1 which way to take, but rather it decides at M_1 and M_2 which way it has taken.

2.2 Quantum Information

The smallest amount of classical data is the bit. Such a classical bit B always has exactly one value, either $B = 0$ or $B = 1$. In a string $(B_1, \ldots, B_n)$ of n bits each bit takes either the value 0 or the value 1. The value of the entire string can be viewed as an element of

$$\{0,1\}^n = \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{n \text{ times}},$$

hence a string of n bits can altogether take 2^n different values.

The main difference between classical information and quantum information is the possibility of *superposition.* Like the electron and the cat and the photon, a *quantum bit* or *qubit* can take two states at the same time, i.e., it can be both 0 and 1. Mathematically, we model a qubit as an element on the unit sphere in $\mathbb{C}^2$, or equivalently as a vector in the projective space of $\mathbb{C}^2$ factoring out the length of the vectors. The classical bit values 0 and 1 will correspond to the designated basis vectors $|0\rangle$ and $|1\rangle$. The general value of a qubit $|q\rangle$ is of the form

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha^2 + \beta^2 = 1$. By slight abuse of notation, we may state the state of Schrödinger's cat as

$$|\text{cat}\rangle = \frac{1}{\sqrt{2}}\Big(|\text{dead}\rangle + |\text{alive}\rangle\Big)$$

for supporting the intuition.

Deep physical reasons suggest that an adequate model for systems of n qubits is the tensor product $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ of n copies of $\mathbb{C}^2$. Its basis

$$\{|00\ldots000\rangle, |00\ldots001\rangle, |00\ldots010\rangle, |00\ldots011\rangle, \ldots |11\ldots110\rangle, |11\ldots111\rangle\}$$

will be refered to as the *computational basis.*

When we do a computation, we are usually also interested in its result. This means that we have to measure the physical state of a system. In the classical setting, there is nothing special about this step, but in the quantum world there is. We have seen in the example of the Mach-Zehnder interferometer that the measurement of a quantum system necessarily influences the system. In our mathematical model, we will regard a measurement as a projection of a vector to the computational basis. A measurement of a state $\alpha|0\rangle + \beta|1\rangle$ will return the value $|0\rangle$ with probability α^2 and the value $|1\rangle$ with probability β^2. (Again, compare this to the example of Schrödinger's cat.)

Note that in a system of more than one qubit, the individual bits might be *entangled.* This means that the measurement of one of the bits also determines the value of the other bits. For instance, when measuring the first qubit of the state $|q\rangle = \alpha|00\rangle + \beta|11\rangle$, the measurement of the second qubit will yield the same value with probability 1. This is clear in the mathematical model since

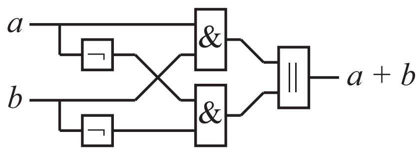
$$\alpha|00\rangle + \beta|11\rangle = \alpha|00\rangle + \beta|11\rangle + 0|01\rangle + 0|10\rangle$$

and hence the states $|01\rangle$ and $|10\rangle$ are observed only with probability 0, however, the physical intuition behind may suggest something else at first glance.

2.3 Quantum Gates

One theoretical model for classical computation is the circuit model. A circuit consists of *wires* and *gates.* An (n, m)-gate is an elementary boolean function that maps a tuple of n input values to a tuple of m output values. Typical gates are the $(2, 1)$-gate AND, or the $(1, 1)$-gate NOT. A circuit with n input wires and m output wires encodes a boolean function $f\colon \{0,1\}^n \to \{0,1\}^m$.

For instance, the following circuit encodes addition in $\mathbb{Z}_2$.



Classically, to every boolean function $f\colon \{0,1\}^n \to \{0,1\}^m$ there exists a circuit implementing it which contains only AND, OR, and NOT gates. We say that $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$ build a *universal set* of operators. The *complexity* of a circuit is the number of gates it contains. The *complexity* of a boolean function $f\colon \{0,1\}^n \to \{0,1\}^m$ is the complexity of the smallest circuit implementing f.

Functions $f\colon \mathbb{N} \to \mathbb{N}$ cannot be described by circuits because the input size for a circuit is fixed. Instead, a *family* of circuits is needed which contains for each $n \in \mathbb{N}$ exactly one circuit with n input wires. A function $f\colon \mathbb{N} \to \mathbb{N}$ is computable if and only if there exists a Turing machine that computes to a given $n \in \mathbb{N}$ in time polynomial in n a circuit with n input wires that outputs $f(i)$ for every $i < 2^n$.

Quantum computers can be modeled by quantum circuits. A quantum circuit is a circuit whose wires carry qubits instead of bits and whose gates preform quantum operations. Quantum gates may act on either one or two wires and they perform *unitary operations* on their input. Note that qubits live in $\mathbb{C}^2$, so a one-qubit gate is described by a 2×2-matrix and two-qubits are described by 4×4-matrices.

An example for a one-qubit gate is the *Hadamard-Transform* defined by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

wrt. the computational basis $\{|0\rangle, |1\rangle\}$. An example for a two-qubit gate is the *controlled not* gate defined by the matrix

$$\mathrm{cNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

wrt. the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Speaking in terms of classical circuits, the cNOT gate flips the second bit if and only if the first bit is 1. The first bit remains unchanged.

The cNOT gate together with all one-qubit gate form a universal set of gates, i.e., all functions $f\colon (\mathbb{C}^2)^n \to (\mathbb{C}^2)^m$ can be implemented in a quantum circuit involving only one-qubit gates and the cNOT gate.

To compute functions with varying input length, a family of quantum circuits is employed — as in the classical case. As there, we require the circuits for input size n to be computed in time polynomial in n by a (classical) Turing machine A. In addition, we have to ensure that no uncomputable (or hard-to-compute) functions are hidden in the one-qubit gates used by machine A. Thus, we additionally require another Turing machine which computes the kth digit of the matrix elements of all quantum gates occuring in the output of A in time polynomial in k.

Functions are said to be of *bounded error probabilistic polynomial quantum complexity* if they can be implemented by a family of quantum circuits whose size is polynomial in the length of the input and whose output after measurement is correct with probability greater than 1/2. By BQP we denote the class of these functions.

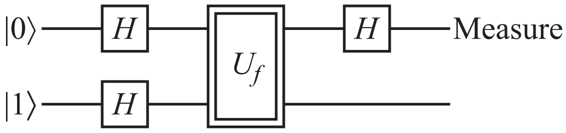
3 Quantum Algorithms

3.1 Deutsch-Jozsa Algorithm

Before we turn tho the description of the factoring algorithm, we present another small algorithm, mainly in order to get accustumed to the notions introduced in the previous sections, and to see the special features of quantum algorithms first at an easier example.

Consider a boolean function $f\colon \{0,1\} \to \{0,1\}$, and assume that we are given a quantum circuit that computes f. We are about to decide if f is balanced, i.e. if $f(0) \neq f(1)$, using as few computations of $f(x)$ as possible. In a classical computer, there is no chance: we have to compute $f(0)$ and $f(1)$ and to compare the results. If we want to solve the problem faster by a quantum computer, we have to answer the query with only one computation of $f(x)$.

Somewhat counterintuitively, this is indeed possible. Given $f\colon \{0,1\} \to \{0,1\}$, consider the operator U_f defined by $U_f|xy\rangle = |x, y \oplus f(x)\rangle$ for $x, y \in \{0,1\}$ and linearly extended to $\mathbb{C}^2 \otimes \mathbb{C}^2$. U_f is a unitary operator and hence an admissible quantum gate. Consider the following quantum circuit



which computes
$$(H \otimes I) \cdot U_f \cdot (H \otimes H) \cdot |01\rangle.$$

The first two Hadamard gates produce superpositions of $|0\rangle$ and $|1\rangle$,
$$(H \otimes H)|01\rangle = (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) = (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$
where we omit the constant factors. Application of U_f to this state leads to
$$|0\rangle\big((|0\rangle - |1\rangle) \oplus |f(0)\rangle\big) + |1\rangle\big((|0\rangle - |1\rangle) \oplus |f(1)\rangle\big) \tag{1}$$

By considering the special cases $f(x) = 0$ and $f(x) = 1$,
$$\begin{aligned}
(|0\rangle - |1\rangle) \oplus |0\rangle &= |0\rangle - |1\rangle, \\
(|0\rangle - |1\rangle) \oplus |1\rangle &= |1\rangle - |0\rangle = (-1)(|0\rangle - |1\rangle),
\end{aligned}$$
we find that we can write
$$(|0\rangle - |1\rangle) \oplus |f(x)\rangle = (-1)^{f(x)}(|0\rangle - |1\rangle).$$

This allows to write (1) in the form
$$\Big((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big)(|0\rangle - |1\rangle).$$

Finally, another Hadamard transform is applied to the first register giving, up to a constant,
$$\begin{aligned}
&H \cdot \Big((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big) \\
&= (-1)^{f(0)}(|0\rangle + |1\rangle) + (-1)^{f(1)}(|0\rangle - |1\rangle) \\
&= \Big((-1)^{f(0)} + (-1)^{f(1)}\Big)|0\rangle + \Big((-1)^{f(0)} - (-1)^{f(1)}\Big)|1\rangle.
\end{aligned}$$

What happens if we measure this state? If $f(0) \neq f(1)$ then the coefficient of $|0\rangle$ vanishes and we obtain 1 with certainty. Likewise, if $f(0) = f(1)$ then the coefficient of $|1\rangle$ vanishes and we obtain 0 with certainty.

The measurement result indicates if f is balanced, and only one computation of f was sufficient to decide this.

The algorithm outlined above is called the Deutsch-Jozsa algorithm. It can be extended to functions $f\colon \{0,1\}^n \to \{0,1\}$ which are called balanced if exactly have of the values are 0 and half of the values are 1. Even here, where a classical computer would need exponential evaluations of f for a decision, only one evaluation suffices in the quantum setting.

3.2 Quantum Fourier Transform

As an important subroutine for the factoring algorithm, we will need to compute a quantum version of the *Discrete Fourier Transform* that will be introduced in this section. Let $q = 2^l$ be a power of two, and fix $\omega = \omega_q = \exp(2\pi \mathrm{i}/q)$ as a qth root of unity for defining the *Quantum Fourier Transform* $\mathrm{QFT} = \mathrm{QFT}_q\colon (\mathbb{C}^2)^l \to (\mathbb{C}^2)^l$ by

$$\mathrm{QFT}\,|a\rangle := \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} \omega^{ab} |b\rangle \qquad (0 \leq a < q).$$

In this definition, as introduced earlier, we use the convention that $|n\rangle$ denotes the tensor

$$|n\rangle = |b_{l-1}\rangle \otimes |b_{l-2}\rangle \otimes \cdots \otimes |b_0\rangle$$

where $b_{l-1}b_{l-2}\ldots b_0$ is the binary representation of n.

Comparing this definition of QFT to the usual DFT of size q, defined in familiar notation by

$$\mathrm{DFT}\,e_i = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \omega^{ij} e_j \qquad (i = 0, \ldots, q-1)$$

where e_i denotes the ith basis vector of an ONB of $\mathbb{C}^q$, we see that the two definitions formally coince (up to notational details).

The algorithm of Cooley-Tucky allows to perform the DFT on a classical computer in time $\mathrm{O}(q \log q)$ using a divide and conquer approach. It is based on the decomposition

$$\mathrm{DFT}_{mn} = (\mathrm{DFT}_m \otimes I_n) \cdot T \cdot (I_m \otimes \mathrm{DFT}_n) \cdot P$$

with a permutation matrix P and a “Twiddle matrix” T. We don't care about P because permutations are for free in the circuit model of computation (just reorder the wires). The twiddle matrix T is defined by

$$T = \mathrm{diag}(\Delta^0, \ldots, \Delta^{m-1})$$

with

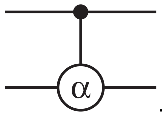
$$\Delta = \mathrm{diag}(\omega^0, \omega^1, \ldots, \omega^{n-1}).$$

The decomposition of $q = 2 \cdot (q/2)$ leads to the well-known classical divide and conquer algorithm. For the quantum algorithm, we will use the decomposition $q = (q/2) \cdot 2$.

First, we note that $\mathrm{QFT}_2 = H$, so the quantum fourier transform of one single qubit is possible at the cost of one single quantum gate. For the circuits of higher dimension, let the unitary operator

$$T_\alpha := \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \alpha \end{pmatrix} \qquad (\alpha \neq 0)$$

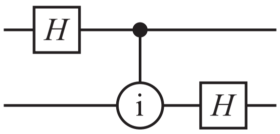
be denoted by the gate symbol



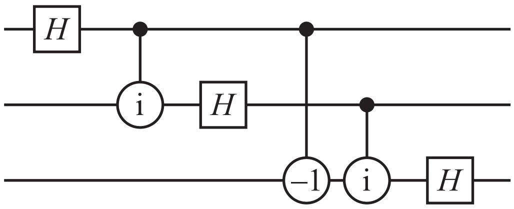
The gates T_α will be used to implement the twiddle matrix in the decomposition. Consider the case $q = 4$. Then

$$\begin{aligned} \mathrm{QFT}_4 &= (\mathrm{QFT}_2 \otimes I_2) \cdot T \cdot (I_2 \otimes \mathrm{QFT}_2) \\ \text{with} \quad T &= \mathrm{diag}(1, 1, 1, \mathrm{i}), \end{aligned}$$

and this representation can be directly translated into the following circuit.



Larger QFT circuits are build in the same fashion. The circuit for the next size, $q = 8 = 2^3$, has the form



corresponding to the decomposition

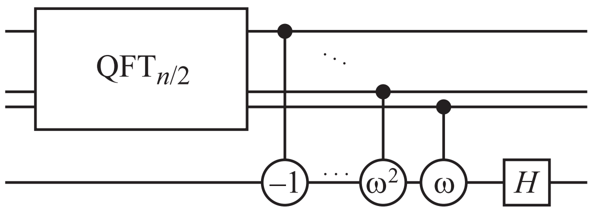
$$\begin{aligned} \mathrm{QFT}_8 &= (\mathrm{QFT}_4 \otimes I_2) \cdot T \cdot (I_4 \otimes \mathrm{QFT}_4) \\ \text{with} \quad T &= \mathrm{diag}(1, 1, 1, \mathrm{i}, 1, -1, 1, -\mathrm{i}). \end{aligned}$$

What might not be obvious at first glance is the correctness of the twiddle matrix' implementation, but indeed, compare the matrix rows to the corresponding basis vectors (tensors):

$$\begin{array}{rr} |000\rangle : & 1 \\ |001\rangle : & 1 \\ |010\rangle : & 1 \\ |011\rangle : & \mathrm{i} \\ |100\rangle : & 1 \\ |101\rangle : & -1 \\ |110\rangle : & 1 \\ |111\rangle : & -\mathrm{i} \end{array}$$

Indeed, T_{i} applies whenever the second and the third bit are 1, and T_{-1} applies whenever the the first and the third bit are 1.

Of course, this can be shown to work in general, giving us the circuit family



for the QFT$_n$ ($q = n^l$). Note that this circuits are of size $l(l-1)/2 = \mathrm{O}(l^2)$ only whereas the complexity of the corresponding classical algorithm is $\mathrm{O}(n \log n)$, i.e., exponentially higher.

3.3 Factoring Integers

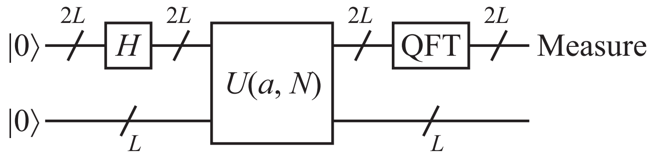
We are now ready to present Shor's integer factorisation algorithm. Let N be an L-digit number. Suppose further that $a > 1$ is coprime to N. The residues $a^i\,(N)$ ($i \in \mathbb{N}$) form a finite cyclic group. Let r be the order of this group, i.e., let $r \in \mathbb{N}$ be the least number such that $a^r \equiv 1\,(N)$. If we are lucky then r is even, and we can rewrite

$$\begin{array}{rr} & a^r \equiv 1\ (N) \\ \Longleftrightarrow & (a^{r/2})^2 - 1^2 \equiv 0\ (N) \\ \Longleftrightarrow & (a^{r/2}-1)(a^{r/2}+1) \equiv 0\ (N). \end{array}$$

If we have a bit more luck, then $a^{r/2} \not\equiv \pm 1\,(N)$, and in this case, $\gcd(a^{r/2}-1, N)$ gives a nontrivial factor of N.

The idea outlined in the previous paragraph is common to many algorithms for factoring integers also for classical computers. For instance, Pollard's rho method [1] is based on the same trick. Basic number theoretic arguments show that if N is large, then almost every $a < N$ will be coprime to N (if not, we have anyway found a factor when we check $\gcd(a, N) = 1$), and almost every a has an even order.

The only problem is to efficiently find the order r. This is the main step of Shor's algorithm which is depicted in the following circuit.



The machine has two registers, one of length $2L$ and one of length L. The operation $U(a,N)$ is supposed to map (x,y) to $(x, y \oplus a^x\ (N))$. The "$y\oplus$" is only needed to make $U(a,N)$ unitary, as required, but actually we are only interested in computing $a^x\ (N)$ and output it to the second register. The computation of $U(a,N)$ can be done in time $\mathrm{O}(l^3)$ using repeated squaring and classical multiplication algorithms which can be implemented on a quantum computer without loss of complexity. The Hadamard transform on the left is meant to be applied to each qubit of this register independently. This leads to a cost of $2l = \mathrm{O}(l)$ gates. The quantum fourier transform at the end can be implemented in a circuit with $\mathrm{O}(l^2)$ gates as has been shown in the previous section.

Let's see what the circuit computes. Let $N \in \mathbb{N}$ be an L-bit number, not even (otherwise 2 is a factor), and not power of a prime. Let $a > 1$ be coprime to N. We denote the content of the two registers by different ket signs, so that the initial state of the machine is $|0\rangle|0\rangle$. This means that the first $|0\rangle$ denotes $2L$ qubits and the second $|0\rangle$ denotes L qubits all of which are set to 0.

The first step is to apply Hadamard transforms to each of the qubits in the first register. This transforms the machine into the state

$$\frac{1}{2^{L/2}} \sum_{x=0}^{2^L-1} |x\rangle|0\rangle.$$

In the next step, the first register is raised to the power of x modulo N; the result is put into the second register:

$$\frac{1}{2^{L/2}} \sum_{x=0}^{2^L-1} |x\rangle|a^x\ (N)\rangle.$$

Now, a quantum fourier transform is applied to the first register, leaving the machine in the state

$$\frac{1}{2^L} \sum_{x=0}^{2^L-1} \Big(\sum_{y=0}^{2^L-1} \omega^{xy}|y\rangle \Big)|a^x\ (N)\rangle.$$

The final step is to measure the two registers.

The measurement will result in a pair $(y_0, a^{x_0}\ (N))$. The probability to observe a particular output $(y_0, a^{x_0}\ (N))$ amounts to

$$\Big| \frac{1}{2^L} \sum_{x:a^x \equiv a^{x_0}\ (N)} \omega^{xy} \Big|^2,$$

because every term a^x that is equal to a^{x_0} modulo N contributes to the probability. Note once again that the classical view regarding the content of the two registers as independend from each other, does not work in the quantum case. Some manipulation of the probability formula shows that the probability distribution strongly favors values of y_0 with $ry_0 \approx k2^{2L}$ for some integer k. Hence,

$$\frac{y_0}{2^{2L}} \approx \frac{k}{r}.$$

The value y_0 is known from the measurement, and r is known to be less than N. By a continued fraction expansion, k/r can be guessed to be the closest faction to $y_0/2^{2L}$ whose denominator is less than N. This finally determines the order r which we were after.

Every quantum algorithm is inherently probabilistic because every measurement of a computational result is necessarily indeterministic. However, there is even more "probabilism" in Shor's factoring algorithm: Hoping for r to be even, hoping for $a^{r/2} \neq \pm 1$, and hoping for guessing k/r correctly.

Anyway, it can be shown that the factoring algorithm has a reasonalbe probability to succeed, and its complexity is only cubic in the length of the number to be factored.

4 Conclusion

Integers can be factored a great deal faster on quantum computers than on classical ones. As many algorithms in symbolic computation are of enormous complexity, it might be worthwhile to investigate how which speed-up can be achived for other problems. For instance, it would be interesting know how fast a Gröbner basis can be computed by a quantum computer.

In the case of integer factorisation, the mathematical background is not too involved. However, development of other quantum algorithms for problems of symbolic computation might well also bring new ideas to light that are also interesting for their own. In this sense, it does not really matter whether or not it will ever be possible to build large scale quantum computers.

References

[1] Joachim v.z. Gathen, Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press. 2000.

[2] Peter W. Shor. Polynomial-time Algorithms for Prime Factorisation and Discrete Logarithms. In *SIAM J. Computing 26.* pp. 1484–1509.

[3] Peter W. Shor. Quantum Computing. In *Proceedings of the ICM.* 1998.

[4] Artur Ekert, Patrick Hayden and Hitoshi Inamori. *Basic Concepts in Quantum Computing.* Centre for Quantum Computation. 2000.

[5] Stan Gudder. Quantum Computation. *Americal Math Monthly,* March 2003.

[6] John Preskill. *Quantum Computation.* Lecture Notes to course Physics 219/ Computer Science 219.
http://www.theory.caltech.edu/people/preskill/ph229/ 1998.