Basic concepts in quantum computation

Artur Ekert, Patrick Hayden and Hitoshi Inamori
Centre for Quantum Computation,
University of Oxford, Oxford OX1 3PU, United Kingdom

April 22, 2001

1 Qubits, gates and networks

Consider the two binary strings,

$$011, \tag{1}$$
$$111. \tag{2}$$

The first one can represent, for example, the number 3 (in binary) and the second one the number 7. In general three physical bits can be prepared in $2^3 = 8$ different configurations that can represent, for example, the integers from 0 to 7. However, a register composed of three classical bits can store only one number at a given moment of time. Enter qubits and quantum registers:

A *qubit* is a quantum system in which the Boolean states 0 and 1 are represented by a prescribed pair of normalised and mutually orthogonal quantum states labeled as $\{|0\rangle, |1\rangle\}$ [1]. The two states form a 'computational basis' and any other (pure) state of the qubit can be written as a superposition $\alpha|0\rangle+\beta|1\rangle$ for some α and β such that $|\alpha|^2+|\beta|^2=1$. A qubit is typically a microscopic system, such as an atom, a nuclear spin, or a polarised photon. A collection of n qubits is called a *quantum register* of size n.

We shall assume that information is stored in the registers in binary form. For example, the number 6 is represented by a register in state $|1\rangle\otimes|1\rangle\otimes|0\rangle$. In more compact notation: $|a\rangle$ stands for the tensor product $|a_{n-1}\rangle\otimes|a_{n-2}\rangle\ldots|a_1\rangle\otimes|a_0\rangle$, where $a_i\in\{0,1\}$, and it represents a quantum register prepared with the value $a = 2^0a_0+2^1a_1+\ldots 2^{n-1}a_{n-1}$. There are 2^n states of this kind, representing all binary strings of length n or numbers from 0 to 2^n-1, and they form a convenient computational basis. In the following $a\in\{0,1\}^n$ (a is a binary string of length n) implies that $|a\rangle$ belongs to the computational basis.

Thus a quantum register of size three can store individual numbers such as 3 or 7,

$$|0\rangle\otimes|1\rangle\otimes|1\rangle \equiv |011\rangle\equiv|3\rangle, \tag{3}$$
$$|1\rangle\otimes|1\rangle\otimes|1\rangle \equiv |111\rangle\equiv|7\rangle, \tag{4}$$

but, it can also store the two of them simultaneously. For if we take the first qubit and instead of setting it to $|0\rangle$ or $|1\rangle$ we prepare a superposition $1/\sqrt{2}\,(|0\rangle+|1\rangle)$ then we obtain

$$\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes|1\rangle\otimes|1\rangle \equiv \frac{1}{\sqrt{2}}(|011\rangle+|111\rangle), \tag{5}$$
$$\equiv \frac{1}{\sqrt{2}}(|3\rangle+|7\rangle). \tag{6}$$

In fact we can prepare this register in a superposition of all eight numbers – it is enough to put each qubit into the superposition $1/\sqrt{2}\,(|0\rangle+|1\rangle)$. This gives

$$\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle), \tag{7}$$

which can also be written in binary as (ignoring the normalisation constant $2^{-3/2}$),

$$|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle. \tag{8}$$

or in decimal notation as

$$|0\rangle+|1\rangle+|2\rangle+|3\rangle+|4\rangle+|5\rangle+|6\rangle+|7\rangle, \tag{9}$$

or simply as

$$\sum_{x=0}^{7}|x\rangle\,. \tag{10}$$

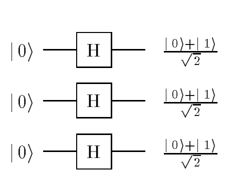
These preparations, and any other manipulations on qubits, have to be performed by unitary operations. A *quantum logic gate* is a device which performs a fixed unitary operation on selected qubits in a fixed period of time and a *quantum network* is a device consisting of quantum logic gates whose computational steps are synchronised in time [2]. The outputs of some of the gates are connected by wires to the inputs of others. The *size* of the network is the number of gates it contains.

The most common quantum gate is the Hadamard gate, a single qubit gate H performing the unitary transformation known as the Hadamard transform. It is defined as

$$H=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad |x\rangle \;—\boxed{H}—\; (-1)^x|x\rangle+|1-x\rangle$$

The matrix is written in the computational basis $\{|0\rangle,|1\rangle\}$ and the diagram on the right provides a schematic representation of the gate H acting on a qubit in state $|x\rangle$, with $x=0,1$.

And here is a network, of size three, which affects the Hadamard transform on three qubits. If they are initially in state $|000\rangle$ then the output is the superposition of all eight numbers from 0 to 7.



IN BINARY

$$= \frac{1}{2^{3/2}} \left\{ \begin{array}{c} |000\rangle + |001\rangle + |010\rangle + |011\rangle + \\ + |100\rangle + |101\rangle + |110\rangle + |111\rangle \end{array} \right\}$$

$$= \frac{1}{2^{3/2}} \left\{ \begin{array}{c} |0\rangle + |1\rangle + |2\rangle + |3\rangle + \\ + |4\rangle + |5\rangle + |6\rangle + |7\rangle \end{array} \right\}$$

IN DECIMAL

If the three qubits are initially in some other state from the computational basis then the result is a superposition of all numbers from 0 to 7 but exactly half of them will appear in the superposition with the minus sign, for example,

$$|101\rangle \mapsto \frac{1}{2^{3/2}} \left\{ \begin{array}{c} |000\rangle - |001\rangle + |010\rangle - |011\rangle + \\ - |100\rangle + |101\rangle - |110\rangle + |111\rangle \end{array} \right\}. \tag{11}$$

In general, if we start with a register of size n in some state $y \in \{0,1\}^n$ then

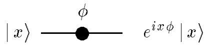
$$|y\rangle \mapsto 2^{-n/2} \sum_{x \in \{0,1\}^n} (-1)^{y \cdot x} |x\rangle, \tag{12}$$

where the product of $y = (y_{n-1}, \ldots, y_0)$ and $x = (x_{n-1}, \ldots, x_0)$ is taken bit by bit:

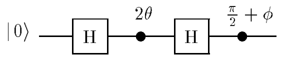
$$y \cdot x = (y_{n-1}x_{n-1} + \ldots y_1 x_1 + y_0 x_0). \tag{13}$$

We will need another single qubit gate – the phase shift gate ϕ defined as $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto e^{i\phi}|1\rangle$, or, in matrix notation,

$$\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \tag{14}$$



The Hadamard gate and the phase gate can be combined to construct the following network (of size four), which generates the most general pure state of a single qubit (up to a global phase),



$$\cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle. \tag{15}$$

Consequently, the Hadamard and phase gates are sufficient to construct *any* unitary operation on a single qubit.

Thus the Hadamard gates and the phase gates can be used to transform the input state $|0\rangle|0\rangle...|0\rangle$ of the n qubit register into any state of the type $|\Psi_1\rangle$ $|\Psi_2\rangle$... $|\Psi_n\rangle$, where $|\Psi_i\rangle$ is an arbitrary superposition of $|0\rangle$ and $|1\rangle$. These are rather special n-qubit states, called the product states or the separable states. In general, a quantum register of size $n > 1$ can be prepared in states which are not separable – they are known as entangled states. For example, for two qubits ($n = 2$), the state

$$\alpha\,|00\rangle + \beta\,|01\rangle = |0\rangle \otimes (\alpha\,|0\rangle + \beta\,|1\rangle) \tag{16}$$

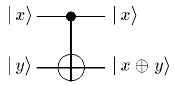
is separable, $|\Psi_1\rangle = |0\rangle$ and $|\Psi_2\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, whilst the state

$$\alpha\,|00\rangle + \beta\,|11\rangle \neq |\Psi_1\rangle \otimes |\Psi_2\rangle \tag{17}$$

is entangled ($\alpha, \beta \neq 0$), because it cannot be written as a tensor product.

In order to entangle two (or more qubits) we have to extend our repertoire of quantum gates to two-qubit gates. The most popular two-qubit gate is the controlled-NOT (C-NOT), also known as the XOR or the measurement gate. It flips the second (target) qubit if the first (control) qubit is $|1\rangle$ and does nothing if the control qubit is $|0\rangle$. The gate is represented by the unitary matrix

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{18}$$



where $x, y = 0$ or 1 and $\oplus$ denotes XOR or addition modulo 2. If we apply the C-NOT to Boolean data in which the target qubit is $|0\rangle$ and the control is either $|0\rangle$ or $|1\rangle$ then the effect is to leave the control unchanged while the target becomes a copy of the control, *i.e.*

$$|x\rangle|0\rangle \mapsto |x\rangle|x\rangle \qquad x = 0, 1. \tag{19}$$

One might suppose that this gate could also be used to copy superpositions such as $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, so that

$$|\Psi\rangle|0\rangle \mapsto |\Psi\rangle|\Psi\rangle \tag{20}$$

for any $|\Psi\rangle$. This is not so! The unitarity of the C-NOT requires that the gate turns superpositions in the control qubit into *entanglement* of the control and the target. If the control qubit is in a superposition state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, ($\alpha, \beta \neq 0$), and the target in $|0\rangle$ then the C-NOT generates the entangled state

$$(\alpha|0\rangle + \beta|1\rangle)\,|0\rangle \mapsto \alpha|00\rangle + \beta|11\rangle. \tag{21}$$

Let us notice in passing that it is impossible to construct a universal quantum cloning machine effecting the transformation in Eq.(20), or even the more general

$$|\Psi\rangle|0\rangle|W\rangle \mapsto |\Psi\rangle|\Psi\rangle|W'\rangle \tag{22}$$

where $|W\rangle$ refers to the state of the rest of the world and $|\Psi\rangle$ is *any* quantum state [3]. To see this take any two normalised states $|\Psi\rangle$ and $|\Phi\rangle$ which are non-identical ($|\langle\Phi|\Psi\rangle| \neq 1$) and non-orthogonal ($\langle\Phi|\Psi\rangle \neq 0$), and run the cloning machine,

$$|\Psi\rangle|0\rangle|W\rangle \quad \mapsto \quad |\Psi\rangle|\Psi\rangle|W'\rangle \tag{23}$$

$$|\Phi\rangle|0\rangle|W\rangle \quad \mapsto \quad |\Phi\rangle|\Phi\rangle|W''\rangle \tag{24}$$

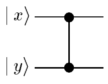
As this must be a unitary transformation which preserves the inner product hence we must require

$$\langle \Phi | \Psi \rangle = \langle \Phi | \Psi \rangle^2 \langle W' | W'' \rangle \tag{25}$$

and this can only be satisfied when $|\langle \Phi | \Psi \rangle| = 0$ or 1, which contradicts our assumptions. Thus states of qubits, unlike states of classical bits, cannot be faithfully cloned. This leads to interesting applications, quantum cryptography being one such.

Another common two-qubit gate is the controlled phase shift gate $B(\phi)$ defined as

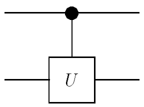
$$B(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$$



$$\Bigg\} e^{ixy\phi} \,|\,x\rangle \,|\,y\rangle\,. \tag{26}$$

Again, the matrix is written in the computational basis $\{|\,00\rangle\,, |\,01\rangle\,, |\,10\rangle\,, |\,11\rangle\}$ and the diagram on the right shows the structure of the gate.

More generally, these various 2-qubit controlled gates are all of the form controlled-U, for some single-qubit unitary transformation U. The controlled-U gate applies the identity transformation to the auxiliary (lower) qubit when the control qubit is in state $|\,0\rangle$ and applies an arbitrary prescribed U when the control qubit is in state $|\,1\rangle$. The gate maps $|\,0\rangle\,|\,y\rangle$ to $|\,0\rangle\,|\,y\rangle$ and $|\,1\rangle\,|\,y\rangle$ to $|\,1\rangle\,(U\,|\,y\rangle)$, and is graphically represented as



The Hadamard gate, all phase gates, and the C-NOT, form an infinite *universal set of gates i.e.* if the C-NOT gate as well as the Hadamard and all phase gates are available then any n-qubit unitary operation can be simulated exactly with $O(4^n n)$ such gates [4]. (Here and in the following we use the asymptotic notation – $O(T(n))$ means bounded above by $c\,T(n)$ for some constant $c > 0$ for sufficiently large n.) This is not the only universal set of gates. In fact, almost any gate which can entangle two qubits can be used as a universal gate [6, 8]. Mathematically, an elegant choice is a pair of the Hadamard and the controlled-V (C-V) where V is described by the unitary matrix

$$V = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}. \tag{27}$$

The two gates form a finite universal set of gates – networks containing only a finite number of these gates can approximate any unitary transformation on two (and more) qubits. More precisely, if U is any two-qubit gate and $\varepsilon > 0$

then there exists a quantum network of size $O(\log^d(1/\varepsilon))$ (where d is a constant) consisting of only H and C-V gates which computes a unitary operation U' that is within distance ε from U [49]. The metric is induced by the Euclidean norm - we say that U' is within distance ε from U if there exists a unit complex number λ (phase factor) such that $||U - \lambda U'|| \leq \varepsilon$. Thus if U' is substituted for U in a quantum network then the final state $\sum_x \alpha'_x\,|x\rangle$ approximates the final state of the original network $\sum_x \alpha_x\,|x\rangle$ as follows: $\sqrt{\sum_x |\lambda\alpha'_x - \alpha_x|^2} \leq \varepsilon$. The probability of any specified measurement outcome on the final state is affected by at most ε.

A *quantum computer* will be viewed here as a quantum network (or a family of quantum networks)and quantum computation is defined as a unitary evolution of the network which takes its initial state "input" into some final state "output". We have chosen the network model of computation, rather than Turing machines, because it is relatively simple and easy to work with and because it is much more relevant when it comes to physical implementation of quantum computation.

2 Quantum arithmetic and function evaluations

Let us now describe how quantum computers actually compute, how they add and multiply numbers, and how they evaluate Boolean functions by means of unitary operations. Here and in the following we will often use the modular arithmetic [9]. Recall that

$$a \bmod b \tag{28}$$

denotes the remainder obtained by dividing integer b into integer a, which is always a number less than b. Basically $a = b \bmod n$ if $a = b + kn$ for some integer k. This is expressed by saying that a is *congruent* to b modulo n or that b is the *residue* of a modulo n. For example, $1 \bmod 7 = 8 \bmod 7 = 15 \bmod 7 = 50 \bmod 7 = 1$. Modular arithmetic is commutative, associative, and distributive.

$$(a \pm b) \bmod n = ((a \bmod n) \pm (b \bmod n)) \bmod n \tag{29}$$

$$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n \tag{30}$$

$$(a \times (b + c)) \bmod n = (((ab) \bmod n + ((ac) \bmod n)) \bmod n \tag{31}$$

Thus, if you need to calculate, say, $3^8 \bmod 7$ do not use the naive approach and perform seven multiplications and one huge modular reduction. Instead, perform three smaller multiplications and three smaller reductions,

$$((3^2 \bmod 7)^2 \bmod 7)^2 \bmod 7 = (2^2 \bmod 7)^2 \bmod 7 = 16 \bmod 7 = 2. \tag{32}$$

This kind of arithmetic is ideal for computers as it restricts the range of all intermediate results. For l-bit modulus n, the intermediate results of any addition, subtraction or multiplication will not be more than $2l$ bits long. In quantum registers of size n, addition modulo 2^n is one of the most common operations; for all $x \in \{0,1\}^n$ and for any $a \in \{0,1\}^n$,

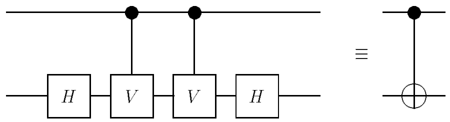
$$|\,x\rangle \mapsto |\,(x + a) \bmod 2^n\rangle \tag{33}$$

is a well defined unitary transformation.

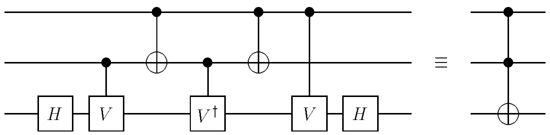
The tricky bit in the modular arithmetic is the inverse operation, and here we need some basic number theory. An integer $a \geq 2$ is said to be *prime* if it is divisible only by 1 and a (we consider only positive divisors). Otherwise, a is called *composite*. The greatest common divisor of two integers a and b is the greatest positive integer d denoted $d = \gcd(a, b)$ that divides both a and b. Two integers a and b are said to be *coprime* or *relatively prime* if $\gcd(a, b) = 1$. Given two integers a and n that are coprime, it can be shown that there exists an unique integer $d \in \{0, \ldots, n-1\}$ such that $ad = 1 \bmod n$ [9]. The integer d is called *inverse modulo* n of a, and denoted a^{-1}. For example, modulo 7 we find that $3^{-1} = 5 \bmod n$, since $3 \times 5 = 15 = 2 \times 7 + 1 = 1 \bmod 7$. This bizarre arithmetic and the notation is due to Karl Friedrich Gauss (1777-1855). It was first introduced in his *Disquistiones Arithmeticae* in 1801.

In quantum computers addition, multiplication, and any other arithmetic operation have to be embedded in unitary evolution. We will stick to the Hadamard and the controlled-V (C-V), and use them as building blocks for all other gates and eventually for quantum adders and multipliers.

If we apply C-V four times we get identity, so any three subsequent applications of C-V give the inverse of C-V, which will be called C-$V^\dagger$. Now, if we have a couple of the C-V gates and a couple of the Hadamard gates we can build the C-NOT as follows



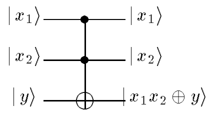
A single qubit operation NOT can be performed via a C-NOT gate if the control qubit is set to $|1\rangle$ and viewed as an auxiliary qubit. This is not to say that we want to do it in practice. The C-NOT gate is much more difficult to build than a single qubit NOT. Right now we are looking into the mathematical structure of quantum Boolean networks and do not care about practicalities. Our two elementary gates also allow us to construct a very useful gate called the controlled-controlled-NOT gate (c^2-NOT) or the Toffoli gate [10]. The construction is given by the following network,



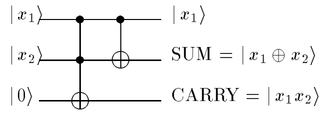
This gate has two control qubits (the top two wires on the diagram) and one target qubit which is negated only when the two controls are in the state $|1\rangle|1\rangle$. The c^2-NOT gate gives us the logical connectives we need for arithmetic. If the target is initially set to $|0\rangle$ the gate acts as a reversible AND gate - after the gate operation the target becomes the logical AND of the two control qubits.

$$|x_1, x_2\rangle\,|0\rangle \mapsto |x_1, x_2\rangle\,|x_1 \wedge x_2\rangle \tag{34}$$

Once we have in our repertoire operations such as NOT, AND, and C-NOT, all of them implemented as unitary operations, we can, at least in principle, evaluate any Boolean function $\{0,1\}^n \to \{0,1\}^m$ which map n bits of input into m bits of output. A simple concatenation of the Toffoli gate and the C-NOT gives a simplified quantum adder, shown below, which is a good starting point for constructing full adders, multipliers and more elaborate networks.



TOFFOLI GATE



QUANTUM ADDER

We can view the Toffoli gate and the evolution given by Eq. (34) as a quantum implementation of a Boolean function $f : \{0,1\}^2 \to \{0,1\}$ defined by $f(x_1, x_2) = x_1 \wedge x_2$. The operation AND is not reversible, so we had to embed it in the reversible operation c^2-NOT. If the third bit is initially set to 1 rather than 0 then the value of $x_1 \wedge x_2$ is negated. In general we write the action of the Toffoli gate as the function evaluation,

$$|x_1, x_2\rangle\,|y\rangle \mapsto |x_1, x_2\rangle\,|(y + (x_1 \wedge x_2)) \bmod 2\rangle\,. \tag{35}$$

This is how we compute any Boolean function $\{0,1\}^n \to \{0,1\}^m$ on a quantum computer. We require at least two quantum registers; the first one, of size n, to store the arguments of f and the second one, of size n, to store the values of f. The function evaluation is then a unitary evolution of the two registers,

$$|x, y\rangle \mapsto |x, (y + f(x)) \bmod 2^m\rangle. \tag{36}$$

for any $y \in \{0,1\}^m$. (In the following, if there is no danger of confusion, we may simplify the notation and omit the mod suffix.)

For example, a network computing $f : \{0,1\}^2 \to \{0,1\}^3$ such that $f(x) = x^2$ acts as follows

$$|00\rangle|000\rangle \mapsto |00\rangle|000\rangle, \qquad |10\rangle|000\rangle \mapsto |10\rangle|100\rangle \tag{37}$$

$$|01\rangle|000\rangle \mapsto |01\rangle|001\rangle, \qquad |11\rangle|000\rangle \mapsto |11\rangle|001\rangle \tag{38}$$

which can be written as

$$|x, 0\rangle \mapsto |x, x^2 \bmod 8\rangle, \tag{39}$$

e.g. $3^2 \bmod 2^3 = 1$ which explains why $|11\rangle|000\rangle \mapsto |11\rangle|001\rangle$.

In fact, for these kind of operations we also need a third register with the so-called working bits which are set to zero at the input and return to zero at the output but which can take non-zero values during the computation.

What makes quantum function evaluation really interesting is its action on a superposition of different inputs x, for example,

$$\sum_x |x, 0\rangle \mapsto \sum_x |x, f(x)\rangle \tag{40}$$

produces $f(x)$ for all x in a single run. The snag is that we cannot get them all from the entangled state $\sum_x |x, f(x)\rangle$ because any bit by bit measurement on the first register will yield one particular value $x' \in \{0,1\}^n$ and the second register will then be found with the value $f(x') \in \{0,1\}^m$.

3 Algorithms and their complexity

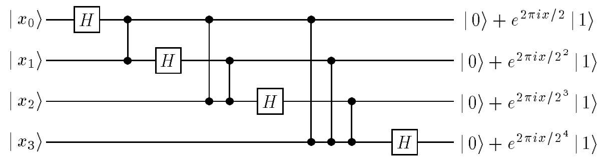
In order to solve a particular problem, computers, be it classical or quantum, follow a precise set of instructions that can be mechanically applied to yield the solution to any given instance of the problem. A specification of this set of instructions is called an algorithm. Examples of algorithms are the procedures taught in elementary schools for adding and multiplying whole numbers; when these procedures are mechanically applied, they always yield the correct result for any pair of whole numbers. Any algorithm can be represented by a family of Boolean networks $(N_1, N_2, N_3, ...)$, where the network N_n acts on all possible input instances of size n bits. Any useful algorithm should have such a family specified by an example network N_n and *a simple rule* explaining how to construct the network N_{n+1} from the network N_n. These are called *uniform* families of networks [11].[1]

The quantum Hadamard transform defined by Eq.(12) has a uniform family of networks whose size is growing as n with the number of input qubits. Another good example of a uniform family of networks is the quantum Fourier transform (QFT) [12] defined in the computational basis as the unitary operation

$$|\,y\rangle \mapsto 2^{-n/2} \sum_x e^{i\frac{2\pi}{2^n}yx}\,|\,x\rangle\,, \tag{41}$$

Suppose we want to *construct* such a unitary evolution of n qubits using our repertoire of quantum logic gates. We can start with a single qubit and notice that in this case the QFT is reduced to applying a Hadamard gate. Then we can take two qubits and notice that the QFT can be implemented with two Hadamard gates and the controlled phase shift $B(\pi)$in between. Progressing this way we can construct the three qubit QFT and the four qubit QFT, whose network looks like this:

[1]This means that the network model is not a self-contained model of computation. We need an algorithm, a Turing machine, which maps each n into an explicit description of N_n.



$$H\;\,B(\pi)\;\,H\;\,B(\pi/2)B(\pi)\;\,H\;\,B(\pi/4)B(\pi/2)B(\pi)\;\,H$$

(*N.B.* there are three different types of the $B(\phi)$ gate in the network above: $B(\pi)$, $B(\pi/2)$ and $B(\pi/4)$.)

The general case of n qubits requires a trivial extension of the network following the same sequence pattern of gates H and B. The QFT network operating on n qubits contains n Hadamard gates H and $n(n-1)/2$ phase shifts B, in total $n(n+1)/2$ elementary gates.

The big issue in designing algorithms or their corresponding families of networks is the optimal use of physical resources required to solve a problem. Complexity theory is concerned with the inherent cost of computation in terms of some designated elementary operations, memory usage, or network size. An algorithm is said to be fast or efficient if the number of elementary operations taken to execute it increases no faster than a polynomial function of the size of the input. We generally take the input size to be the total number of bits needed to specify the input (for example, a number N requires $\log_2 N$ bits of binary storage in a computer). In the language of network complexity - an algorithm is said to be *efficient* if it has a uniform and polynomial-size network family ($O(n^d)$ for some constant d) [11]. For example, the quantum Fourier transform can be performed in an efficient way because it has a uniform family of networks whose size grows only as a quadratic function of the size of the input, *i.e.* $O(n^2)$. Changing from one set of gates to another, *e.g.* constructing the QFT out of the Hadamard and the controlled-V gates with a prescribed precision ϵ, can only affect the network size by a multiplicative constant which does not affect the quadratic scaling with n. Thus the complexity of the QFT is $O(n^2)$ no matter which set of adequate gates we use. Problems which do not have efficient algorithms are known as hard problems.

Elementary arithmetic operations taught at schools, such as long addition, multiplication or division of n bit numbers require $O(n^2)$ operations. For example, to multiply $x = (x_{n-1}...x_1x_0)$ and $y = (y_{n-1}...y_1y_0)$ we successively multiply y by x_0, x_1 and so on, shift, and then add the result. Each multiplication of y by x_k takes about n single bit operations, the addition of the n products takes of the order of n^2 bit operations, which adds to the total $O(n^2)$ operations. Knowing the complexity of elementary arithmetic one can often assess the complexity of other algorithms. For example, the greatest common divisor of two integers x and $y < x$ can be found using Euclid's algorithm; the oldest nontrivial algorithm which has been known and used since 300 BC.[2] First divide x by y obtaining remainder r_1. Then divide y by r_1 obtaining remainder r_2, then divide r_1 by r_2 obtaining remainder r_3, etc., until the remainder is zero. The last non-zero remainder is $\gcd(x, y)$ because it divides all previous remainders and hence also x and y (it is obvious from the construction that it is the *greatest* common divisor). For example, here is a sequence or remainders (r_j, r_{j+1}) when we apply Euclid's algorithm to compute $\gcd(12378, 3054) = 6$: (12378,3054), (3054,162), (162, 138), (138, 24), (24, 18), (18,6), (6,0). What is the complexity of this algorithm? It is easy to see that the largest of the two numbers is at least halved every two steps, so every two steps we need one bit less to represent the number, and so the number of steps is at most $2n$, where n is the number of bits in the two integers. Each division can be done with at most $O(n^2)$ operations hence the total number of operations is $O(n^3)$.

There are basically three different types of Boolean networks: classical deterministic, classical probabilistic, and quantum. They correspond to, respectively, deterministic, randomised, and quantum algorithms.

Classical deterministic networks are based on logical connectives such as AND, OR, and NOT and are required to always deliver correct answers. If a problem admits a deterministic uniform network family of polynomial size, we say that the problem is in the class P [11].

Probabilistic networks have additional "coin flip" gates which do not have any inputs and emit one uniformly-distributed random bit when executed during a computation. Despite the fact that probabilistic networks may generate erroneous answers they may be more powerful than deterministic ones. A good example is primality testing – given an n-bit number x decide whether or not x is prime. The smallest known uniform deterministic network family that solves this problem is of size $O(n^{d \log\log n})$, which is not polynomially bounded. However, there is a probabilistic algorithm, due to Solovay and Strassen [13], that can solve the same problem with a uniform probabilistic network family of size $O(n^3 \log(1/\epsilon))$, where ϵ is the probability of error. *N.B.* ϵ does not depend on n and we can choose it as small as we wish and still get an efficient algorithm.

The $\log(1/\epsilon)$ part can be explained as follows. Imagine a probabilistic network that solves a decision problem [3] and that errs with probability smaller than $\frac{1}{2} + \delta$ for fixed $\delta > 0$. If you run r of these networks in parallel (so that the size of the overall network is increased by factor r) and then use the majority voting for the final YES or NO answer your overall probability of error will bounded by $\epsilon = \exp(-\delta^2 r)$. (This follows directly from the Chernoff bound- see for instance, [14]). Hence r is of the order $\log(1/\epsilon)$. If a problem admits such a family of networks then we say the problem is in the class BPP (stands for "bounded-error probabilistic polynomial") [11].

Last but not least we have quantum algorithms, or families of quantum networks, which are more powerful than their probabilistic counterparts. The

[2] This truly 'classical' algorithm is described in Euclid's *Elements*, the oldest Greek treatise in mathematics to reach us in its entirety. Knuth (1981) provides an extensive discussion of various versions of Euclid's algorithm.

[3] A decision problem is a problem that admits only two answers: YES or NO

example here is the factoring problem – given an n-bit number x find a list of prime factors of x. The smallest known uniform probabilistic network family which solves the problem is of size $O(2^{d\sqrt{n \log n}})$. One reason why quantum computation is such a fashionable field today is the discovery, by Peter Shor, of a uniform family of quantum networks of $O(n^2 \log\log n \log(1/\epsilon))$ in size, that solve the factoring problem [15]. If a problem admits a uniform quantum network family of polynomial size that for any input gives the right answer with probability larger than $\frac{1}{2} + \delta$ for fixed $\delta > 0$ then we say the problem is in the class BQP (stands for "bounded-error quantum probabilistic polynomial").

We have

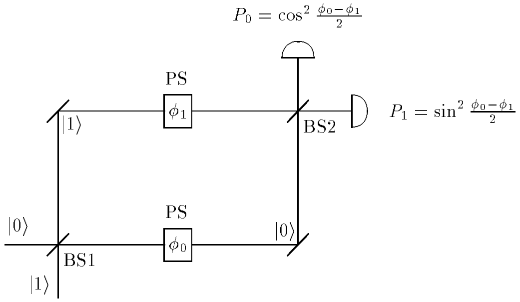
$$P \subseteq BPP \subseteq BQP \tag{42}$$

Quantum networks are potentially more powerful because of multiparticle quantum interference, an inherently quantum phenomenon which makes the quantum theory radically different from any classical statistical theory.

Richard Feynman [16] was the first to anticipate the unusual power of quantum computers. He observed that it appears to be impossible to simulate a general quantum evolution on a classical probabilistic computer in an *efficient* way *i.e.* any classical simulation of quantum evolution appears to involve an exponential slowdown in time as compared to the natural evolution since the amount of information required to describe the evolving quantum state in classical terms generally grows exponentially in time. However, instead of viewing this fact as an obstacle, Feynman regarded it as an opportunity. Let us then follow his lead and try to construct a computing device using inherently quantum mechanical effects.

4 From interferometers to computers

A single particle interference in the Mach-Zehnder interferometer works as follows. A particle, in this case a photon, impinges on a beam-splitter (BS1), and, with some probability amplitudes, propagates via two different paths to another beam-splitter (BS2) which directs the particle to one of the two detectors. Along each path between the two beam-splitters, is a phase shifter (PS).



If the lower path is labeled as state $|\,0\rangle$ and the upper one as state $|\,1\rangle$ then the particle, initially in path $|\,0\rangle$, undergoes the following sequence of transformations

$$
\begin{aligned}
|\,0\rangle &\overset{\text{BS1}}{\mapsto} \frac{1}{\sqrt{2}}\left(|\,0\rangle + |\,1\rangle\right) \overset{\text{PS}}{\mapsto} \frac{1}{\sqrt{2}}(e^{i\phi_0}\,|\,0\rangle + e^{i\phi_1}\,|\,1\rangle) && (43)\\
&= e^{i\frac{\phi_0+\phi_1}{2}} \frac{1}{\sqrt{2}}(e^{i\frac{\phi_0-\phi_1}{2}}\,|\,0\rangle + e^{i\frac{-\phi_0+\phi_1}{2}}\,|\,1\rangle) \\
&\overset{\text{BS2}}{\mapsto} e^{i\frac{\phi_1+\phi_2}{2}} (\cos\tfrac{1}{2}(\phi_0-\phi_1)\,|\,0\rangle + i\sin\tfrac{1}{2}(\phi_0-\phi_1)\,|\,1\rangle), && (44)
\end{aligned}
$$

where ϕ_0 and ϕ_1 are the settings of the two phase shifters and the action of the beam-splitters is defined as

$$|\,0\rangle \mapsto \tfrac{1}{\sqrt{2}}(|\,0\rangle + |\,1\rangle), \quad |\,1\rangle \mapsto \tfrac{1}{\sqrt{2}}(|\,0\rangle - |\,1\rangle). \qquad (45)$$

(We have ignored the phase shift in the reflected beam.) The global phase shift $e^{i\frac{\phi_0+\phi_0}{2}}$ is irrelevant as the interference pattern depends on the difference between the phase shifts in different arms of the interferometer. The phase shifters in the two paths can be tuned to effect any prescribed relative phase shift $\phi = \phi_0 - \phi_1$ and to direct the particle with probabilities

$$
\begin{aligned}
P_0 &= \cos^2\left(\frac{\phi}{2}\right) = \frac{1}{2}\left(1+\cos\phi\right) && (46)\\
P_1 &= \sin^2\left(\frac{\phi}{2}\right) = \frac{1}{2}\left(1-\cos\phi\right) && (47)
\end{aligned}
$$

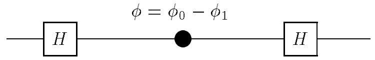
respectively to detectors "0" and "1".

The roles of the three key ingredients in this experiment are clear. The first beam splitter prepares a superposition of possible paths, the phase shifters modify quantum phases in different paths and the second beam-splitter combines all the paths together erasing all information about which path was actually taken

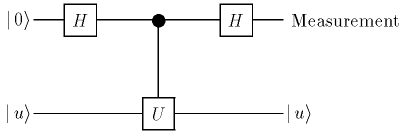
by the particle between the two beam-splitters. This erasure is very important as we shall see in a moment.

Needless to say, single particle interference experiments are not restricted to photons. One can go for a different "hardware" and repeat the experiment with electrons, neutrons, atoms or even molecules. When it comes to atoms and molecules both external and internal degrees of freedom can be used.

Although single particle interference experiments are worth discussing in their own right, here we are only interested in their generic features simply because they are all "isomorphic" and once you know and understand one of them you, at least for our purposes, understand them all (modulo experimental details, of course). Let us now describe any single particle interference experiment in more general terms. It is very convenient to view this experiment in a diagramatic way as a *quantum network* with three quantum logic gates [17]. The beam-splitters will be now called the Hadamard gates and the phase shifters the phase shift gates. In particular any single particle quantum interference can be represented by the following simple network,



In order to make a connection with a quantum function evaluation let us now describe an alternative construction which simulates the action of the phase shift gate. This construction introduces a phase factor ϕ using a controlled-U gate. The phase shift ϕ is "computed" with the help of an auxiliary qubit in a prescribed state $|\,u\rangle$ such that $U\,|\,u\rangle = e^{i\phi}\,|\,u\rangle$.



In our example, shown above, we obtain the following sequence of transformations on the two qubits

$$
\begin{aligned}
|\,0\rangle\,|\,u\rangle \overset{H}{\mapsto} \tfrac{1}{\sqrt{2}}(|\,0\rangle + |\,1\rangle)\,|\,u\rangle &\overset{c-U}{\mapsto} \tfrac{1}{\sqrt{2}}(|\,0\rangle + e^{i\phi}\,|\,1\rangle)\,|\,u\rangle \\
&\overset{H}{\mapsto} (\cos\tfrac{\phi}{2}\,|\,0\rangle + i\sin\tfrac{\phi}{2}\,|\,1\rangle)\,|\,u\rangle\,. && (48)
\end{aligned}
$$

We note that the state of the auxiliary qubit $|\,u\rangle$, being an eigenstate of U, is not altered along this network, but its eigenvalue $e^{i\phi}$ is "kicked back" in front of the $|\,1\rangle$ component in the first qubit. The sequence (48) is the exact simulation of the Mach-Zehnder interferometer and, as we shall see later on, the kernel of quantum algorithms.

Some of the controlled-U operations are special - they represent quantum function evaluations! Indeed, a unitary evolution which computes $f:\{0,1\}^n \mapsto \{0,1\}^m$,

$$|\,x\rangle\,|\,y\rangle \mapsto |\,x\rangle\,|\,(y+f(x)) \bmod 2^m\rangle\,, \tag{49}$$

is of the controlled-U type. The unitary transformation of the second register, specified by

$$|\,y\rangle \mapsto |\,(y+f(x)) \bmod 2^m\rangle\,, \tag{50}$$

depends on x – the state of the first register. If the initial state of the second register is set to

$$|\,u\rangle = \frac{1}{2^{m/2}}\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}y\right)|y\rangle, \tag{51}$$

by applying the QFT to the state $|\,111...1\rangle$, then the function evaluation generates

$$\begin{aligned}
|x\rangle\,|\,u\rangle &= \frac{1}{2^{m/2}}|x\rangle\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}y\right)|y\rangle && (52)\\
&\mapsto \frac{1}{2^{m/2}}|x\rangle\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}y\right)|f(x)+y\rangle && (53)\\
&= \frac{e^{\frac{2\pi i}{2^m}f(x)}}{2^{m/2}}|x\rangle\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}(f(x)+y)\right)|f(x)+y\rangle && (54)\\
&= \frac{e^{\frac{2\pi i}{2^m}f(x)}}{2^{m/2}}|x\rangle\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}y\right)|y\rangle && (55)\\
&= e^{\frac{2\pi i}{2^m}f(x)}|x\rangle\,|\,u\rangle\,, && (56)
\end{aligned}$$

where we have relabelled the summation index in the sum containing 2^m terms

$$\sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}(f(x)+y)\right)|f(x)+y\rangle = \sum_{y=0}^{2^m-1}\exp\left(-\frac{2\pi i}{2^m}y\right)|y\rangle. \tag{57}$$

Again, the function evaluation effectively introduces the phase factors in front of the $|x\rangle$ terms in the first register.

$$|x\rangle\,|\,u\rangle \mapsto \exp\left(\frac{2\pi i}{2^m}f(x)\right)|x\rangle\,|\,u\rangle \tag{58}$$

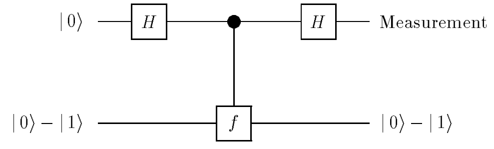
Please notice that the resolution in $\phi(x) = \frac{2\pi}{2^m}f(x)$ is determined by the size m of the second register. For $m=1$ we obtain $\phi(x)=\pi f(x)$, *i.e.* the phase factors are $(-1)^{f(x)}$. Let us see how this approach explains the internal working of quantum algorithms.

5 The first quantum algorithms

The first quantum algorithms showed advantages of quantum computation without referring to computational complexity measured by the scaling properties of network sizes. The computational power of quantum interference was discovered by counting how many times certain Boolean functions have to be evaluated in order to find the answer to a given problem. Imagine a "black box" (also called an *oracle*) computing a Boolean function and a scenario in which one wants to learn about a given property of the Boolean function but has to pay for each use of the "black box" (often referred to as a *query*). The objective is to minimise number of queries.

Consider, for example, a "black box" computing a Boolean function f : $\{0,1\}\mapsto\{0,1\}$. There are exactly four such functions: two constant functions ($f(0)=f(1)=0$ and $f(0)=f(1)=1$) and two "balanced" functions ($f(0)=0, f(1)=1$ and $f(0)=1, f(1)=0$). The task is to deduce, by queries to the "black box", whether f is constant or balanced (in other words, whether $f(0)$ and $f(1)$ are the same or different).

Classical intuition tells us that we have to evaluate both $f(0)$ and $f(1)$, which involves evaluating f twice (two queries). We shall see that this is not so in the setting of quantum information, where we can solve this problem with a single function evaluation (one query), by employing an algorithm that has the same mathematical structure as the Mach-Zehnder interferometer. The quantum algorithm that accomplishes this is best represented as the quantum network shown below, where the middle operation is the "black box" representing the function evaluation [17].



The initial state of the qubits in the quantum network is $|\,0\rangle\,(|\,0\rangle - |\,1\rangle)$ (apart from a normalization factor, which will be omitted in the following). After the first Hadamard transform, the state of the two qubits has the form $(|\,0\rangle + |\,1\rangle)(|\,0\rangle - |\,1\rangle)$. To determine the effect of the function evaluation on this state, first recall that, for each $x \in \{0,1\}$,

$$|\,x\rangle\,(|\,0\rangle - |\,1\rangle) \stackrel{f}{\mapsto} (-1)^{f(x)}\,|\,x\rangle\,(|\,0\rangle - |\,1\rangle). \tag{59}$$

Therefore, the state after the function evaluation is

$$[(-1)^{f(0)}\,|\,0\rangle + (-1)^{f(1)}\,|\,1\rangle](|\,0\rangle - |\,1\rangle)\,. \tag{60}$$

That is, for each x, the $|\,x\rangle$ term acquires a phase factor of $(-1)^{f(x)}$, which corresponds to the eigenvalue of the state of the auxiliary qubit under the action of the operator that sends $|\,y\rangle$ to $|\,y+f(x)\rangle$. The second qubit is of no interest to us any more but the state of the first qubit

$$(-1)^{f(0)}\,|\,0\rangle + (-1)^{f(1)}\,|\,1\rangle \tag{61}$$

is equal either to

$$\pm\,(|\,0\rangle + |\,1\rangle)\,, \tag{62}$$

when $f(0) = f(1)$, or

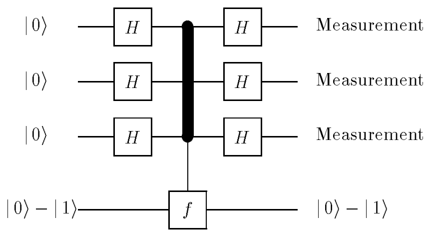
$$\pm\,(|\,0\rangle - |\,1\rangle)\,, \tag{63}$$

when $f(0) \neq f(1)$. Hence, after applying the second Hadamard gate the state of the first qubit becomes $|\,0\rangle$ if the function f is constant and $|\,1\rangle$ if the function is balanced! A bit-value measurement on this qubit distinguishes these cases with certainty.

This example [17] is an improved version of the first quantum algorithm proposed by Deutsch [18] (The original Deutsch algorithm provides the correct answer with probability 50%.) Deutsch's result laid the foundation for the new field of quantum computation, and was followed by several other quantum algorithms.

Deutsch's original problem was subsequently generalised to cover "black boxes" computing Boolean functions $f : \{0,1\}^n \mapsto \{0,1\}$. Assume that, for one of these functions, it is "promised" that it is either constant or balanced (*i.e.* has an equal number of 0's outputs as 1's), and the goal is to determine which of the two properties the function actually has. How many queries to f are required to do this? Any classical algorithm for this problem would, in the worst-case, require $2^{n-1}+1$ queries before determining the answer with certainty. There is a quantum algorithm that solves this problem with a single evaluation of f.

The algorithm is illustrated by a simple extension of the network which solves Deutsch's problem.



The control register, now composed out of n qubits ($n = 3$ in the diagram above), is initially in state $|\,00\cdots0\rangle$ and an auxiliary qubit in the second register starts and remains in the state $|\,0\rangle - |\,1\rangle$.

Stepping through the execution of the network, the state after the first n-qubit Hadamard transform is applied is

$$\sum_x |\,x\rangle\,(|\,0\rangle - |\,1\rangle)\,, \tag{64}$$

which, after the function evaluation, is

$$\sum_x (-1)^{f(x)}\,|\,x\rangle\,(|\,0\rangle - |\,1\rangle). \tag{65}$$

Finally, after the last Hadamard transform, the state is

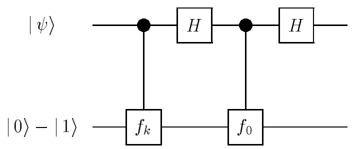
$$\sum_{x,y} (-1)^{f(x)+(x\cdot y)}\,|\,y\rangle\,(|\,0\rangle - |\,1\rangle). \tag{66}$$

Note that the amplitude of $|\,00\cdots0\rangle$ is $\sum_x \frac{(-1)^{f(x)}}{2^n}$ which is $(-1)^{f(0)}$ when f is constant and 0 when f is balanced. Therefore, by measuring the first n qubits, it can be determined with certainty whether f is constant or balanced. The algorithm follows the same pattern as Deutsch's algorithm: the Hadamard transform, a function evaluation, the Hadamard transform (the H-f-H sequence). We recognize it as a generic interference pattern.

6 Quantum search

The generic H-f-H sequence may be repeated several times. This can be illustrated, for example, with Grover's data base search algorithm [19]. Suppose we are given, as an oracle, a Boolean function f_k which maps $\{0,1\}^n$ to $\{0,1\}$ such that $f_k(x) = \delta_{xk}$ for some k. Our task is to find k. Thus in a set of numbers from 0 to $2^n - 1$ one element has been "tagged" and by evaluating f_k we have to find which one. In order to find k with probability of 50% any classical algorithm, be it deterministic or randomised, will need to evaluate f_k a minimum of 2^{n-1} times. In contrast, a quantum algorithm needs only $O(2^{n/2})$ evaluations.

Unlike the algorithms studied so far, Grover's algorithm consists of *repeated* applications of the *same* unitary transformation many ($O(2^{n/2})$) times. The initial state is chosen to be the one that has equal overlap with each of the computational basis states: $|\,S\rangle = 2^{-n/2}\sum_{i=0}^{2^n}|\,i\rangle$. The operation applied at each individual iteration, referred to as the Grover iterate, can be best represented by the following network:



The components of the network are by now familiar: Hadamard transforms (H) and controlled-f gates. It is important to notice that in drawing the network we have used a shorthand notation: the first register (with the $|\psi\rangle$ input) actually consists of n qubits. The Hadamard transform is applied to each of those qubits and the controlled-f gates act on all of them simultaneously. Also, the input to the second register is always $|0\rangle - |1\rangle$ but the input to the first register, denoted $|\psi\rangle$ changes from iteration from iteration, as the calculation proceeds. As usual, the second register will be ignored since it remains constant throughout the computation.

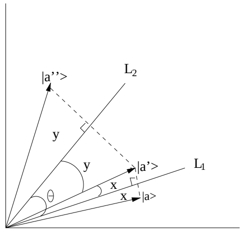
To begin, consider only the controlled-f_k gate. This is just the phase-kickback construction that was introduced in Section 4 but for the specific function f_k. In particular, the transformation does nothing to any basis elements except for $|k\rangle$, which goes to $-|k\rangle$. Geometrically, this is simply a reflection in the hyperplane perpendicular to $|k\rangle$ so let us call it R_k.

Similarly, with respect to the first register only, the controlled-f_0 operation sends $|0\rangle$ to $-|0\rangle$ and fixes all other basis elements, so it can be written R_0. Now consider the sequence of operations HR_0H. Since $H^2 = I$, we can rewrite the triple as HR_0H^{-1} which is simply R_0 performed in a different basis. More specifically, it is reflection about the hyperplane perpendicular to

$$H|0\rangle = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle = |S\rangle \tag{67}$$

so we will simply write the triple as R_S.

We can therefore rewrite the Grover iterate in the simple form $G = R_S R_k$. Now, since each reflection is an orthogonal transformation with negative determinant, their composition must be an orthogonal transformation with unit determinant, in other words, a rotation. The question, of course, is which rotation. To find the answer it suffices to consider rotations in the plane spanned by $|k\rangle$ and $|S\rangle$ since all other vectors are fixed by the Grover iterate. The generic geometrical situation is then illustrated in the following diagram.



If the vector $|a\rangle$ is reflected through the line L_1 to produce the vector $|a'\rangle$ and then reflected a second time through line L_2 to produce the vector $|a''\rangle$, then the net effect is a rotation by the total subtended angle between $|a\rangle$ and $|a''\rangle$, which is $2x + 2y = 2(x + y) = 2\theta$.

Therefore, writing $|k^\perp\rangle$ and $|S^\perp\rangle$ for plane vectors perpendicular to $|k\rangle$ and $|S\rangle$ respectively, the Grover iterate performs a rotation of twice the angle from $|k^\perp\rangle$ to $|S^\perp\rangle$. Setting, $\sin\phi = \frac{1}{2^{n/2}}$, this is easily seen to be a rotation by

$$2(3\frac{\pi}{2} - \phi) = \pi - 2\phi \bmod 2\pi. \tag{68}$$

Thus, up to phases, the Grover iterate rotates the state vector by an angle 2ϕ towards the desired solution $|k\rangle$. Normally, the initial state for the first register is chosen to be $|S\rangle$. Since this initial state $|S\rangle$ is already at an angle ϕ to $|k\rangle$, the iterate should be repeated m times, where

$$(2m+1)\phi \approx \frac{\pi}{2}, \tag{69}$$

giving

$$m \approx \frac{\pi}{4\phi} - \frac{1}{4} \tag{70}$$

to get a probability of success bounded below by $\cos^2(2\phi)$, which goes to 1 as $n \mapsto \infty$. For large n, $\frac{1}{2^{n/2}} = \sin\phi \approx \phi$, so

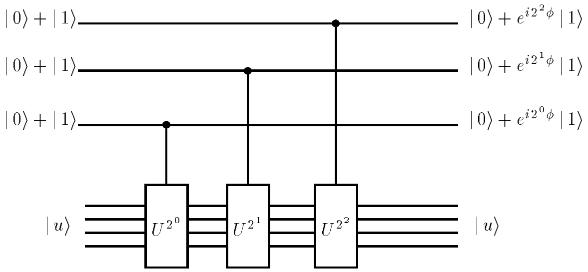
$$m \approx \frac{\pi}{4}\frac{1}{2^{n/2}}. \tag{71}$$

This is an astounding result: any search of an unstructured database can be performed in time proportional to the square-root of the number of entries in the database. Subsequent work extended the result to searches for multiple items [20], searches of structured databases [21], and many other situations. Also, Zalka [22], Boyer et. al [20] and others have demonstrated that Grover's algorithm is optimal, in the sense that any other quantum algorithm for searching an unstructured database must take time at least $O(2^{n/2})$.

7 Optimal phase estimation

Query models of quantum computation provided a natural setting for subsequent discoveries of "real quantum algorithms". The most notable example is Shor's quantum factoring algorithm [15] which evolved from the the order-finding problem, which was originally formulated in the language of quantum queries. Following our "interferometric approach" we will describe this algorithm in the terms of multiparticle quantum interferometry. We start with a simple eigenvalue or phase estimation problem.

Suppose that U is any unitary transformation on m qubits and $|u\rangle$ is an eigenvector of U with eigenvalue $e^{i\phi}$ and consider the following scenario. We do not explicitly know U or $|u\rangle$ or $e^{i\phi}$, but instead we are given devices that perform controlled-U, controlled-U^{2^1}, controlled-U^{2^2} and so on until we reach controlled-$U^{2^{n-1}}$. Also, assume that we are given a single preparation of the state $|u\rangle$. Our goal is to obtain an n-bit estimator of ϕ. We start by constructing the following network,



The second register of m qubits is initially prepared in state $|u\rangle$ and remains in this state after the computation, whereas the first register of n qubits evolves into the state,

$$(|0\rangle + e^{i2^{n-1}\phi}|1\rangle)(|0\rangle + e^{i2^{n-2}\phi}|1\rangle)\cdots(|0\rangle + e^{i\phi}|1\rangle) = \sum_{y=0}^{2^n-1} e^{2\pi i \frac{\phi y}{2^n}} |y\rangle . \quad (72)$$

Consider the special case where $\phi = 2\pi x/2^n$ for $x = \sum_{i=0}^{n-1} 2^i x_i$, and recall the quantum Fourier transform (QFT) introduced in Section 2. The state which gives the binary representation of x, namely, $|x_{n-1}\cdots x_0\rangle$ (and hence ϕ) can be obtained by applying the inverse of the QFT , that is by running the network for the QFT in the backwards direction (consult the diagram of the QFT). If x is an n-bit number this will produce the exact value ϕ.

However, ϕ does not have to be a fraction of a power of two (and may not even be a rational number). For such a ϕ, it turns out that applying the inverse of the QFT produces the best n-bit approximation of ϕ with probability at least $4/\pi^2 \approx 0.405$.

To see why this is so, let us write $\phi = 2\pi(a/2^n + \delta)$, where $a = (a_{n-1}\ldots a_0)$ is the best n-bit estimate of $\frac{\phi}{2\pi}$ and $0 < |\delta| \le 1/2^{n+1}$. Applying the inverse QFT to the state in Eq. (72) now yields the state

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n}(a-x)y} e^{2\pi i \delta y} |x\rangle \quad (73)$$

and the coefficient in front of $|x = a\rangle$ in the above is the geometric series

$$\frac{1}{2^n}\sum_{y=0}^{2^n-1}(e^{2\pi i\delta})^y = \frac{1}{2^n}\left(\frac{1-(e^{2\pi i\delta})^{2^n}}{1-e^{2\pi i\delta}}\right) . \quad (74)$$

Since $|\delta| \le \frac{1}{2^{n+1}}$, it follows that $2^n|\delta| \le 1/2$, and using the inequality $2z \le \sin\pi z \le \pi z$ holding for any $z \in [0, 1/2]$, we get $|1 - e^{2\pi i\delta 2^n}| = 2|\sin(\pi\delta 2^n)| \ge$

$4|\delta|2^n$. Also, $|1 - e^{2\pi i\delta}| = 2|\sin\pi\delta| \le 2\pi\delta$. Therefore, the probability of observing $a_{n-1}\cdots a_0$ when measuring the state is

$$\left|\frac{1}{2^n}\left(\frac{1-(e^{2\pi i\delta})^{2^n}}{1-e^{2\pi i\delta}}\right)\right|^2 \ge \left(\frac{1}{2^n}\left(\frac{4\delta 2^n}{2\pi\delta}\right)\right)^2 = \frac{4}{\pi^2}, \quad (75)$$

which proves our assertion. In fact, the probability of obtaining the best estimate can be made $1-\delta$ for any $0 < \delta < 1$, by creating the state in Eq.(72) but with $n + O(\log(1/\delta))$ qubits and rounding the answer off to the nearest n bits [17].

8 Periodicity and quantum factoring

Amazingly, the application of optimal phase estimation to a very particular unitary operator will allow us to factor integers efficiently. In fact, it will allow us to solve a more general class of problems related to the periodicity of certain integer functions.

Let N be an m-bit integer, and let a be an integer smaller than N, and coprime to N. Define a unitary operator U_a acting on m qubits such that for all $y < N$

$$|y\rangle \mapsto U_a|y\rangle = |ay \bmod N\rangle . \quad (76)$$

This unitary operation can be called multiplication by a modulo N. Since a is coprime to N, as discussed in Section 2, there exists a least strictly positive r such that $a^r = 1 \bmod N$. This r is called the *order* of a modulo N. Equivalently, r is the period of the function $f(x) = a^x \bmod N$, *i.e.* the least $r > 0$ such that $f(x) = f(x+r)$ for all x. We are after the optimal n-bit estimate of this period, given some specified precision n.

Now let the vectors $|u_k\rangle$ $(k \in \{1,\ldots,r\})$ be defined by

$$|u_k\rangle = r^{-1/2}\sum_{j=0}^{r-1} e^{-\frac{2\pi i k j}{r}} |a^j \bmod N\rangle . \quad (77)$$

It is easy to check [23] that for each $k \in \{1,\ldots,r\}$, $|u_k\rangle$ is an eigenvector with eigenvalue $e^{2\pi i\frac{k}{r}}$ of the modular multiplication operator U_a defined above.

It is important to observe that one can efficiently construct a quantum network for controlled multiplication modulo some number N. Moreover, for any j, it is possible to efficiently implement a controlled-$U_a^{2^j}$ gate [24, 25]. Therefore, we can apply the techniques for optimal phase estimation discussed in Section 7. For any $k \in \{1,\ldots,r\}$, given the state $|u_k\rangle$ we can obtain the best n-bit approximation to $\frac{k}{r}$. This is tantamount to determining r itself. Unfortunately, there is a complication.

Our task is: given an m bit long number N and randomly chosen $a < N$ coprime with N, find the order of a modulo N. The problem with the above method is that we are aware of no straightforward efficient way to prepare any of the states $|\,u_k\rangle$. However, the state

$$|\,1\rangle = r^{-1/2}\sum_{k=1}^{r}|\,u_k\rangle \tag{78}$$

is most definitely an easy state to prepare.

If we start with $|\,1\rangle$ in place of the eigenvector $|\,u_k\rangle$, apply the phase estimation network and measure the first register bit by bit we will obtain n binary digits of x such that, with probability exceeding $4/\pi^2$, $\frac{x}{2^n}$ is the best n-bit estimate of $\frac{k}{r}$ for a randomly chosen k from $\{1,\ldots,r\}$. The question is: given x how to compute r? Let us make few observations:

- *k/r is unique, given x.*
  Value $x/2^n$, being the n-bit estimate, differs by at most $1/2^n$ from k/r. Hence, as long as $n > 2m$, the n bit estimate x determines a unique value of $\frac{k}{r}$ since r is an m-bit number.
- *Candidate values for k/r are all convergents to $x/2^m$.*
  For any real number θ, there is a unique sequence of special rationals $(\frac{p_n}{q_n})_{n\in\mathbf{N}}$ ($\gcd(p_n,q_n)=1$) called the *convergents* to θ that tend to θ as n grows. A theorem [9] states that if p and q are integers with $\left|\theta - \frac{p}{q}\right| < \frac{1}{2q^2}$ then p/q is a convergent to θ. Since we have $\frac{1}{2^n} \le \frac{1}{2(2^m)^2} \le \frac{1}{2r^2}$, this implies $\left|\frac{x}{2^n} - \frac{k}{r}\right| < \frac{1}{2r^2}$ and k/r is a convergent to $x/2^n$.
- *Only one convergent is eligible.*
  It is easy to show that there is at most one fraction a/b satisfying both $b \le r$ and $\left|\frac{x}{2^n} - \frac{a}{b}\right| < \frac{1}{2r^2}$.

Convergents can be found efficiently using the well-known *continued fraction* method [9]. Thus we employ continued fractions and our observations above to find a fraction a/b such that $b \le 2^m$ and $\left|\frac{x}{2^n} - \frac{a}{b}\right| < \frac{1}{2^n}$. We get the rational k/r, and $k=a, r=b$, provided k and r are coprime. For randomly chosen k, this happens with probability greater than or equal to $1/\ln r$ [26].

Finally, we show how order-finding can be used to factor a composite number N. Let a be a randomly chosen positive integer smaller than N such that $\gcd(a,N)=1$. Then the order of a modulo N is defined, and we can find it efficiently using the above algorithm. If r is even, then we have:

$$\begin{aligned} a^r &= 1 \bmod N & (79)\\ \Leftrightarrow \quad (a^{r/2})^2 - 1^2 &= 0 \bmod N & (80)\\ \Leftrightarrow \quad (a^{r/2}-1)(a^{r/2}+1) &= 0 \bmod N. & (81)\end{aligned}$$

The product $(a^{r/2}-1)(a^{r/2}+1)$ must be some multiple of N, so unless $a^{r/2} = \pm 1 \bmod N$ at least one of terms must have a nontrivial factor in common with N. By computing the greatest common divisor of this term and N, one gets a non-trivial factor of N.

Furthermore, if N is odd with prime factorisation

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_s^{\alpha_s}, \tag{82}$$

then it can be shown [26] that if $a < N$ is chosen at random such that $\gcd(a,N) = 1$ then the probability that its order modulo N is even and that $a^{r/2} \ne \pm 1$ mod N is:

$$\Pr(r \text{ is even AND } a^{r/2} \ne \pm 1 \bmod N) \ge 1 - \frac{1}{2^{s-1}}. \tag{83}$$

Thus, combining our estimates of success at each step, with probability greater than or equal to

$$\frac{4}{\pi^2}\frac{1}{\ln r}\left(1 - \frac{1}{2^{s-1}}\right) \ge \frac{2}{\pi^2}\frac{1}{\ln N} \tag{84}$$

we find a factor of N [4]. (Here we have used that N is composite and $r < N$.) If N is $\log N = n$ bits long then by repeating the whole process $O(n)$ times, or by a running $O(n)$ computations in parallel by a suitable extension of a quantum factoring network, we can then guarantee that we will find a factor of N with a fixed probability greater than $\frac{1}{2}$. This, and the fact that the quantum network family for controlled multiplication modulo some number is uniform and of size $O(n^2)$, tells us that factoring is in the complexity class BQP.

But why should anybody care about efficient factorisation?

9 Cryptography

Human desire to communicate secretly is at least as old as writing itself and goes back to the beginnings of our civilisation. Methods of secret communication were developed by many ancient societies, including those of Mesopotamia, Egypt, India, and China, but details regarding the origins of cryptology[5] remain unknown [27].

Originally the security of a cryptosystem or a cipher depended on the secrecy of the entire encrypting and decrypting procedures; however, today we use ciphers for which the algorithm for encrypting and decrypting could be revealed to anybody without compromising their security. In such ciphers a set of specific parameters, called a *key*, is supplied together with the plaintext as an input to the encrypting algorithm, and together with the cryptogram as an input to the decrypting algorithm [28]. This can be written as

$$\hat{E}_k(P) = C, \text{ and conversely, } \hat{D}_k(C) = P, \tag{85}$$

where P stands for plaintext, C for cryptotext or cryptogram, k for cryptographic key, and $\hat{E}$ and $\hat{D}$ denote an encryption and a decryption operation respectively.

[4] *N.B.* by Eq.(83), the method fails if N is a prime power, $N = p^\alpha$, but prime powers can be efficiently recognised and factored by classical means.

[5] The science of secure communication is called cryptology from Greek *kryptos* hidden and *logos* word. Cryptology embodies cryptography, the art of code-making, and cryptanalysis, the art of code-breaking.

The encrypting and decrypting algorithms are publicly known; the security of the cryptosystem depends entirely on the secrecy of the key, and this key must consist of a *randomly chosen*, sufficiently long string of bits. Probably the best way to explain this procedure is to have a quick look at the Vernam cipher, also known as the one-time pad [29].

If we choose a very simple digital alphabet in which we use only capital letters and some punctuation marks such as

A	B	C	D	E	...	...	X	Y	Z		?	,	.
00	01	02	03	04	...	...	23	24	25	26	27	28	29

we can illustrate the secret-key encrypting procedure by the following simple example (we refer to the dietary requirements of 007):

S	H	A	K	E	N		N	O	T		S	T	I	R	R	E	D
18	07	00	10	04	13	26	13	14	19	26	18	19	08	17	17	04	03
15	04	28	13	14	06	21	11	23	18	09	11	14	01	19	05	22	07
03	11	28	23	18	19	17	24	07	07	05	29	03	09	06	22	26	10

In order to obtain the cryptogram (sequence of digits in the bottom row) we add the plaintext numbers (the top row of digits) to the key numbers (the middle row), which are randomly selected from between 0 and 29, and take the remainder after division of the sum by 30, that is we perform addition modulo 30. For example, the first letter of the message "S" becomes a number "18"in the plaintext, then we add $18 + 15 = 33$; $33 = 1 \times 30 + 3$, therefore we get 03 in the cryptogram. The encryption and decryption can be written as $P_i + k_i \pmod{30} = C_i$ and $C_i - k_i \pmod{30} = P_i$ respectively for the symbol at position i.

The cipher was invented in 1917 by the American AT&T engineer Gilbert Vernam. It was later shown, by Claude Shannon [30], that as long as the key is truly random, has the same length as the message, and is never reused then the one-time pad is perfectly secure. So, if we have a truly unbreakable system, what is wrong with classical cryptography?

There is a snag. It is called *key distribution*. Once the key is established, subsequent communication involves sending cryptograms over a channel, even one which is vulnerable to total passive eavesdropping (*e.g.* public announcement in mass-media). This stage is indeed secure. However in order to establish the key, two users, who share no secret information initially, must at a certain stage of communication use a reliable and a very secure channel. Since the interception is a set of measurements performed by an eavesdropper on this channel, however difficult this might be from a technological point of view, *in principle* any classical key distribution can always be passively monitored, without the legitimate users being aware that any eavesdropping has taken place.

In the late 1970s Whitfield Diffie and Martin Hellman [31] proposed an interesting solution to the key distribution problem. It involved two keys, one public key π for encryption and one private key κ for decryption:

$$\hat{E}_\pi(P) = C, \text{ and } \hat{D}_\kappa(C) = P. \tag{86}$$

In these systems users do not need to share any private key before they start sending messages to each other. Every user has his own two keys; the public key is publicly announced and the private key is kept secret. Several public-key cryptosystems have been proposed since 1976; here we concentrate our attention on the most popular one namely the RSA [32]. In fact the techniques were first discovered at CESG in the early 1970s by James Ellis, who called them "Non-Secret Encryption" [33]. In 1973, building on Ellis' idea, C. Cocks designed what we now call RSA [34], and in 1974 M. Williamson proposed what is essentially known today as the Diffie-Hellman key exchange protocol.

Suppose that Alice wants to send an RSA encrypted message to Bob. The RSA encryption scheme works as follows:

Key generation Bob picks randomly two distinct and large prime numbers p and q. We denote $n = pq$ and $\phi = (p-1)(q-1)$. Bob then picks a random integer $1 < e < \phi$ that is coprime with ϕ, and computes the inverse d of e modulo ϕ ($\gcd(e, \phi) = 1$). This inversion can be achieved efficiently using for instance the extended Euclidean algorithm for the greatest common divisor[9]. Bob's private key is $\kappa = d$ and his public key is $\pi = (e, n)$

Encryption Alice obtains Bob's public key $\pi = (e, n)$ from some sort of yellow pages or an RSA public key directory. Alice then writes her message as a sequence of numbers using, for example, our digital alphabet. This string of numbers is subsequently divided into blocks such that each block when viewed as a number P satisfies $P \leq n$. Alice encrypts each P as

$$C = \hat{E}_\pi(P) = P^e \bmod n \tag{87}$$

and sends the resulting cryptogram to Bob.

Decryption Receiving the cryptogram C, Bob decrypts it by calculating

$$\hat{D}_\kappa(C) = C^d \bmod n = P \tag{88}$$

where the last equality will be proved shortly.

The mathematics behind the RSA is a lovely piece of number theory which goes back to the XVI century when a French lawyer Pierre de Fermat discovered that if a prime p and a positive integer a are coprime, then

$$a^{p-1} = 1 \bmod p. \tag{89}$$

The cryptogram $C = P^e \bmod n$ is decrypted by $C^d \bmod n = P^{ed} \bmod n$ because $ed = 1 \bmod \phi$, implying the existence of an integer k such that $ed = k\phi + 1 = k(p-1)(q-1) + 1$. If $P \neq 0 \bmod p$, using equation (9.5) this implies

$$P^{ed} \bmod p = \left(P^{(p-1)}\right)^{k(q-1)} P \bmod p = P \bmod p. \tag{90}$$

The above equality holds trivially in the case $P = 0 \bmod p$. By identical arguments, $P^{ed} \bmod q = P \bmod q$. Since p and q are distinct primes, it follows that

$$P^{ed} \bmod n = P. \tag{91}$$

For example, let us suppose that Bob's public key is $\pi = (e, n) = (179, 571247)$.[6] He generated it following the prescription above choosing $p = 773$, $q = 739$ and $e = 179$. The private key d was obtained by solving $179d = 1 \bmod 772 \times 738$ using the extended Euclidean algorithm which yields $d = 515627$. Now if we want to send Bob encrypted "SHAKEN NOT STIRRED" we first use our digital alphabet to obtain the plaintext which can be written as the following sequence of six digit numbers

180700 100413 261314 192618 190817 170403

Then we encipher each block P_i by computing $C_i = P_i^e \bmod n$; *e.g.* the first block $P_1 = 180700$ will be eciphered as

$$P_1^e \bmod n = 180700^{179} \bmod 571247 = 141072 = C_1, \tag{92}$$

and the whole message is enciphered as:

141072 253510 459477 266170 286377 087175

The cryptogram C composed of blocks C_i can be send over to Bob. He can then decrypt each block using his private key $d = 515627$, *e.g.* the first block is decrypted as

$$141072^{515627} \bmod 571247 = 180700 = P_1. \tag{93}$$

In order to recover plaintext P from cryptogram C, an outsider, who knows C, n, and e, would have to solve the congruence

$$P^e \bmod n = C, \tag{94}$$

for example, in our case,

$$P_1^{179} \bmod 571247 = 141072. \tag{95}$$

Solving such an equation is believed to be a hard computational task for classical computers. So far, no classical algorithm has been found that computes the solution efficiently when n is a large integer (say 200 decimal digits long or more). However, if we know the prime decomposition of n it is a piece of cake to figure out the private key d: we simply follow the key generation procedure and solve the congruence $ed = 1 \bmod (p-1)(q-1)$. This can be done efficiently even when p and q are very large. Thus, in principle, anybody who knows n can find d by factoring n. The security of RSA therefore relies among others on the assumption that factoring large numbers is computationally difficult.

[6] Needless to say, number n in this example is too small to guarantee security, do not try this public key with Bob.

In the context of classical computation, such difficulty has never been proved. Worse still, we have seen in Section 8 that there is a quantum algorithm that factors large number efficiently. This means that the security of the RSA cryptosystem will be completely compromised if large-scale quantum computation becomes one day practical. This way, the advent of quantum computation rules out public cryptographic schemes commonly used today that are based on the "difficulty" of factoring or the "difficulty" of another mathematical operation called discrete logarithm [9].

On the other hand, quantum computation provides novel techniques to generate a shared private key with perfect confidentiality, regardless the computational power (classical or quantum) of the adversaries. Such techniques are referred to as *quantum key distribution* protocols and were proposed independently in the United States (S.Wiesner [35], C.H. Bennett and G. Brassard [36]) and in Europe (A. Ekert [37]). Discussion on quantum key distribution is outside the scope of this lecture.

10 Conditional quantum dynamics

Quantum gates and quantum networks provide a very convenient language for building any quantum computer or (which is basically the same) quantum multiparticle interferometer. But can we build quantum logic gates?

Single qubit quantum gates are regarded as relatively easy to implement. For example, a typical quantum optical realisation uses atoms as qubits and controls their states with laser light pulses of carefully selected frequency, intensity and duration; any prescribed superposition of two selected atomic states can be prepared this way.

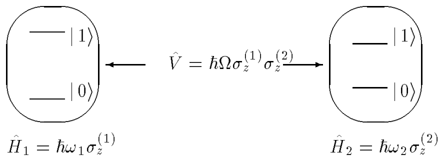
Two-qubit gates are much more difficult to build.

In order to implement two-qubit quantum logic gates it is sufficient, from the experimental point of view, to induce a conditional dynamics of physical bits, *i.e.* to perform a unitary transformation on one physical subsystem conditioned upon the quantum state of another subsystem,

$$U = |\,0\rangle\langle 0\,| \otimes U_0 + |\,1\rangle\langle 1\,| \otimes U_1 + \cdots + |\,k\rangle\langle k\,| \otimes U_k, \tag{96}$$

where the projectors refer to quantum states of the control subsystem and the unitary operations U_i are performed on the target subsystem [6]. The simplest non-trivial operation of this sort is probably a conditional phase shift such as $B(\phi)$ which we used to implement the quantum Fourier transform and the quantum controlled-NOT (or XOR) gate.

Let us illustrate the notion of the conditional quantum dynamics with a simple example. Consider two qubits, *e.g.* two spins, atoms, single-electron quantum dots, which are coupled via a $\sigma_z^{(1)}\sigma_z^{(2)}$ interaction (*e.g.* a dipole-dipole interaction):



The first qubit, with resonant frequency ω_1, will act as the control qubit and the second one, with resonant frequency ω_2, as the target qubit. Due to the coupling $\hat{V}$ the resonant frequency for transitions between the states $|0\rangle$ and $|1\rangle$ of one qubit *depends on the neighbour's state*. The resonant frequency for the first qubit becomes $\omega_1 \pm \Omega$ depending on whether the second qubit is in state $|0\rangle$ or $|1\rangle$. Similarly the second qubit's resonant frequency becomes $\omega_2 \pm \Omega$, depending on the state of the first qubit. Thus a π-pulse at frequency $\omega_2 + \Omega$ causes the transition $|0\rangle \leftrightarrow |1\rangle$ in the second qubit only if the first qubit is in $|1\rangle$ state. This way we can implement the quantum controlled-NOT gate.

11 Decoherence and recoherence

Thus in principle we know how to build a quantum computer; we can start with simple quantum logic gates and try to integrate them together into quantum networks. However, if we keep on putting quantum gates together into networks we will quickly run into some serious practical problems. The more interacting qubits are involved the harder it tends to be to engineer the interaction that would display the quantum interference. Apart from the technical difficulties of working at single-atom and single-photon scales, one of the most important problems is that of preventing the surrounding environment from learning about which computational path was taken in the multi-particle interferometer. This "welcher Weg" information can destroy the interference and the power of quantum computing.

Consider the following qubit-environment interaction, known as *decoherence*[38],

$$|0,m\rangle \mapsto |0,m_0\rangle, \qquad |1,m\rangle \mapsto |1,m_1\rangle, \tag{97}$$

where $|m\rangle$ is the initial state and $|m_0\rangle$, $|m_1\rangle$ are the two final states of the environment. This is basically a measurement performed by the environment on a qubit. Suppose that in our single qubit interference experiment (see Eqs. (43)) a qubit in between the two Hadamard transformation is "watched" by the environment which learns whether the qubit is in state $|0\rangle$ or $|1\rangle$. The evolution of the qubit and the environment after the first Hadamard and the phase gate is described by the following transformation,

$$|0\rangle|m\rangle \overset{H}{\mapsto} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|m\rangle \overset{\phi}{\mapsto} \frac{1}{\sqrt{2}}(e^{i\phi/2}|0\rangle + e^{-i\phi/2}|1\rangle)|m\rangle\,. \tag{98}$$

We write the decoherence action as

$$\frac{1}{\sqrt{2}}(e^{i\frac{\phi}{2}}|0\rangle + e^{-i\frac{\phi}{2}}|1\rangle)|m\rangle \mapsto \frac{1}{\sqrt{2}}(e^{i\frac{\phi}{2}}|0\rangle|m_0\rangle + e^{-i\frac{\phi}{2}}|1\rangle|m_1\rangle). \tag{99}$$

The final Hadamard gate generates the output state

$$\frac{1}{\sqrt{2}}(e^{i\frac{\phi}{2}}|0\rangle|m_0\rangle + e^{-i\frac{\phi}{2}}|1\rangle|m_1\rangle) \tag{100}$$

$$\overset{H}{\mapsto} \frac{1}{2}|0\rangle\left(e^{i\frac{\phi}{2}}|m_0\rangle + e^{-i\frac{\phi}{2}}|m_1\rangle\right) + \frac{1}{2}|1\rangle\left(e^{i\frac{\phi}{2}}|m_0\rangle - e^{-i\frac{\phi}{2}}|m_1\rangle\right). \tag{101}$$

Taking $|m_0\rangle$ and $|m_1\rangle$ to be normalised and $\langle m_0|m_1\rangle$ to be real we obtain the probabilities P_0 and P_1,

$$P_0 = \frac{1}{2}(1 + \langle m_0|m_1\rangle\cos\phi)\,, \tag{102}$$

$$P_1 = \frac{1}{2}(1 - \langle m_0|m_1\rangle\cos\phi)\,. \tag{103}$$

It is instructive to see the effect of decoherence on the qubit alone when its state is written in terms as a density operator. The decoherence interaction entangles qubits with the environment,

$$(\alpha|0\rangle + \beta|1\rangle)|m\rangle \mapsto \alpha|0\rangle|m_0\rangle + \beta|1\rangle|m_1\rangle. \tag{104}$$

Rewriting in terms of density operators and tracing over the environment's Hilbert space on the both sides, we obtain

$$\begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix} \mapsto \begin{pmatrix} |\alpha|^2 & \alpha\beta^*\langle m_0|m_1\rangle \\ \alpha^*\beta\langle m_1|m_0\rangle & |\beta|^2 \end{pmatrix}. \tag{105}$$

The off-diagonal elements, originally called by atomic physicists coherences, vanish as $\langle m_1|m_0\rangle \mapsto 0$, that is why this particular interaction with the environment is called decoherence.

How does decoherence affect, for example, Deutsch's algorithm? Substituting 0 or π for ϕ in Eq.(102) we see that we obtain the correct answer only with some probability, which is

$$\frac{1 + \langle m_0|m_1\rangle}{2}. \tag{106}$$

If $\langle m_0|m_1\rangle = 0$, the perfect decoherence case, then the network outputs 0 or 1 with equal probabilities, *i.e.* it is useless as a computing device. It is clear that we want to avoid decoherence, or at least diminish its impact on our computing device.

In general when we analyse *physically realisable* computations we have to consider errors which are due to the computer-environment coupling and from the computational complexity point of view we need to assess how these errors scale with the input size n. If the probability of an error in a single run, $\delta(n)$, grows exponentially with n, *i.e.* if $\delta(n) = 1 - A\exp(-\alpha n)$, where A and α are positive constants, then the randomised algorithm cannot technically be regarded as efficient any more regardless of how weak the coupling to the environment may be. Unfortunately, the computer-environment interaction leads to just such an unwelcome exponential increase of the error rate with the input size. To see this consider a register of size n and assume that each qubit decoheres separately,

$$\begin{aligned} |\,x\rangle\,|\,M\rangle &= |\,x_{n-1}\ldots x_1x_0\rangle\,|\,m\rangle\ldots|\,m\rangle\,|\,m\rangle \\ &\mapsto\ |\,x_{n-1}\ldots x_1x_0\rangle\,\big|\,m_{x_{n-1}}\big\rangle\ldots|\,m_{x_1}\rangle\,|\,m_{x_0}\rangle = |\,x\rangle\,|\,M_x\rangle\,, \end{aligned} \tag{107}$$

where $x_i \in \{0,1\}$. Then a superposition $\alpha\,|\,x\rangle + \beta\,|\,y\rangle$ evolves as

$$(\alpha\,|\,x\rangle + \beta\,|\,y\rangle)\,|\,M\rangle \mapsto \alpha\,|\,x\rangle\,|\,M_x\rangle + \beta\,|\,y\rangle\,|\,M_y\rangle\,, \tag{108}$$

but now the scalar product $\langle M_x\,|\,M_y\rangle$ which reduces the off-diagonal elements of the density operator of the whole register and which affects the probabilities in the interference experiment is given by

$$\langle M_x\,|\,M_y\rangle = \langle m_{x_0}\,|\,m_{y_0}\rangle\,\langle m_{x_1}\,|\,m_{y_1}\rangle\ldots\big\langle m_{x_{n-1}}\,\big|\,m_{y_{n-1}}\big\rangle \tag{109}$$

which is of the order of

$$\langle M_x\,|\,M_y\rangle = \langle m_0\,|\,m_1\rangle^{H(x,y)}, \tag{110}$$

where $H(x,y)$ is the Hamming distance between x and y, *i.e.* the number of binary places in which x and y differ (*e.g.* the Hamming distance between 101101 and 111101 is 1 because the two binary string differ only in the second binary place). Hence there are some coherences which disappear as $\langle m_0\,|\,m_1\rangle^n$ and therefore in some interference experiments the probability of error may grow exponentially with n.

It is clear that for quantum computation of any reasonable length to ever be physically feasible it will be necessary to incorporate some efficiently realisable stabilisation scheme to combat the effects of decoherence. Deutsch was the first one to discuss this problem. During the Rank Prize Funds Mini–Symposium on Quantum Communication and Cryptography, Broadway, England in 1993 he proposed 'recoherence' based on a symmetrisation procedure (for details see [39]). The basic idea is as follows. Suppose we have a quantum system, we prepare it in some initial state $|\,\Psi_i\rangle$ and we want to implement a prescribed unitary evolution $|\,\Psi(t)\rangle$ or just preserve $|\,\Psi_i\rangle$ for some period of time t. Now, suppose that instead of a single system we can prepare R copies of $|\,\Psi_i\rangle$ and subsequently we can project the state of the combined system into the symmetric subspace *i.e.* the subspace containing all states which are invariant under any permutation of the sub-systems. The claim is that frequent projections into the symmetric subspace will reduce errors induced by the environment.

The intuition behind this concept is based on the observation that a prescribed error-free storage or evolution of the R independent copies starts in the symmetric sub-space and should remain in that sub-space. Therefore, since the error-free component of any state always lies in the symmetric subspace, upon successful projection it will be unchanged and part of the error will have been removed. Note however that the projected state is generally not error–free since the symmetric subspace contains states which are not of the simple product form $|\,\Psi\rangle\,|\,\Psi\rangle\ldots|\,\Psi\rangle$. Nevertheless it has been shown that the error probability will be suppressed by a factor of $1/R$ [39].

More recently projections on symmetric subspaces were replaced by more complicated projections on carefully selected subspaces. These projections, proposed by Shor [40], Calderbank and Shor [41], Steane [42] and others [43, 44, 45, 46, 47], are constructed on the basis of classical error-correcting methods but represent intrinsically new quantum error-correction and stabilisation schemes; they are the subject of much current study.

Let us illustrate the main idea of recoherence by describing a simple method for protecting an unknown state of a single qubit in a noisy quantum register. Consider the following scenario: we want to store in a computer memory one qubit in an unknown quantum state of the form $|\,\phi\rangle = \alpha\,|\,0\rangle + \beta\,|\,1\rangle$ and we know that any single qubit which is stored in a register undergoes a decoherence type entanglement with an environment described by Eq.(104). To see how the state of the qubit is affected by the environment, we calculate the fidelity of the decohered state at time t with respect to the initial state $|\,\phi\rangle$

$$F(t) = \langle\phi\,|\,\rho(t)\,|\,\phi\rangle\,, \tag{111}$$

where $\rho(t)$ is given by Eq. (105). It follows that

$$F(t) = |\alpha|^4 + |\beta|^4 + 2|\alpha|^2|\beta|^2\,\mathrm{Re}[\langle m_0(t)\,|\,m_1(t)\rangle]\,. \tag{112}$$

The expression above depends on the initial state $|\,\phi\rangle$ and clearly indicates that some states are more vulnerable to decoherence than others. In order to get rid of this dependence we consider the average fidelity, calculated under the assumption that any initial state $|\,\phi\rangle$ is equally probable. Taking into account the normalisation constraint the average fidelity is given by

$$\bar{F}(t) = \int_0^1 F(t)\;d\,|\alpha|^2 = \frac{1}{3}(2 + \mathrm{Re}[\langle m_0(t)\,|\,m_1(t)\rangle])\,. \tag{113}$$

If we assume an exponential-type decoherence, where $\langle m_0(t)\,|\,m_1(t)\rangle = e^{-\gamma t}$, the average fidelity takes the simple form

$$\bar{F}(t) = \frac{1}{3}(2 + e^{-\gamma t})\,. \tag{114}$$

In particular, for times much shorter than the decoherence time $t_d = 1/\gamma$, the above fidelity can be approximated as

$$\bar{F}(t) \simeq 1 - \frac{1}{3}\gamma t + O(\gamma^2 t^2)\,. \tag{115}$$

Let us now show how to improve the average fidelity by quantum encoding. Before we place the qubit in the memory register we *encode* it: we can add two qubits, initially both in state $|0\rangle$, to the original qubit and then perform an encoding unitary transformation

$$\begin{aligned} |000\rangle &\mapsto |\bar{0}\bar{0}\bar{0}\rangle = (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle), & (116)\\ |100\rangle &\mapsto |\bar{1}\bar{1}\bar{1}\rangle = (|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle), & (117)\end{aligned}$$

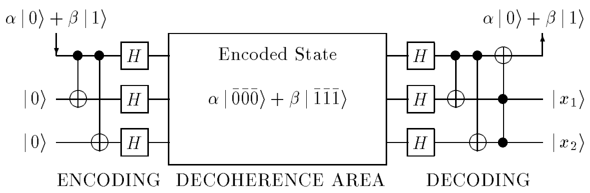
generating state $\alpha\,|\bar{0}\bar{0}\bar{0}\rangle + \beta\,|\bar{1}\bar{1}\bar{1}\rangle$, where $|\bar{0}\rangle = |0\rangle + |1\rangle$ and $|\bar{1}\rangle = |0\rangle - |1\rangle$. Now, suppose that only the second stored qubit was affected by decoherence and became entangled with the environment:

$$\begin{aligned} &\alpha(|0\rangle + |1\rangle)(|0\rangle|m_0\rangle + |1\rangle|m_1\rangle)(|0\rangle + |1\rangle) + \\ &\beta(|0\rangle - |1\rangle)(|0\rangle|m_0\rangle - |1\rangle|m_1\rangle)(|0\rangle - |1\rangle), \end{aligned} \quad (118)$$

which can also be written as

$$(\alpha\,|\bar{0}\bar{0}\bar{0}\rangle + \beta\,|\bar{1}\bar{1}\bar{1}\rangle)(|m_0\rangle + |m_1\rangle) + (\alpha\,|\bar{0}\bar{1}\bar{0}\rangle + \beta\,|\bar{1}\bar{0}\bar{1}\rangle)(|m_0\rangle - |m_1\rangle). \quad (119)$$

The decoding unitary transformation can be constructed using a couple of quantum controlled-NOT gates and the Toffoli gate, thus completing the error-correcting network:



Careful inspection of the network shows that any single phase-flip $|\bar{0}\rangle \leftrightarrow |\bar{1}\rangle$ will be corrected and the environment will be effectively disentangled from the qubits. In our particular case we obtain

$$(\alpha\,|0\rangle + \beta\,|1\rangle)\,[|00\rangle\,(|m_0\rangle + |m_1\rangle) + |10\rangle\,(|m_0\rangle - |m_1\rangle)]. \quad (120)$$

The two auxiliary outputs carry information about the error syndrome - 00 means no error, 01 means the phase-flip occurred in the third qubit, 10 means the phase-flip in the second qubit and 11 signals the phase flip in the first qubit.

Thus if only one qubit in the encoded triplet decoheres we can recover the original state perfectly. In reality all three qubits decohere simultaneously and, as the result, only partial recovery of the original state is possible. In this case lengthy but straightforward calculations show that the average fidelity of

the reconstructed state after the decoding operation for an exponential-type decoherence is

$$\bar{F}_{\rm ec}(t) = \frac{1}{6}[4 + 3e^{-\gamma t} - e^{-3\gamma t}]\,. \quad (121)$$

For short times this can be written as

$$\bar{F}_{\rm ec}(t) \simeq 1 - \frac{1}{2}\gamma^2 t^2 + O(\gamma^3 t^3). \quad (122)$$

Comparing Eq. (114) with Eq. (121), we can easily see that for all times t,

$$\bar{F}_{\rm ec}(t) \geq \bar{F}(t). \quad (123)$$

This is the essence of recoherence via encoding and decoding. There is much more to say (and write) about quantum codes and the reader should be warned that we have barely scratched the surface of the current activities in quantum error correction, neglecting topics such as group theoretical ways of constructing good quantum codes [45, 46], concatenated codes [47], quantum fault tolerant computation [48] and many others.

12 Concluding remarks

Research in quantum computation and in its all possible variations has become vigorously active and any comprehensive review of the field must be obsolete as soon as it is written. Here we have decided to provide only some very basic knowledge, hoping that this will serve as a good starting point to enter the field. Many interesting papers in these and many related areas can be found at the Los Alamos National Laboratory e-print archive (http://xxx.lanl.gov/archive/quant-ph) and on the web site of the Center for Quantum Computation (www.qubit.org).

13 Acknowledgments

This work was supported in part by the European TMR Research Network ERP-4061PL95-1412, The Royal Society London, and Elsag plc. PH acknowledges the support of the Rhodes Trust.

References

[1] The term was coined by B. Schumacher. See, for example, *Phys. Rev. A* **51** 2738 (1995).

[2] D. Deutsch, *Proc. R. Soc. Lond. A* **425** 73 (1989).

[3] W. K. Wootters and W. H. Zurek, *Nature* **299** 802 (1982).

[4] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. Smolin and H.Weinfurter, *Phys. Rev. A* **52** 3457 (1995).

[5] D. Deutsch, A. Barenco and A. Ekert, *Proc. R. Soc. Lond. A* **449** 669 (1995).

[6] A. Barenco, D. Deutsch, A. Ekert and R. Jozsa, *Phys. Rev. Lett.* **74** 4083 (1995).

[7] D. P. DiVincenzo, *Phys. Rev. A* **51** 1015 (1995).

[8] S. Lloyd *Phys. Rev. Lett.* **75** 346 (1995).

[9] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers* (Oxford University Press, Oxford, 1979).

[10] T. Toffoli, *Mathematical Systems Theory* **14** 13 (1981).

[11] C. H. Papadimitriou, *Computational Complexity* (Addison-Wesley, 1994).

[12] D. Coppersmith, *IBM Research report* (1994).

[13] R. Solovay and V. Strassen *SIAM J. Comp.* **6** 84 (1977).

[14] R. Motwani and P. Raghavan, *Randomised Algorithms* (Cambridge University Press, 1995).

[15] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring" *Proc. 35th Annual Symposium on the Foundations of Computer Science*, p. 124 Edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos, CA 1994). Expanded version of this paper is available at LANL quant-ph archive.

[16] R. P. Feynman, *International Journal of Theoretical Physics* **21** 467 (1982).

[17] R. Cleve, A. Ekert, C. Macchiavello and M. Mosca, *Proc. R. Soc. Lond. A* **454** 339 (1998).

[18] D. Deutsch, *Proc. R. Soc. Lond. A* **400** 97 (1985).

[19] L. K. Grover, "A fast quantum mechanical algorithm for database search", *Proc. 28th Annual ACM Symposium on the Theory of Computing (STOC'96)* p. 212 (ACM, Philadelphia, Pennsylvania, 1996).

[20] M. Boyer, G. Brassard, P. Hoyer, A. Tapp, *Proc. of the Workshop on Physics and Computation (PhysComp96)* 36 (1996).

[21] T. Hogg, *Physica* **D120** 102 (1998).

[22] C. Zalka, *Physical Review* **A60** 2746 (1999).

[23] A. Y. Kitaev, LANL quant-ph archive, quant-ph/9511026 (1995).

[24] V. Vedral, A. Barenco and A. Ekert, *Phys. Rev. A* **54** 147 (1996).

[25] D. Beckman, A. Chari, S. Devabhaktuni and J. Preskill, *Phys. Rev. A* **54** 1034 (1996).

[26] A. Ekert and R. Jozsa, *Rev. Mod. Phys.* **68** 733 (1996).

[27] D. Kahn, *The Codebreakers: The Story of Secret Writing*, (Macmillan, New York,1967).

[28] D. Stinson, *Cryptography: Theory and Practice* (CRC Press, 1995).

[29] G. S. Vernam, *J. AIEE* **45** 109 (1926).

[30] C. E. Shannon, *Bell Syst. Tech. J.* **28** 657 (1949).

[31] W. Diffie and M. E. Hellman, *IEEE Transactions on Information Theory* **22** 644 (1976).

[32] R. L. Rivest, A. Shamir and L. M. Adleman, *Communication of the ACM* **21** 120 (1978).

[33] J. H. Ellis, *Tech. report* Communications-Electronics Security Group, United Kingdom (1970).

[34] C. Cocks, *Tech. report* Communications-Electronics Security Group, United Kingdom (1973).

[35] S. Wiesner, *Sigact News* **15**, 78 (1983).

[36] C. H. Bennett and G. Brassard, *Proc. IEEE Int. Conference on Computers, Systems and Signal Processing* (IEEE, New York, 1984).

[37] A. Ekert, *Phys. Rev. Lett.* **67**, 661 (1991).

[38] W. H. Zurek, *Phys. Today* **44** October p.36 (1991).

[39] A. Berthiaume, D. Deutsch and R. Jozsa, Proceedings of the Workshop on the Physics and Computation—PhysComp '94, IEEE Computer Society Press, Dallas, Texas (1994); A. Barenco, A. Berthiaume, D. Deutsch, A. Ekert, R. Jozsa and C. Macchiavello, SIAM J. Comput. **26**, 1541 (1997).

[40] P. W. Shor, *Phys. Rev. A* **52**, R2493 (1995).

[41] R. Calderbank and P. W. Shor, *Phys. Rev. A* **54**, 1098 (1996).

[42] A. Steane, *Phys. Rev. Lett.* **77**, 793 (1996); A. Steane, *Proc. R. Soc. Lond. A* **452**, 2551 (1996).

[43] A. Ekert and C. Macchiavello, Phys. Rev. Lett. **77**, 2585 (1996).

[44] R. Laflamme, C. Miquel, J.P. Paz and W.H. Zurek, Phys. Rev. Lett. **77**, 198 (1996).

[45] D. Gottesman, Phys. Rev. A **54**, 1862 (1996).

[46] A.R. Calderbank, E.M. Rains, P.W. Shor and N.J.A. Sloane, Phys. Rev. Lett. **78**, 405 (1997).

[47] E. Knill and R. Laflamme, e-print quant-ph/9608012 (1996).

[48] P.W. Shor, e-print quant-ph/9605011 (1996); D.P. DiVincenzo and P.W. Shor, Phys. Rev. Lett. **77**, 3260 (1996).

[49] R.Solovay, "Lie groups and quantum circuits", preprint 1999.