Quantum computation and Shor's factoring algorithm

Artur Ekert and Richard Jozsa

Clarendon Laboratory, University of Oxford Oxford OX1 3PU, United Kingdom and School of Mathematics and Statistics, University of Plymouth, Plymouth, Devon PL4 8AA, United Kingdom

Current technology is beginning to allow us to manipulate rather than just observe individual quantum phenomena. This opens up the possibility of exploiting quantum effects to perform computations beyond the scope of any classical computer. Recently Peter Shor discovered an efficient algorithm for factoring whole numbers, which uses characteristically quantum effects. The algorithm illustrates the potential power of quantum computation, as there is no known efficient classical method for solving this problem. The authors give an exposition of Shor's algorithm together with an introduction to quantum computation and complexity theory. They discuss experiments that may contribute to its practical implementation. [S0034-6861(96)00303-0]

CONTENTS

I. INTRODUCTION

The concepts of information and computation can be properly formulated only in the context of a physical theory—information is stored, transmitted, and processed always by *physical* means. When quantum effects become important, for example, at the level of single atoms and photons, the existing, purely abstract, classical theory of computation becomes fundamentally inadequate. Entirely new modes of computation and information processing become possible. Phenomena such as quantum interference and quantum entanglement can be exploited for computations. Quantum computers can accept input states that represent a coherent superposition of many different possible inputs and subsequently evolve them into a corresponding superposition of outputs. Computation, i.e., a sequence of unitary transformations, simultaneously affects each element of the superposition, generating a massive parallel data processing, albeit within one piece of quantum hardware. Consequently quantum computers can efficiently solve some problems that are believed to be intractable on any classical computer. One problem of this type is factorization. In the following we describe Shor's quantum algorithm for factorizing numbers and use it to illustrate both the potential power of quantum computation and the difficulties involved in the experimental realizations.

In Sec. II we introduce the Turing-machine model for classical computation and describe its quantum generalization. The notion of *efficient* computation, which underlies the possible benefits of quantum computation, is explained in Sec. III. In Sec. IV we describe an alternative model of computation—quantum networks built out of quantum logic gates. This is particularly significant since elementary quantum logic gates are within the scope of currently proposed experimental realizations. Section V introduces the quantum discrete Fourier transform and its implementation as a quantum network. This transform is a fundamental ingredient in Shor's algorithm. Further mathematical preliminaries are described in Sec. VI. The algorithm itself is set out in Sec. VII. The remaining sections discuss issues related to experimental implementations of quantum computation. We describe the destabilizing effects of environmental interaction, which is a major experimental (and theoretical) obstacle. Finally we outline various proposed experimental realizations of quantum logic gates, including cavity quantum electrodynamics, quantum dot arrays, and the selective excitation of trapped ions.

At the present time it is not clear whether it will be practical to build physical devices that can perform coherent quantum computations. This notwithstanding, the theoretical study of quantum physics from the point of view of computational complexity may at least be expected to shed new light on the foundations of quantum theory. The theory of computational complexity will also need to be modified to account for the new quantum modes of computation, and indeed the study of the optimal use of general physical resources in the process of computation may eventually become a branch of physics itself. On the experimental side the current challenge is not to build a full quantum computer right away, but rather to move from the experiments in which we merely observe quantum interference and entanglement to experiments in which we can *control* these quantum phenomena.

II. COMPUTATION—BASIC IDEAS

Computation may be defined as the systematic creation of symbols (the "output") which, under a given method of interpretation, have abstract properties that were specified in other symbols (the "input"). "Symbols" here are physical objects, and computation is a physical process performed by a physical device called a computer. Indeed, given a rule for interpreting physical states as symbols, *any* physical process may be thought of as a kind of computation. Quantum computers are machines that rely on characteristically quantum phenomena, such as quantum interference and quantum entanglement, in order to perform computation. The classical theory of computation usually does not refer to physics. Pioneers such as Turing, Church, Post, and Gödel managed to capture the correct classical theory by intuition alone. As a result it is often falsely assumed that the foundations of computation theory are self-evident and purely abstract. The fundamental connection between the laws of physics and what is computable was emphasized by Feynman (1982) and Deutsch (1985) [see also Landauer (1987, 1991) and Lloyd (1993a, 1994)].

The classical theory of computation is essentially the theory of the universal Turing machine—the most popular mathematical model of classical computation. Its significance relies on the fact that, given a large but finite amount of time, the universal Turing machine is capable of any computation that can be done by any modern classical digital computer, no matter how powerful. In this section we provide a brief description of Turing machines and show how the concept of Turing machines may be modified to incorporate nonclassical computation. This leads to a model of quantum computation first described by Deutsch (1985) and developed by Bernstein and Vazirani (1993)—the so-called quantum Turing machine [see also Benioff (1980, 1986 and references therein), who described a classical Turing machine made of quantum components].

A Turing machine $\mathcal{M}$ (Welsh, 1988; Penrose,1989; Papadimitriou, 1994) can be visualized as a device composed of a processing unit in the form of a write/read head and a memory with unlimited storage capacity in the form of an infinite tape (as shown in Fig. 1).

The tape is divided into discrete cells. Each cell can have a symbol from a finite alphabet Σ written on it. The tape is scanned, one cell at a time, by the read/write head. The head can be in one of a finite set of states Q, where $Q=\{q_0,q_1,\ldots,q_h\}$. The machine action is made up entirely of discrete steps, and each step is determined by two initial conditions: the current state of the head and the symbol that occupies the tape cell currently being scanned. Given a pair of initial conditions,

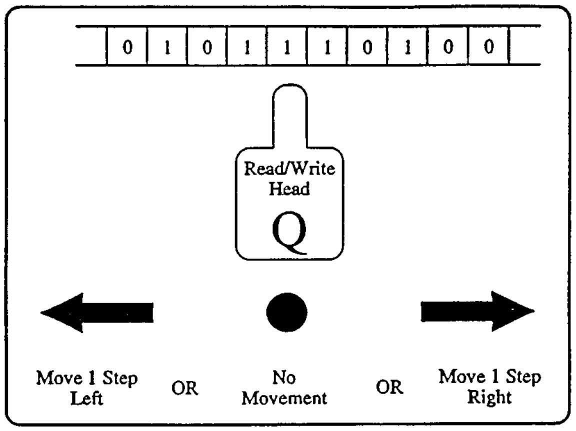

FIG. 1. A schematic picture of a Turing machine.

the machine receives a three-part instruction for its next step. The first part of the instruction specifies the next state of the head. The second part designates the symbol the head is to write into the scanned cell. The third part determines whether the head is to move one cell to the left or to the right along the tape or to stay in its present position. The list of all possible instructions corresponding to all initial conditions of $\mathcal{M}$ is finite (because both the set of symbols, i.e., the alphabet Σ and the set of internal states of the machine Q are finite) and provides a complete description of the action of $\mathcal{M}$. Such a description can be written as a *finite* set of quintuples of the form

$$(q,s,q',s',d). \tag{1}$$

The first two characters describe the initial condition and the remaining three characters describe the instruction. Reading from the left: q is the current state of the head, s is the symbol currently under the read/write head, q' is the state that the head is to enter next, s' is the symbol to be written in place of s, and d is the direction in which the head is to move (or stay fixed) relative to the tape. It is customary to single out one of the head states as a "halting state" q_h. It has the property that all instructions of the form (q_h,s,q',s',d) have $s'=s$, $q'=q_h$, and $d=$ "no movement." Thus, once the head enters state q_h, no further change occurs—the computation halts.

In this language a computation on the machine consists of presenting it with an input, which is a *finite* string of symbols from the alphabet Σ written in the tape cells (all remaining cells containing some chosen standard symbol of Σ), then allowing the machine to start in the initial state q_0 with the head scanning the leftmost symbol of the input and to proceed with its basic operations [Eq. (1)] until it stops in its final (halting) state q_h. (In some cases the computation might not terminate.) The output of the computation is defined as the contents of some chosen part of the tape when (and if) the machine reaches its halting state.

In the process of computation the machine goes through a sequence of configurations; each configuration provides a global description of the machine and is determined by the string written on the entire tape, the state of the head, and the position of the head. For example, the initial configuration is given by the input string, q_0 and the head scanning the leftmost symbol from the input. There are infinitely many possible configurations of the machine, but in all successful computations the machine goes through a finite sequence of configurations. The transition between configurations is completely determined by the rules in Eq. (1). Note that the operation of the machine is "local" in the sense that the transition between configurations depends only on the currently scanned tape symbol (and the head state) rather than on the whole global configuration.

A computation is said to be reversible if the transition between configurations is reversible, i.e., the $(n+1)$th configuration uniquely determines the nth one. (Note, however, that the time reverse of a Turing machine is not a Turing machine, since, for example, the time reverse of "head writing and then moving" is "head moving and then writing," which is not a valid Turing machine action.) Thus the physical action corresponding to a reversible computation is thermodynamically reversible and hence involves zero energy dissipation (Landauer, 1961; Bennett, 1973, 1982). A fundamental result of Bennett (1973, 1982, 1989) [see also Lecerf (1963)] states that, given any Turing machine $\mathcal{M}$, there is a reversible Turing machine $\mathcal{M}$' which performs the same computation [and furthermore $\mathcal{M}$' is an efficient computation (see next section) if $\mathcal{M}$ is efficient]. Hence demanding that all computation be reversible imposes no restriction at all on computing power. This is an important observation since the model of quantum computation introduced below is automatically reversible. Bennett's result also shows that the performance of any computation does *not* require any necessary intrinsic energy dissipation.

Remark. The concept of halting state needs to be reconsidered in the context of reversible computation, since these computations cannot halt, i.e., cannot have two successive configurations equal. We may either run the computation for some predetermined number of steps (being given, for example, by a prescribed function of the input size) or alternatively we may designate one of the tape squares as an "end of computation" flag. When the machine has completed the computation and written the answer on some prescribed part of the tape, the flag is set to some prescribed letter of Σ, and the machine simply continues operating in a reversible but trivial way which does not alter the part of the tape containing the answer. A periodic observation of the flag will enable us to determine when the answer is ready for reading out.

Computations do not have to be deterministic. Indeed, we can allow a Turing machine "to toss an unbiased coin," and to choose its steps randomly. Such a probabilistic computation can be viewed as a directed, treelike graph, where each node corresponds to a configuration of the machine and each edge represents one step of the computation (see Fig. 2). The computation starts from the root node representing the initial configuration and it subsequently branches into other nodes representing configurations reachable with nonzero probability from the initial configuration. The action of the machine is completely specified by a finite description of the form

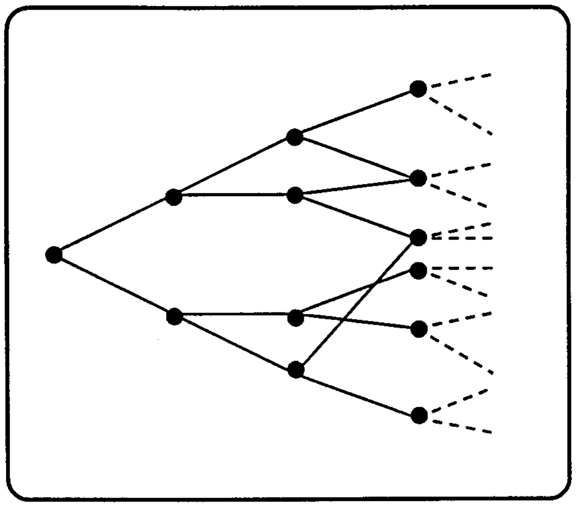
FIG. 2. Branching tree of possible configurations in a probabilistic computation.

$$\delta: Q\times\Sigma\times Q\times\Sigma\times\{\text{Left, Right, No movement}\}\mapsto[0,1], \tag{2}$$

where $\delta(q,s,q',s',d)$ gives the probability that, if the machine is in state q reading symbol s, it will enter state q' , write s', and move in direction d. This description must conform to the laws of probability, as applied to the computation tree. Associating each edge in the graph, with the probability that the computation follows that edge, yields the following requirements: (1) We must require that the sum of the probabilities on edges leaving any single node is always equal to 1. (2) The probability of a particular path being followed from the root to a given node is the product of the probabilities along the path's edges. (3) The probability of a particular configuration's being reached after n steps is equal to the sum of the probabilities along all paths which in n steps connect the initial configuration with that particular configuration. Such randomized algorithms can solve some problems (with probability arbitrarily close to 1) much faster than any known deterministic algorithms. [For examples, see Welsh (1988), pages 150–155].

The classical model described above suggests a natural quantum generalization. The various head states, and tape symbols correspond to orthogonal quantum states and the head position corresponds to a quantum observable with integer spectrum. The quantum computation can be represented by a graph similar to that of a probabilistic computation. Following the rules of quantum dynamics, we associate with each edge in the graph the probability *amplitude* that the computation follows that edge. As before, the probability amplitude of a particular path's being followed is the product of the probability amplitudes along the path's edges. The probability amplitude of a particular configuration is the sum of the amplitudes along all possible paths leading from the root to that configuration. Probability amplitudes are complex numbers (of modulus not greater than unity); the corresponding probability is obtained by taking a modulus squared of the probability amplitude. This way of calculating probabilities gives quantum computation the novel nonclassical feature of quantum interference. For example, if a particular final configuration can be reached via two different paths with amplitudes α and $-\alpha$, then the probability of reaching that configuration is $|\alpha-\alpha|^2=0$, despite the fact that the probability for the computation to follow either of the two paths separately is $|\alpha|^2$ in both cases. Furthermore, a single quantum computer can follow many distinct computational paths in superposition and produce a final output depending on the interference of all of them. This is in contrast to a classical probabilistic Turing machine, which follows only some *single* (randomly chosen) path. This feature has been called "computation by quantum parallelism" (Deutsch, 1985; Jozsa, 1991, 1992) and underlies all known quantum algorithms (Deutsch and Jozsa, 1992; Bernstein and Vazirani, 1993; Berthiaume and Brassard, 1994; Shor, 1994; Simon, 1994).

The action of any such quantum machine is completely specified by a finite description of the form

$$\delta{:}Q\times\Sigma\times Q\times\Sigma\times\{\text{Left, Right, No movement}\}\mapsto\mathbf{C}, \tag{3}$$

where $\delta(q,s,q',s',d)$ gives the probability *amplitude* that, if the machine is in state q reading symbol s, it will enter state q', write s', and move in direction d. The amplitudes in Eq. (3) [like the probabilities in Eq. (2)] cannot be specified arbitrarily. They are constrained by the condition that the induced transition on general global configurations be *unitary*. This condition is analyzed by Bernstein and Vazirani (1993). This is the most studied model of quantum computation and is called the quantum Turing machine (QTM) (Deutsch, 1985; Bernstein and Vazirani, 1993). It is a direct quantum generalization of Turing's classical definition.

The QTM model is based on the idea that each computational step is the result of a fixed unitary operation, induced by Eq. (3), acting on the current global configuration of the whole machine. This unitary operation is required to be "local" in the sense that it allows transitions only between suitably neighboring configurations, e.g., with head positions shifted by at most one and the entire tape contents identical except for the currently scanned cell. One may entertain other possible models, e.g., using a *Hamiltonian* that is local in a similar sense and evolving it for a fixed time to generate each computational step. Indeed it has been suggested that a local Hamiltonian is physically more realistic than a local unitary evolution (Feynman, 1985; Margolus, 1986, 1991). The relationship between the computing powers of these two models is not known.

A QTM cannot perform any computation beyond those that can be done on classical Turing machines. Given an input, one may simply program a classical computer to compute iterations of the rules given by Eq. (3), calculating the full (finite) list of configuration amplitudes at each step. The final probability distribution is then simulated classically by "tossing coins." However this classical simulation of a QTM is inefficient (i.e., involves an exponential slowdown), since the number of amplitudes generally grows exponentially with the number of computational steps. Thus quantum computers cannot exceed classical computers in *what* may be computed, but may possibly exceed them in the *efficiency* of computation. The concept of efficient (or "polynomial-time") computation is fundamental in all that follows and is explained in the next section.

In the classical theory of computation many other models of the notion of "mechanical computation" have been studied, e.g., network (or circuit) models, cellular automata, etc. However, these models can all be proved equivalent in the following sense: any computation performable within any of the models may be simulated on a Turing machine. Furthermore this simulation preserves *efficiency* of computation (see next section). It is generally asserted that *any* conceivable, "reasonable" model of computation has this property. This is the so-called (classical) Church-Turing thesis (in its modern form incorporating efficient computability rather than just computability). No counterexample to this thesis has ever been found. In the quantum context one may similarly consider other models of quantum computation, e.g., quantum cellular automata, quantum networks, etc. Quantum networks have been considered by Deutsch (1989) and Yao (1993), and a form of quantum cellular automaton by Margolus (1991). Yao (1993) has shown that the quantum network model is computationally equivalent to the QTM model. The quantum network model is especially useful in that it relates far more directly to proposed experimental realizations of quantum computation than does the QTM model. In Sec. IV below we shall give an account of the essential ingredients of the quantum network model of computation. It remains an open question whether further possible models of quantum computation are also equivalent in the sense of a "quantum Church-Turing thesis."

III. SLOW AND FAST ALGORITHMS—EFFICIENT COMPUTATION

In order to solve a particular problem, a computer follows 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. Some algorithms are fast (e.g., multiplication); others are very slow (e.g., factorization, playing chess). Consider, for example, the following factoring problem:

$$\square\times\square=29083. \tag{4}$$

Using only paper and pencil it would probably take about an hour to find the two whole numbers different from 1 which should be written in the two boxes (the solution is unique). Solving the reverse problem

$$127\times 229=\square, \tag{5}$$

again using paper and pencil, takes less than a minute. This is because we know fast algorithms for multiplication, but we do not know fast algorithms for factoring.

There is a rigorous way of defining what makes an algorithm fast (tractable, usable) or slow (intractable, unusable) (Welsh, 1988; Papadimitriou, 1994). We emphasize that the algorithm here is viewed as a prescription for solving a whole *class* of problems, e.g., multiplying *any* two given numbers, rather than just a single instance, e.g., multiplying two particular numbers. The crucial question then is, how does the length of the computation increase with increasing size of the input? An algorithm is said to be fast or efficient if the time 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), and we measure the execution time as the number of computational steps. Thus an efficient algorithm on a general input N runs in $\mathrm{poly}(\log N)$ time, i.e.,

$$\#\text{ steps for input } N \leqslant p(\log N) \text{ for all } N, \tag{6}$$

where p is a fixed polynomial. Clearly, in assessing efficiency, we may think of any computation already known to be efficient as a single computational step, since sums, products, and composites of polynomials are still polynomials.

Example. Consider the most naive factoring method: try dividing N by each number from 1 to $\sqrt{N}$ (as any composite N must have a factor in this range). This requires at least $\sqrt{N}$ steps (at least one step for each tried factor). However $\sqrt{N}=2^{1/2\log N}$ is *exponential* in $\log N$, so this is not an efficient algorithm.

We now turn to randomized algorithms. Consider an algorithm $\mathcal{A}$ that runs successfully only with probability $1-\epsilon$, and we know when it is successful. (For example, $\mathcal{A}$ may produce a candidate factor m of the input N, followed by a trial division to check whether m really is a factor or not.) Here $\epsilon>0$ is independent of the input N. By repeating $\mathcal{A}$ k times, we get an algorithm $\mathcal{A}_k$, which will be successful with probability $1-\epsilon^k$ (i.e., having at least one success). This can be made arbitrarily close to 1 by choosing a fixed k sufficiently large. Furthermore, if $\mathcal{A}$ is efficient, then $\mathcal{A}_k$ will also be efficient, since k is independent of N. Thus the success probability of any efficient randomized algorithm of this type may be amplified arbitrarily close to 1 while retaining efficiency. Indeed we may even let the success probability $1-\epsilon$ decrease with N as $1/\mathrm{poly}(\log N)$ and k increase as $\mathrm{poly}(\log N)$ and still retain efficiency while amplifying the success probability as close to 1 as desired. Shor's quantum factoring algorithm will be of this type, based on an efficient algorithm that provides a factor of the input N with probability that decreases as $1/\mathrm{poly}(\log N)$.

A more fundamental class of randomized algorithms—known in the literature as BPP (bounded error, probabilistic, polynomial time algorithms) [see Papadimitriou (1994)]—concerns decision problems for which the output is just "yes" or "no." For example, given N and $m<N$, is there a factor of N less than m? A BPP algorithm $\mathcal{A}$ is an efficient algorithm providing an answer which, for any input, is correct with a probability greater than some constant $\delta>1/2$. However, in this case we do not know if the answer is correct or not. As above we may repeat $\mathcal{A}$ some fixed number k times and then take a majority vote of all the k answers. For sufficiently large k the majority answer will be correct with probability as close to 1 as desired [in fact the probability even converges to 1 *exponentially* fast in k (Papadimitriou, 1994)]. In computational complexity theory it is customary to view problems in BPP as being "tractable" or "solvable in practice" and problems not in BPP as "intractable" or "unsolvable in practice on a computer."

Example. There exist several randomized algorithms that test for primality. For example, the Rabin randomized algorithm tests whether a given number N is prime or composite (Rabin, 1980). If the test shows N to be composite, then it is composite; if the test shows N to be prime, then it is prime with the probability $1-0.25$, which does not depend on the size of the number. The algorithm is efficient, so we can repeat the computation several times to reach any required degree of confidence. However, the algorithm does not display an actual factor of a composite number.

There is no known efficient classical algorithm for factoring, even if we allow it to be probabilistic in the above senses. Currently the best classical factoring algorithms are the Multiple Polynomial Quadratic Sieve (Silverman, 1987), for numbers less than 120 decimal digits long, and the Number Field Sieve (Lenstra *et al.*, 1990), particularly good for numbers more than 110 decimal digits long. Still, even the fastest algorithms run in time of order $\exp[(\log N)^{1/3}(\log\log N)^{2/3}]$ and would need a couple of billion years to factorize a 200-digit number. It is not known whether a fast classical algorithm for factorization exists or not—none has yet been found.

Remark. There is much interest in an efficient factoring algorithm, because the problem of factoring underpins the security of many classical public-key cryptosystems. For example, RSA—the most popular public-key cryptosystem, named after the three inventors: Ron Rivest, Adi Shamir, and Leonard Adleman (1979)—relies for its security on the presumed intractability of factoring large numbers.

Feynman (1982) observed that it appears impossible to simulate a general quantum evolution on a classical probabilistic computer in an *efficient* way. This is a profound feature distinguishing classical from quantum physics. It provides an entirely novel avenue for contrasting classical and quantum theories. For example, in a QTM the number of tape cells (two-state quantum systems) visited generally increases linearly with the number of computational steps n, so that the dimension of the affected Hilbert space increases exponentially with n. Thus the "obvious" classical simulation—just computing and storing all the amplitudes using the transition rules of the QTM—involves an exponential slowdown. Note that a similar situation occurs for a *classical* probabilistic Turing machine: the number of possible configurations reached in the branching tree of Fig. 2 generally also increases exponentially with n. Thus computing the whole probability distribution at each step again involves an exponential slowdown, but here we may actually *simulate* the probability distribution by "tossing coins" and keeping only the *single* probabilistic outcome at each step, i.e., traversing a single path through the tree of branching possibilities, respecting the correct probability distribution in the choice at each step. By contrast, in the QTM case, because of the phenomenon of *quantum interference* of different paths, we cannot traverse a single path through the tree of branching amplitudes to simulate the final probability distribution. Apparently, according to quantum physics, Nature is able to efficiently keep track of exponentially many branching amplitudes in a way that we cannot simulate classically! This is precisely the phenomenon of "computation by quantum parallelism" mentioned previously. Thus we may suspect that the computing power of a quantum device can exceed that of any classical device, not in *what* may be computed, but in *efficiency* of computation. This provides a fundamental impetus for the study of quantum computation and its possible experimental realization.

IV. BUILDING UNITARY TRANSFORMATIONS—QUANTUM NETWORKS

Instead of working from the basic definition of a QTM, it will be sufficient here to adopt a (somewhat less rigorous) higher-level description of a quantum computer. We envisage a familiar programmable computer $\mathcal{C}$ in which each memory cell is a qubit (Schumacher 1995), i.e., a two-state quantum system capable of supporting coherent superpositions. There is a chosen basis ("computational" basis) $|0\rangle,|1\rangle$ for each cell, corresponding to the classical bit values 0 and 1. The programming language for $\mathcal{C}$ includes a finite set of instructions for applying some chosen set of unitary transformations to any prescribed cell. It is fundamental to the notion of "mechanical computation" that programming and computation occur by "finite means." We cannot just assume that any unitary transformation U may be efficiently implemented—U must be constructed (i.e., programmed) using some *finite* basic set of transformations. Since there is a continuum of possible unitary transformations, we shall be able to program them only approximately in general, but to any desired degree of approximation, using sufficiently long programs.

The set of basic unitary transformations is chosen using the following result.

Theorem. Consider

$$V_0=\begin{pmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{pmatrix},\quad V_1=\begin{pmatrix}\cos\alpha & i\sin\alpha\\ i\sin\alpha & \cos\alpha\end{pmatrix},$$

$$V_2=\begin{pmatrix}e^{i\alpha} & 0\\ 0 & 1\end{pmatrix},\qquad V_3=\begin{pmatrix}1 & 0\\ 0 & e^{i\alpha}\end{pmatrix} \tag{7}$$

and their respective inverses V_4,V_5,V_6,V_7.

For α an irrational multiple of π, these eight transformations generate under composition a group dense in all two-dimensional unitary transformations. For almost all such α any given unitary transformation U may be approximated "efficiently" in the following sense: Given any $\epsilon>0$, U may be approximated to within ϵ by a concatenation of V_i's of length $\leqslant \mathrm{poly}(1/\epsilon)$.

The latter property may be intuitively understood as follows. Consider, for example, real rotations in the plane, and let R_α be any fixed irrational rotation. If we accept that the first k powers of R_α are essentially randomly distributed around the circle, then, since the circle has finite size, by doubling k, we get twice as many points around the circle. We can thus approximate any given rotation twice as well. For example, for accuracy ϵ we should require $O(1/\epsilon)$ applications of R_α, a linear polynomial in $1/\epsilon$. A similar argument may be expected to apply to any *compact* group like $U(2)$.

Remark. The above choice of eight transformations was originally given by Deutsch (1985). It may be considerably simplified. The construction of a universal quantum Turing machine by Bernstein and Vazirani (1993) implies that just one irrational rotation (together with the simple transformation defined by $|0\rangle\mapsto|0\rangle$ and $|1\rangle\mapsto-|1\rangle$) suffices.

Thus we endow $\mathcal{C}$ with eight extra instructions for applying $V_0,\ldots,V_7$ to any chosen cell. Then, for any given two-dimensional unitary transformation U, there exists a program of length $\mathrm{poly}(1/\epsilon)$ approximating U with an accuracy ϵ.

For general d-dimensional transformations we have the following Theorem:

Theorem (Deutsch 1985). Let U be any d-dimensional unitary matrix. Then U may be written as a product of $2d^2-d$ unitary matrices (i.e., polynomially many in d), each of which acts only within a two-dimensional subspace spanned by a pair of computational basis states.

Proof. We first note that, given an arbitrary vector with components $(a_1,\ldots,a_d)$ in the computational basis $|e_1\rangle,\ldots,|e_d\rangle$, we may transform it to $(1,0,\ldots,0)$ using a sequence of $d-1$ two-dimensional transformations. First apply

$$A_2=\frac{1}{\sqrt{|a_1|^2+|a_2|^2}}\begin{pmatrix}a_1^* & a_2^*\\ a_2 & -a_1\end{pmatrix} \tag{8}$$

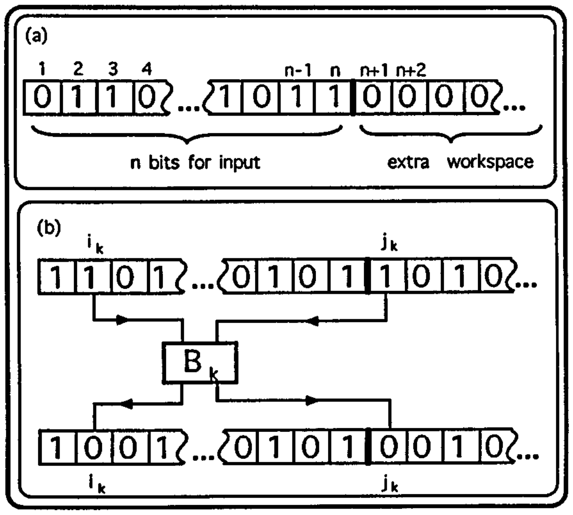

FIG. 3. Computational network: (a) Starting configuration; (b) kth computational step.

in the $\{|e_1\rangle,|e_2\rangle\}$ space, which reduces a_2 to zero. Similarly use $A_3,\ldots,A_d$ (in the span of $\{|e_1\rangle,|e_3\rangle\},\ldots,\{|e_1\rangle,|e_d\rangle\}$) to reduce $a_3,\ldots,a_d$ to zero too. Thus $A_d\ldots A_3A_2$ maps $(a_1,\ldots,a_d)$ to $(1,0,\ldots,0)$, and $A_2^{-1}\ldots A_d^{-1}$ reverses the action.

Now let $|\phi_1\rangle,\ldots,|\phi_d\rangle$ be eigenvectors of U with eigenvalues $e^{i\phi_1},\ldots,e^{i\phi_d}$. Using the above procedure, transform $|\phi_1\rangle$ to $|e_1\rangle$, then multiply by $e^{i\phi_1}$ in span $\{|e_1\rangle\}$, then map $|e_1\rangle$ back to $|\phi_1\rangle$. This involves $(d-1)+1+(d-1)=2d-1$ two-dimensional transformations. Repeat for each eigenvector, thus expressing U as a product of $d(2d-1)$ two-dimensional transformations.

Using the above results (together with some classical programs giving some simple permutations of the computational basis states), we see that, for any n-dimensional unitary transformation U, there exists a program of length poly(n/ϵ) which approximates U to within accuracy ϵ.

Thus if we have an algorithm $\mathcal{A}$ which runs for k steps and so involves at most k unitary transformations U_i, we may approximate it on $\mathcal{C}$ to any desired accuracy ϵ by approximating each U_i to accuracy ϵ/k. This requires at most k poly(dk/ϵ) steps, where d is the maximum size of the U_i's. Furthermore, if $\mathcal{A}$ is an efficient algorithm, i.e., $k=$ poly(log$n$) for input n, and each U_i has size poly(log$n$), then, for each fixed accuracy ϵ, we will have an efficient approximating algorithm.

It is important to note that, in the above construction, a general unitary transformation of dimension d requires poly(d) steps for its implementation. Consider now a prospective quantum factoring algorithm with input N. The input, written in $\log_2 N$ two-state systems, occupies N Hilbert-space dimensions. Hence a general unitary transformation (implemented as above) on the input requires poly(N) steps, which is exponential in the input size $\log_2 N$. Thus, if the algorithm is to be efficient, we cannot cavalierly apply unitary transformations to the input, but must demonstrate that any such transformations used are rather special—implementable in poly (log$N$) steps rather than poly(N) steps.

The sequential construction of unitary transformations out of basic components may also be viewed in terms of *quantum networks* (Deutsch 1989). The efficacy of this model was highlighted by Shor (1994), and it fits in particularly well with proposed experimental realizations of quantum computation (see Sec. IX below).

A quantum computational network is the natural generalization of a classical (acyclic) Boolean network (see Dunne, 1988 and Papadimitriou, 1994, Secs. 4.3 and 11.4). Note first that a classical computation may be viewed in the following way (see Fig. 3). For any input of size n we set up the input in n bits, followed by a string of bits in a standard state 0, providing extra working space. Each step of the computation consists of selecting two bits—bits (i_k,j_k) for the kth step—and applying a specified *Boolean gate* B_k to these bits, which are subsequently replaced in the string. Here a Boolean gate B is a Boolean operation with two input bits and two output bits. The gates B_k are chosen from some finite fixed set of gates. It can be shown that various small finite sets of gates suffice to perform any computation in this way.

Thus for each input size n we have a specified sequence of Boolean gate operations (the "program"), and their concatenated sequential application may be viewed as a network of gates $\mathcal{G}_n$. The full computation $\mathcal{C}$ corresponds to a family of networks $\mathcal{C}=\{\mathcal{G}_0,\mathcal{G}_1,\ldots\}$, parametrized by the input size.

Remark. The complete description of this model of computation requires a further technical condition—that the family of networks $\{\mathcal{G}_n\}$ be uniform in n. This condition is discussed in detail in Papadimitriou (1994) Sec. 11.4. Roughly speaking, given n, it must be possible to generate the network $\mathcal{G}_n$ in a suitably regular and efficient way. If $\mathcal{G}_n$ is merely required to exist without further restriction, then one could encode complex computations and even noncomputable functions in the structure of $\mathcal{G}_n$ as n changes, which is clearly unsatisfactory.

To transfer the above model to the quantum context, we simply replace the string of bits by a string of qubits, and the Boolean gates B are replaced by *quantum gates*, i.e., unitary transformations U operating on two qubits chosen from some finite subset of four-dimensional unitary transformations. This leads directly to the concept of a family of quantum networks $\mathcal{Q}_n$. More generally, we may also consider quantum gates that operate on any finite number of qubits. Vedral *et al.* (1995) describe explicit constructions of quantum networks, effecting basic arithmetic operations such as modular addition, multiplication, and exponentiation.

The computational complexity of a family of networks $\{\mathcal{Q}_n\}$ (or $\{\mathcal{G}_n\}$) may now be defined in terms of the size of $\mathcal{Q}_n$, i.e., the number of gates involved in $\mathcal{Q}_n$. A computation $\mathcal{C}$ will be efficient if it has a (uniform) family of *polynomial-size* networks, i.e., a family $\{\mathcal{Q}_n\}$, such that the size of $\mathcal{Q}_n$ grows only polynomially with n. In the next section we shall decribe a simple example of a family of polynomial-size quantum networks for the discrete Fourier transform on n qubits—a unitary transformation central to Shor's algorithm.

The issue of which finite sets of quantum gates suffice to build any unitary transformation has been much studied. Deutsch (1989) described a single three-qubit gate (now referred to as the Deutsch gate) which suffices to construct an arbitrary unitary transformation with arbitrary precision. Subsequently Barenco, Bennett, *et al.* (1995) and DiVincenzo (1995a) showed that the Deutsch gate can be implemented using families of two-qubit gates. It was then shown (Barenco 1995; Sleator *et al.* 1995) that a single two-qubit gate suffices to implement the Deutsch gate. Finally, it has been shown (Deutsch *et al.* 1995; Lloyd 1995a) that almost any two-qubit gate by itself is universal, i.e., suffices to build any unitary transformation with arbitrary precision.

The reduction to two-qubit gates is especially significant for experimental considerations. Such gates correspond to the interaction of just two physical systems. A particularly relevant example of a two-qubit gate is the quantum controlled NOT gate. The classical controlled NOT gate is a reversible logic gate operating on two bits, ϵ_1 and ϵ_2, of which ϵ_1 is called the control bit and ϵ_2 the target bit. The value of ϵ_2 is negated if $\epsilon_1=1$; otherwise ϵ_2 is left unchanged (in both cases the control bit ϵ_1 remains unchanged). The quantum controlled NOT gate $\mathcal{C}_{12}$ is the unitary operation on two qubits, which, in a chosen orthonormal basis $\{|0\rangle,|1\rangle\}$, reproduces the controlled NOT operation:

$$|\epsilon_1\rangle|\epsilon_2\rangle \overset{\mathcal{C}_{12}}{\rightarrow} |\epsilon_1\rangle|\epsilon_1\oplus\epsilon_2\rangle, \tag{9}$$

where $\oplus$ denotes addition modulo 2. It may be shown (c.f. Barenco, Bennett, *et al.*, 1995) that the quantum controlled NOT gate, together with simple *one*-qubit gates, is sufficient for any arbitrary quantum computation.

It must be emphasized that, in all of the above constructions, the implementation of a general n-qubit gate typically requires a network whose size is exponential in n, in contrast to special transformations like the discrete Fourier transform on n qubits, which has polynomial-size networks.

V. THE DISCRETE FOURIER TRANSFORM

The discrete Fourier transform modulo$q$, denoted DFT$_q$, is a unitary transformation in q dimensions. It is defined relative to a chosen basis $|0\rangle,\ldots,|q-1\rangle$ by

$$\mathrm{DFT}_q:|a\rangle\mapsto\frac{1}{\sqrt{q}}\sum_{c=0}^{q-1}\exp(2\pi iac/q)|c\rangle. \tag{10}$$

Note that $|0\rangle$ is transformed into an equal superposition of all $c$mod$q$. More generally, DFT$_q$ effects the discrete Fourier transform of the input amplitudes. If

$$\mathrm{DFT}_q:\sum_a f(a)|a\rangle\mapsto\sum_c \tilde{f}(c)|c\rangle, \tag{11}$$

then the coefficients $\tilde{f}(c)$ are the discrete Fourier transform of $f(a)$'s, i.e.,

$$\tilde{f}(c)=\frac{1}{\sqrt{q}}\sum_a \exp(2\pi iac/q)f(a). \tag{12}$$

Looking ahead a little, let us mention that we shall need to apply DFT$_q$ for $q\approx N^2$, where N is the number we want to factorize. Hence we shall need an efficient method for performing DFT$_q$, using poly(log$q$) steps rather than the poly(q) steps given by the general construction in Sec. IV. Furthermore, it will be sufficient to take $q=2^L$, with L an integer. In this particular case the efficient quantum algorithm can be constructed as a quantum analog of the fast-Fourier-transform algorithm [e.g., as described in Knuth (1981)]. This efficient classical algorithm needs to be reexpressed in terms of unitary operations (Coppersmith, 1994; Deutsch, 1994). To do this we introduce a quantum register composed of L qubits. We follow the convention in the literature of using the same ket notation to denote qubit states and states of a register. The context will make clear which is meant. Each qubit has two basis states, which are labeled as $|0\rangle$ and $|1\rangle$; these states define the computational basis in the 2^L-dimensional state space of the register.

$$|i_{L-1},\ i_{L-2},\ldots i_0\rangle=|i_{L-1}\rangle\otimes|i_{L-2}\rangle\otimes\cdots\otimes|i_0\rangle. \tag{13}$$

The register can store any number from 0 to 2^L-1. Numbers such as a and c are represented by kets $|a\rangle$ and $|c\rangle$, which can be expressed in binary form as

$$|a\rangle=|a_{L-1},\ a_{L-2},\ldots a_0\rangle,\quad a=\sum_{i=0}^{L-1}a_i 2^i, \tag{14}$$

$$|c\rangle=|c_{L-1},c_{L-2},\ldots c_0\rangle,\quad c=\sum_{i=0}^{L-1}c_i 2^i. \tag{15}$$

Let us also define the L-bit integer b which is the reversal of c,

$$|b\rangle\equiv|b_{L-1},b_{L-2},\ldots,b_0\rangle=|c_0,c_1.\ldots,c_{L-1}\rangle. \tag{16}$$

To construct DFT$_{2^L}$ we shall need only two basic unitary operations. In terms of matrices in the computational basis they are A_j operating on the qubit in position j,

$$A_j=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \tag{17}$$

and $B_{j,k}$ operating on the qubits in positions j and k,

$$B_{jk}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{i\theta_{jk}}\end{pmatrix}, \tag{18}$$

where $\theta_{jk}=\pi/2^{k-j}$. The matrix A_j is represented in the $|a_j\rangle$ basis and matrix B_{jk} in the $|a_j,a_k\rangle$ basis ($a_i=0,1$). Transformation B_{jk} is an elementary two-qubit operation, which affects only states with a 1 in both positions j and k, regardless of the state of the remaining qubits.

Following the strategy of the classical fast Fourier transform, to perform DFT_{2^L} on $|a\rangle$, we shall repeat a certain sequence of operations L times. We shall index each pass by j, which will go from $j=L-1$ down to $j=0$. For the first pass we set $j=L-1$ and apply A_j to the leading-order bit a_{L-1}. Then we reduce j by one, $j=L-2$, and, in the second pass, we apply (reading from left to right)

$$B_{j,L-1}A_j. \tag{19}$$

For each successive pass we reduce j by one and apply (reading from left to right)

$$B_{j,j+1}B_{j,j+2}\dots B_{j,L-1}A_j. \tag{20}$$

We continue down to $j=0$, finally completing the procedure with operation A_0. The end result of the L passes is the transformation

$$|a\rangle \mapsto \frac{1}{\sqrt{q}}\sum_{c=0}^{q-1}\exp(2\pi iac/q)|b\rangle. \tag{21}$$

This is equivalent to DFT_{2^L}, if $|b\rangle$ is read in the bit reversed order, giving $|c\rangle$. Alternatively, $|b\rangle$ may be efficiently transformed to $|c\rangle$ by a sequence of state swappings.

Example. DFT_{16} is performed as the sequence of the following 10 elementary operations (read from left to right):

$$(A_3)(B_{23}A_2)(B_{13}B_{12}A_1)(B_{03}B_{02}B_{01}A_0). \tag{22}$$

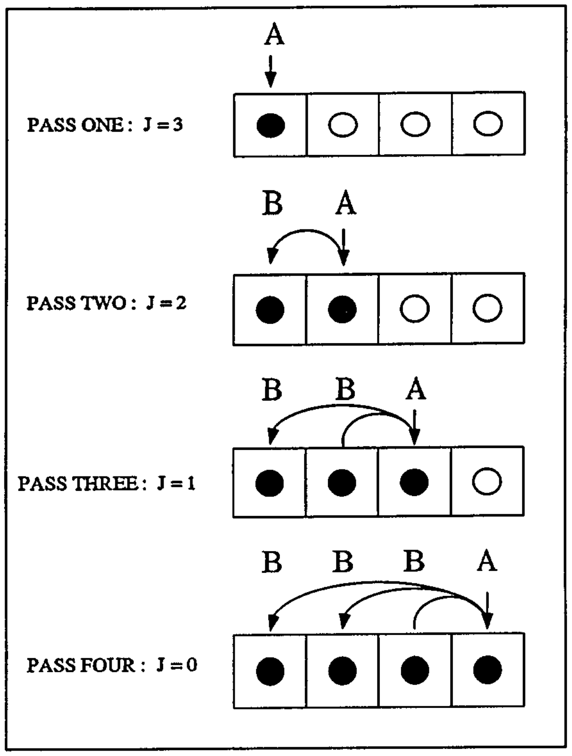

FIG. 4. Schematic illustration of the quantum discrete Fourier transform DFT_{16} performed on a four-qubit register.

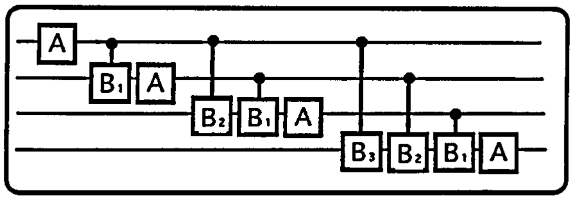

FIG. 5. Quantum network for the quantum discrete Fourier transform DFT_{16} performed on a four-qubit register. In this picture gate B_k performs the conditional phase shift by $2\pi/2^k$.

Figure 4 provides a graphical illustration of this sequence of transformations, and Fig. 5 shows the corresponding quantum network.

In general, for an input of size L we have to perform L operations A_j, and $L(L-1)/2$ operations B_{jk}, in total $L(L+1)/2$ elementary operations. The execution time of DFT_{2^L} grows as a quadratic function of the input size L; so the algorithm is an efficient one.

Remark. Coppersmith (1994) also describes the approximate discrete Fourier transform (ADFT). In ADFT one neglects those operations B_{jk} for which the phase shift $\theta_{jk}\equiv\pi/2^{k-j}<\pi/2^M$ for some M, such that $0\le M\le L-1$. The matrix elements of DFT and ADFT differ by multiplicative factors of the form $\exp(i\epsilon)$ with $|\epsilon|\le 2\pi L/2^M$. The execution time of ADFT grows as $L(M+1)$.

Remark. Shor (1994), in the original version of his paper, showed that DFT_q can be performed efficiently when q is *smooth*, i.e., when all of the prime power factors of q are "suitably small"—bounded by $A(\log q)^d$—where A,d are constants independent of q. Later, Coppersmith (1994) and Deutsch (1994) independently devised the efficient Fourier transform for $q=2^L$ described above. More recently Cleve (1994) has shown that the Fourier transform can also be efficiently performed when the prime factors of q are bounded by $A(\log q)^d$ (although prime power factors need not be bounded in this way). A special case of this is when q is a power of 2.

VI. THE NUMBER-THEORETIC BASIS OF THE FACTORING METHOD IN SHOR'S ALGORITHM

In order to factor a number N, we shall use quantum computation to solve an equivalent problem. Given N, choose randomly a fixed number y coprime to N [that is, the greatest common division of the pair (y,N), denoted by $\gcd(y,N)$ is equal to one]. The major task in the algorithm will be to find the period r of the following function:

$$F_N(a)=y^a \bmod N. \tag{23}$$

In this section we explain the connection between the periodicity of $F_N(a)$ and the factorization of N. Relevant results from number theory are collected in Appendix A.

Consider the quadratic equation (cf. Riesel, 1985)

$$x^2 \equiv 1 \bmod N, \tag{24}$$

which always has solutions $x \equiv \pm 1 \bmod N$, the so-called trivial solutions. If N is an odd prime p, then these are the only solutions (since multiplication modulo$p$ has inverses and $x^2-1 \equiv (x-1)(x+1) \equiv 0 \bmod p$ implies $x-1 \equiv 0$, or $x+1 \equiv 0 \bmod p$ by division). However, if N is composite, then there are also pairs of nontrivial solutions of the form $x \equiv \pm a \bmod N$.

Example. The equation

$$x^2 \equiv 1 \bmod 341, \tag{25}$$

apart from the two trivial solutions, $x \equiv \pm 1 \bmod 341$, also has two nontrivial solutions, $x \equiv \pm 32 \bmod 341$. This arises because 341 is composite, $341 = 11 \times 31$.

To see this in a general case, let $N = n_1 n_2$ with $\gcd(n_1, n_2) = 1$ [where $\gcd(n_1, n_2)$ denotes the greatest common divisor or the highest common factor of n_1 and n_2], and consider the four sets of equations:

$$\text{(a)}\begin{cases} x_1 \equiv 1 \bmod n_1 \\ x_1 \equiv 1 \bmod n_2 \end{cases} \quad \text{(b)}\begin{cases} x_2 \equiv -1 \bmod n_1 \\ x_2 \equiv -1 \bmod n_2 \end{cases}$$

$$\text{(c)}\begin{cases} x_3 \equiv 1 \bmod n_1 \\ x_3 \equiv -1 \bmod n_2 \end{cases} \quad \text{(d)}\begin{cases} x_4 \equiv -1 \bmod n_1 \\ x_4 \equiv 1 \bmod n_2. \end{cases} \tag{26}$$

In each case $x_i^2 \equiv 1 \bmod n_1$ and $\bmod n_2$; so each x_i satisfies Eq. (24). By the Chinese remainder theorem (see Appendix A.1) each set has a unique solution modulo N. From (a) and (b) we clearly get $x_1 = 1$ and $x_2 = -1$, the trivial solutions of Eq. (24), and from (c) and (d) we get $x_3 = a$ and $x_4 = -a \bmod N$, giving a pair of nontrivial solutions. Thus $(a+1)(a-1) \equiv 0 \bmod N$, and $a \pm 1$ are nonzero. Hence N divides $(a+1)(a-1)$, but N does not divide $a \pm 1$ (as $a \pm 1 \leqslant N+1$). Hence (if $a \neq \pm 1$) the greatest common divisor of N and $a \pm 1$ is a nontrivial factor of N. The greatest common divisor of two given numbers can be found efficiently using Euclid's algorithm (see Appendix A.2).

Example. Taking numbers from the preceding example we can easily check, using Euclid's algorithm, that the two factors of 341 can be obtained as $\gcd(31,341) = 31$ and $\gcd(33,341) = 11$.

In summary, given a nontrivial solution x of Eq. (24) we can efficiently find a nontrivial factor of N. We find such an x as follows. Given N, choose a random $y < N$. If y and N are coprime then let r be the order of y mod$N$ (see Appendix A.3). This is precisely the period of $F_N(a)$ in Eq. (23). Thus

$$y^r \equiv 1 \bmod N. \tag{27}$$

If r is also even, then setting

$$x = y^{r/2}, \tag{28}$$

we have $x^2 \equiv 1 \bmod N$, so x is a candidate for our nontrivial solution of Eq. (24). This provides the connection between the periodicity of $F_N(a)$ and the calculation of a nontrivial factor of N. The above process may fail if the chosen y value has an odd order r, or if r is even but $y^{r/2}$ turns out to be a trivial solution of Eq. (24). However, according to the following theorem, these situations can arise only with suitably small probability if y is chosen at random.

Theorem. Let N be odd with prime factorization

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}. \tag{29}$$

Suppose y is chosen at random, satisfying $\gcd(y,N) = 1$. Let r be the order of y mod$N$. Then

$$\text{Prob}\,(r \text{ is even and } y^{r/2} \not\equiv \pm 1 \bmod N) \geqslant 1 - \frac{1}{2^{k-1}}. \tag{30}$$

The proof is provided in Appendix B.

In the above theorem we have excluded all even values of N. This is unimportant for the factorization problem, as all factors of 2 are easy to recognize and remove.

Remark. The proof (see Appendix B) may be easily extended to show that

$$\text{Prob}\,(r \text{ even and } y^{r/2} \not\equiv \pm 1 \bmod N) \geqslant \frac{1}{2} \tag{31}$$

holds for all N which are not of the form p^α or $2p^\alpha$. In these cases the above probability is zero. Pure prime powers $N = p^\alpha$ are known to be efficiently recognizable by a classical probabilistic algorithm.

Note that if $1 \leqslant y \leqslant N$ is selected at random, then the probability of $\gcd(y,N) = 1$ is greater than $1/\log N$ (see Appendix A.3). Thus, assuming that we are able to compute the order r of y, we obtain a nontrivial factor of N with probability greater than $1/(2\log N)$ from the above process applied to a random y.

Example. Let us illustrate the above factoring method for $N = 15$. First we select y, such that $\gcd(y,N) = 1$, i.e., y could be any number from the set $\{2,4,7,8,11,13,14\}$. Let us pick up $y = 11$ and let us compute the order of 11 modulo 15. Values of $11^a \bmod 15$ for $a = 1,2,3,\ldots$ go as $11,1,11,1,11,\ldots$, giving $r = 2$. Then we compute $x = y^{r/2}$, which gives $x = 11$, and we find the largest common factor $\gcd(x \pm 1, N)$, i.e., gcd(10,15) and gcd(12,15), which gives 5 and 3, the two factors of 15.

Respective orders modulo 15 of elements $\{2,4,7,8,11,13,14\}$ are $\{4,2,4,4,2,4,2\}$. In this particular example any choice of y except $y = 14$ leads to the correct result. For $y = 14$ we obtain $r = 2$, $y^{r/2} \equiv -1 \bmod 15$, and the methods fails.

Note that 15 is the smallest odd number that is not a prime power, so it follows from the above discussion that 15 is the smallest integer that can be factorized by the preceding method.

Shor describes a quantum algorithm which provides the order r [when it exists, i.e., when $\gcd(y,N) = 1$] of a randomly chosen y. The algorithm runs in polynomial time, i.e., requires poly($\log N$) steps. It is a probabilistic algorithm, providing the value of r with any prescribed probability of success $1 - \epsilon$, $\epsilon > 0$.

VII. SHOR'S QUANTUM ALGORITHM FOR EFFICIENT FACTORIZATION

Given N, choose $q=2^L$ between N^2 and $2N^2$ (which is clearly always possible). The reason for choosing $q \geqslant N^2$ will become apparent at the end. The choice of q as a power of 2 is to ensure that DFT_q can be performed efficiently.

Next choose a random $y<N$ and begin with an L-bit register in the state $|0\rangle$ (i.e., all L qubits in states $|0\rangle$). Apply DFT_q to the register, giving

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle. \tag{32}$$

Next compute $y^a \bmod N$, storing the result in a second, auxiliary register, giving

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |y^a \bmod N\rangle. \tag{33}$$

This can be done efficiently, e.g., by repeatedly squaring modulo$N$ to get y^{2^i}'s and multiplying selected ones corresponding to the binary expansion of a.

Next perform a measurement in the computational basis to determine the bit values in the second register. Suppose that the result is z where $z=y^l \bmod N$ for some least l. If r is the order of y mod N, then $y^l \equiv y^{jr+l} \bmod N$ for all j. Thus the measurement will select a values (in the first register) of $a=l, l+r, l+2r, \ldots, l+Ar$, where A is the greatest integer less than $(q-l)/r$. Here $l \leqslant r$ is fixed and has been chosen probabilistically by the choice of z in the measurement. Note that $l \leqslant r < N$, and $q \approx O(N^2)$, so $A \approx q/r$.

The post-measurement state of the first register is then

$$|\phi_l\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |jr+l\rangle. \tag{34}$$

Thus we have a uniform superposition of labeled basis states, where the labels have been chosen with a period of r. From this state we wish to extract the information of the periodicity r with a probability that does not decrease too rapidly with the size of N. More precisely, we wish to extract the value of r with a constant probability if the above computation is repeated at most poly($\log N$) times. Note that, upon repetition (with y fixed), the value of l (and hence the final state $|\phi_l\rangle$) will vary, but r remains the same.

The extraction of r will be achieved by applying DFT_q. To see the principle of how this works, consider first the simplified situation in which r divides q exactly, so $A=q/r-1$. The final state corresponding to Eq. (34) is then

$$|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{q/r-1} |jr+l\rangle = \sum_{a=0}^{q-1} f(a)|a\rangle, \tag{35}$$

where $f(a)=\sqrt{(r/q)}$ if $a-l$ is a multiple of r and $f(a)=0$ otherwise. This state is exactly periodic in amplitude as we cycle through the labels, in contrast to Eq. (34), where the periodicity is slightly spoilt in recycling from the last to the first label. Write $M=q/r$. Performing DFT_q on $|\phi_l\rangle$ gives

$$\mathrm{DFT}_q|\phi_l\rangle = \sum_c \tilde{f}(c)|c\rangle, \tag{36}$$

where the amplitude $\tilde{f}(c)$ is the discrete Fourier transform of $f(a)$:

$$\begin{aligned}\tilde{f}(c) &= \frac{\sqrt{r}}{q} \sum_{j=0}^{q/r-1} \exp\left(\frac{2\pi i(jr+l)c}{q}\right) \\ &= \frac{\sqrt{r}}{q}\left[\sum_{j=0}^{q/r-1} \exp\left(2\pi i \frac{jrc}{q}\right)\right] \exp\left(2\pi i \frac{lc}{q}\right).\end{aligned} \tag{37}$$

The term in the square bracket on the rhs is zero unless c is a multiple of q/r. In the latter case it equals q/r. Hence

$$\tilde{f}(c) = \begin{cases} \exp(2\pi i l c/q)/\sqrt{r} & \text{if } c \text{ is a multiple of } M. \\ 0 & \text{otherwise,} \end{cases} \tag{38}$$

i.e., the Fourier transform of a state with period r is a state with period $M=q/r$. Writing $c=jq/r$, we get

$$\mathrm{DFT}_q|\phi_l\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \exp(2\pi i l j/r) \left| j\frac{q}{r} \right\rangle. \tag{39}$$

A measurement of the state's label c will yield a multiple $\lambda q/r$ with $\lambda = 0, \ldots, r-1$ chosen equiprobably. Note that the initial shift l appears neither in the probabilities nor in the seen labels $\lambda q/r$. Thus the Fourier transform "inverts" the periodicity of the input $(r \rightarrow q/r)$ and has a translational invariance property that washes out the shift l (see Fig. 6).

Now after our measurement of the label we see a value of c satisfying $c/q=\lambda/r$. Here c and q are known. If $\gcd(\lambda, r)=1$, we can determine r by canceling c/q down to an irreducible fraction. Since λ is chosen at random, the probability that $\gcd(\lambda, r)=1$ is greater than $1/\log r$ for largish r (see Appendix A.3). Thus, if we repeat the computation $O(\log r)<O(\log N)$ times, we can amplify the success probability as close to 1 as we wish. This gives an efficient determination of r.

Remark. If l could be made constant in repeated preparations of $|\phi_l\rangle$, then we would not need DFT_q at all: just measure the label on three copies of $|\phi_l\rangle$, obtaining $c_i = j_i r + l$ for $i=1,2,3$, so that $c_1-c_2=(j_1-j_2)r$ and $c_1-c_3=(j_1-j_3)r$. Since the j's are all equiprobable, we get by repetition, as above, a high probability of $\gcd(j_1-j_2, j_1-j_3)=1$, so r is obtained as $\gcd(c_1-c_2, c_1-c_3)$. Unfortunately in our application l varies randomly [corresponding to the outcome of the measurement on the last slot of Eq. (33)], and the translational invariance property of DFT_q is its main role here.

Remark. It is interesting to note that the second register is used only to prepare a periodic state in the first register. After the function $y^a \bmod N$ in Eq. (33) has

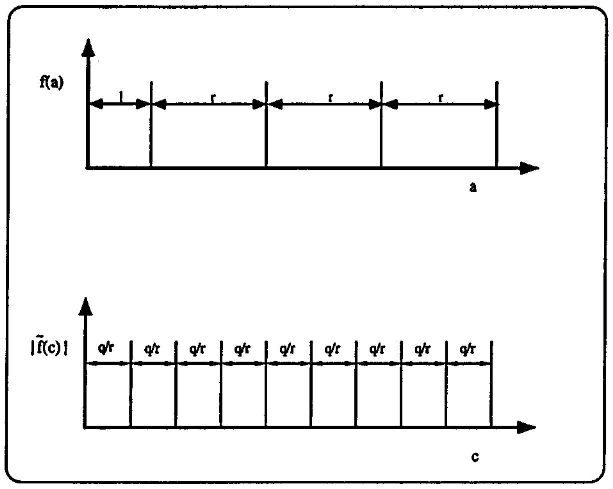

FIG. 6. Function $f(a)$ and the modulus of its Fourier transform $\tilde{f}(c)$.

been computed, one can effectively forget about the existence of the second register. Indeed it is not even necessary to perform a measurement on it! If a measurement is performed as described above, the actual value of the outcome z is not used in the algorithm at all.

Let us now return to our actual starting state Eq. (34) with its slightly imperfect periodicity. For simplicity we shall replace $A+1$ by q/r, neglecting a small roundoff error—this makes no essential difference in the following formulae, which then become more readable. Performing DFT_q on Eq. (34) gives

$$\mathrm{DFT}_q|\phi_l\rangle=\sum_{c=0}^{q-1}\tilde{f}(c)|c\rangle, \tag{40}$$

where

$$\tilde{f}(c)=\frac{\sqrt{r}}{q}\sum_{j=0}^{q/r-1}\exp[2\pi i(jr+l)c/q]. \tag{41}$$

Then the probability of seeing label c is

$$\mathrm{Prob}(c)=\frac{r}{q^2}\left|\sum_{j=0}^{q/r-1}\exp[2\pi ij(rc\bmod q)/q]\right|^2. \tag{42}$$

In the previous example constructive interference [Eq. (38)] occurred precisely for c satisfying $rc \bmod q=0$. In Eq. (42) we look for constructive interference by considering c such that $rc \bmod q$ is suitably small. Then the added terms will all be bunched on one side of the unit circle, $|z|=1$. In fact, if c satisfies

$$-r/2\leqslant rc\bmod q\leqslant r/2, \tag{43}$$

then the terms in $\mathrm{Prob}(c)$ will all be spread around at most a semicircle.

There are precisely r values of $c \bmod q$ satisfying Eq. (43) just as there are r nonzero terms in Eq. (38). To see this consider the multiples of q: $0,q,2q,\ldots,rq$ and the multiples cr of r: $0,r,2r,\ldots,qr$, marked on the same line. The multiples of r are spaced r apart, so, for each of the r multiples of q, there will be exactly one associated multiple of r within distance $\pm r/2$. This gives the r solutions of Eq. (43).

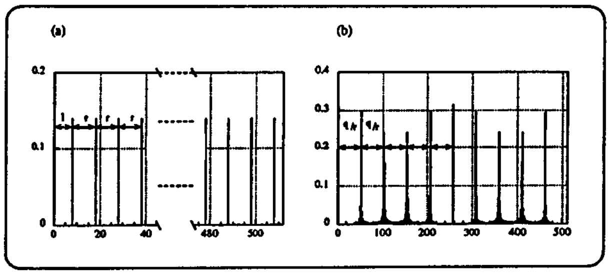

FIG. 7. Function $f(a)$ and the modulus of its Fourier transform $\tilde{f}(c)$; in this example $q=512$ (9 qubits), $l=8$, and $r=10$. The peaks occur for values of c having $rc \bmod q$ small as in Eq. 43, i.e. for c near the multiples of q/r.

Next we wish to estimate the size of $\mathrm{Prob}(c)$ for c satisfying Eq. (43). Write $\theta_c=2\pi(rc \bmod q)/q$ so that $\mathrm{Prob}(c)$ involves a geometric series with ratio $\exp i\theta_c$. By viewing these terms as vectors on an Argand diagram, we see that the total distance from the origin decreases as θ_c increases. Hence $\mathrm{Prob}(c)$ $\geqslant$Prob(the c with largest allowed θ_c). Now by Eq. (43), $\theta_c\leqslant\pi r/q$, and summing the geometric series with $\theta_c=\pi r/q$ gives

$$\mathrm{Prob}(c)\geqslant\frac{r}{q^2}\frac{1}{\sin^2\frac{\pi r}{2q}}\approx\frac{4}{\pi^2}\frac{1}{r}, \tag{44}$$

where we have used $\sin(\pi r/2q)\approx\pi r/2q$, as r/q is small. Thus, since there are r such c's, the probability of seeing a c value satisfying Eq. (43) is greater than $4/\pi^2$.

Finally we wish to extract the information of the value of r, given a value of c satisfying Eq. (43) (see Fig. 7). To do this we note that Eq. (43) is equivalent to

$$|rc-c'q|\leqslant r/2 \tag{45}$$

for some $0\leqslant c'\leqslant r-1$. The r different values of c' are associated with the r possible values of c, so by Eq. (44) each value of c' has

$$\mathrm{Prob}(c')\geqslant\frac{4}{\pi^2}\frac{1}{r}. \tag{46}$$

Equation (45) may be written

$$\left|\frac{c}{q}-\frac{c'}{r}\right|\leqslant\frac{1}{2q}. \tag{47}$$

Here c and q are known, and $r\leqslant N$, $q\geqslant N^2$. Thus because $q\geqslant N^2$, there is exactly one fraction c'/r with denominator at most N in the range given by Eq. (47). This fraction may be found efficiently using the continued fraction expansion of c/q (see Appendix A.4) as one of its convergents. Hence, if $\gcd(c',r)=1$, we get the value of r. There are $\phi(r)$ such coprime values of c', so from Eq. (46) we get

$$\mathrm{Prob}(c'\text{ is coprime to }r)\geqslant\frac{4}{\pi^2}\frac{\phi(r)}{r}. \tag{48}$$

For large r we have $\phi(r)/r > 1/\log r > 1/\log N$ (see Appendix A.3), so with probability better than $4/(\pi^2 \log N)$ we obtain r and hence a factor of N. By repeating this efficient probabilistic algorithm $O(\log N)$ times we get an efficient factoring algorithm with an arbitrarily high success probability.

VIII. STABILITY OF QUANTUM COMPUTATION—DECOHERENCE

So far we have considered an unrealistic scenario, in which the computation was performed in complete isolation from the surrounding environment. The errors in the described algorithm were either of purely "mathematical" origin [e.g., a random choice of y in order to obtain $\gcd(y,N)=1$] or resulted from the probabilistic outcomes in the final bit-by-bit measurement. The probability of this type of error in a single run of computation grows only as a polynomial function of the input size, and therefore the errors do not affect the efficiency of the algorithm (see discussion of the randomized algorithms in Sec. III). However, when we analyze physically realizable computations, we have to consider errors 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 $\log N$. If the probability of success in a single run, $1-\epsilon$, falls exponentially with $\log N$, i.e., if $1-\epsilon = A\exp(-\alpha \log N)$, where A and α are positive constants, then the randomized algorithm can no longer technically be regarded as efficient, regardless of how weak the coupling to the environment may be. Unfortunately, as elaborated below, the computer-environment interaction leads to just such an unwelcome exponential increase of the error rate with input size. The degrading effect of the computer-environment interaction on the computer is generally known as decoherence (Zurek, 1991).

Note that a similar situation pertains to classical computation as well. Each bit in a classical computer is subject to some nonzero (tiny) probability of error (caused, for example, by a random cosmic-ray hit). Treating all bits as independent, this error also grows exponentially with input size. However, for classical computation there exist very efficient error-correcting schemes, which effectively allow the reduction of the constant α with increasing input size, so that $\alpha \log N$ remains small. Broadly speaking, these methods utilize redundancy, and all rely on measuring the computational state during the course of the computation, resetting all redundant copies to a majority answer. Unfortunately they cannot then be directly adopted for the stabilization of quantum computation, since measurement of intermediate computational states will destroy the coherences that underly the power of quantum computation. The formulation of alternative intrinsically quantum error-correction and stabilization schemes for quantum computation is the subject of much current study, and some theoretical proposals have been made by Berthiaume *et al.* (1994), Chuang and Laflamme (1995), Chuang and Yamamoto (1995), and Shor (1995). For quantum computation of any reasonable length ever to be physically feasible, it will be necessary to incorporate some efficiently realizable stabilization scheme to combat the effects of decoherence.

To study the typical effects of decoherence, let us consider a quantum register composed of L qubits with the selected basis states labeled as $|0\rangle$ and $|1\rangle$. Any quantum state of the register can be described by a density operator of the form

$$\rho(t) = \sum_{i,j=0}^{2^L-1} \rho_{ij}(t)|i\rangle\langle j|, \tag{49}$$

where $|i\rangle$ is defined as in Sec. V, as a tensor product of the qubit basis states,

$$|i\rangle = |i_{L-1}\rangle \otimes |i_{L-2}\rangle \otimes \cdots \otimes |i_0\rangle. \tag{50}$$

The rhs is the binary decomposition of the number $i=\sum_{l=0}^{L-1} 2^l i_l$.

Quantum computation derives its power from quantum interference and entanglement. The degree of the interference and entanglement in an L-qubit register is quantified by the coherences, i.e., the off-diagonal elements ρ_{ij} $(i \neq j)$ of the density operator in the computational basis. When a quantum computer is in contact with a thermal reservoir, the resulting dissipation destroys the coherences and changes the populations (the diagonal elements). In time the density matrix will approach the diagonal form,

$$\rho_{\text{thermal}} = \sum_{i=0}^{2^L-1} \frac{\exp(-E_i/kT)}{Z} |i\rangle\langle i|, \tag{51}$$

where we have assumed that states $|i\rangle$ are energy eigenstates with corresponding energies E_i; Z is the partition function, $Z=\sum_{i=0}^{2^L-1}\exp(-E_i/kT)$, k is the Boltzmann constant, and T is the temperature of the reservoir. The time scale in which thermal equilibrium is reached depends on the type of system-environment coupling. The rates of change for the diagonal and the off-diagonal elements of the density matrix are usually different. The decoherence, i.e., the decay of the off-diagonal elements, has the most significant effect on quantum interference in quantum computation, and therefore some simple models of decoherence that ignore changes in the populations are quite adequate. Within such a simplified model of the computer-environment interaction, it is assumed that the register in the computer and the environment undergo the following unitary evolution:

$$|i\rangle|R\rangle \rightarrow |i\rangle|R_i(t)\rangle, \tag{52}$$

where $|i\rangle$ is the state from the computational basis and $|R\rangle$ is the initial state of the environment. States $|R_i(t)\rangle$ are normalized but not necessarily orthogonal to each other. Now, consider the following initial state of the computer and the reservoir,

$$|\Psi(0)\rangle = \overbrace{\left(\sum_i c_i(0)|i\rangle\right)}^{\text{computer}} \otimes \overbrace{|R\rangle}^{\text{reservoir}}. \tag{53}$$

The unitary evolution of the composed system results in an entangled computer-reservoir state, which can be written as

$$|\Psi(t)\rangle = \sum_i c_i(t)|i\rangle \otimes |R_i(t)\rangle, \tag{54}$$

where, in general, $\langle R_i|R_j\rangle \neq 0$ for $i \neq j$. The elements of the density matrix evolve as

$$\rho_{ij}(0) = c_i(0)c_j^*(0) \rightarrow \rho_{ij}(t) = c_i(t)c_j^*(t)\langle R_i(t)|R_j(t)\rangle. \tag{55}$$

According to a popular model of decoherence (Zurek, 1991), the environment effectively acts as a measuring apparatus; in time the reservoir states $\{|R_i\rangle\}$ become more and more orthogonal to each other whilst the coefficients $\{c_i\}$ remain unchanged. Consequently the off-diagonal elements ρ_{ij} disappear due to the $\langle R_i(t)|R_j(t)\rangle$ factors.

If this model is applied to a single qubit coupled to the thermal electromagnetic radiation at a low temperature T, the relevant coherences develop a characteristic decay proportional to $\exp(-t/t_d)$, where t_d is the typical decoherence time. Its value depends on the physical representation of the qubits and their interaction with the environment and can vary from 10^4 s for nuclear spins in a paramagnetic atom to 10^{-12} s for electron-hole excitations in the bulk of a semiconductor. [DiVincenzo (1995b) provides numerical estimates of t_d for several selected physical realizations of qubits.] If we increase the number of qubits to L, then some coherences in the $2^L \times 2^L$ density matrix decay roughly as the product of the L single-qubit coherences, i.e., as $\exp(-tL/t_d)$; the characteristic decoherence time of L qubits becomes effectively t_d/L [for a detailed description of decoherence in quantum computers see Unruh (1994) and Palma *et al.* (1995)]. This gives rise to a probability of error ϵ, which increases exponentially with the register size $L = \log N$ and implies that decoherence cannot be efficiently dealt with by simply increasing the number of runs. As noted above, we shall need some other form of "quantum error correction" to stabilize the computation.

Decoherence puts an upper bound on the length of any feasible quantum computation. If the elementary computational step takes time τ and t_d is a decoherence time of a single qubit, then the requirement that a coherent computation of K steps be completed within the decoherence time of the computer can be written as

$$\tau K < t_d/L. \tag{56}$$

In the case of Shor's algorithm for factorization of an L-bit number, we may take $K = L^2$, and Eq. (56) provides an upper bound,

$$L < \left(\frac{t_d}{\tau}\right)^{1/3}. \tag{57}$$

Thus the ratio t_d/τ, which depends on the technology employed, determines the limits of the algorithm, and it is unrealistic to assume that this ratio can be made infinite.

IX. BUILDING QUANTUM LOGIC GATES

Quantum logic gates are the basic units of quantum control. In order to implement them 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,

$$U = |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1 + \cdots + |k\rangle\langle k| \otimes U_k, \tag{58}$$

where the projectors refer to quantum states of the control subsystem and the unitary operations U_i are performed on the target subsystem. The simplest nontrivial operation of this sort is probably a conditional phase shift, such as B_{jk}, which we used to implement the discrete Fourier transform [see Eq. (18)]. Here we shall illustrate the conditional dynamics with another simple operation, described in Sec. IV, called the quantum controlled NOT. The quantum controlled NOT gate is not a universal gate, but a universal quantum gate can be constructed by a combination of the controlled NOT and simple unitary operations on a single qubit (for more details about the relevance of the controlled NOT gate and its possible implementations, see Barenco, Deutsch, *et al.*, 1995).

We next identify some possible physical processes that can lead to a controllable conditional dynamics. In the following we outline three possible experimental realizations of the quantum controlled NOT gate. The first method is based on cavity quantum electrodynamics (cavity QED; Brune *et al.*, 1994; Davidovich *et al.*, 1994; Turchette *et al.*, 1995), the second on the selective driving of optical resonances of two qubits undergoing a dipole-dipole interaction (Obermayer *et al.*, 1988; Teich *et al.*, 1988; Lloyd, 1993b; Barenco, Deutsch, *et al.*, 1995), and the third on selective excitation of phononic and electronic states of trapped ions (Cirac and Zoller, 1995; Monroe *et al.*, 1995). Some other interesting experimental work in quantum computation is described by Chuang and Yamamoto (1995).

A. Cavity QED

In the cavity-QED and atomic-interferometry method the target qubit is an atom with two selected circular Rydberg states $|\epsilon_2\rangle$, where $\epsilon_2 = 0,1$; the control qubit $|\epsilon_1\rangle$ is the quantized electromagnetic field in a high-Q cavity C. The field in the cavity contains at most one photon, so it can be viewed as a two-state system with the vacuum state $|0\rangle$ and the one-photon state $|1\rangle$ as the basis. The cavity C is sandwiched between two auxiliary microwave cavities R_1 and R_2, in which classical microwave fields produce $\pi/2$ rotations of the atomic Bloch vector,

$$|\epsilon_1\rangle_{\text{field}}|\epsilon_2\rangle_{\text{atom}} \rightarrow |\epsilon_1\rangle_{\text{field}} \frac{1}{\sqrt{2}} \times \left(|\epsilon_2\rangle + (-1)^{\epsilon_2} e^{i\alpha}|1-\epsilon_2\rangle\right)_{\text{atom}}. \tag{59}$$

Here the phase factor α is different for the two cavities R_1 and R_2. In the central cavity C a dispersive interaction with the quantized field introduces phase shifts, which depend on the state of the atom $|\epsilon_2\rangle$ and on the number of photons in the cavity $|\epsilon_1\rangle$. The interaction does not involve any exchange of excitation, so the number of photons in the cavity remains unchanged:

$$|\epsilon_1\rangle_{\text{field}}|\epsilon_2\rangle_{\text{atom}} \rightarrow \exp[i(-1)^{1-\epsilon_2}(\epsilon_1+\epsilon_2)\theta]|\epsilon_1\rangle_{\text{field}}|\epsilon_2\rangle_{\text{atom}}, \tag{60}$$

where θ, the phase shift per photon, can be tuned to be π (θ depends on the atom-cavity crossing time and the atom-field detuning).

The overall process can be viewed as a sequence of half-flopping in R_1, phase shifts in C, and half-flopping in R_2. Depending on the phase shifts the second half-flopping can either put the atom back into its initial state or flop completely into the orthogonal state. The whole interferometer can be adjusted so that, when the atom is sent through the cavities R_1, C, and R_2, the two qubits, i.e., the field and the atom, undergo the transformation

$$|\epsilon_1\rangle_{\text{field}}|\epsilon_2\rangle_{\text{atom}} \rightarrow |\epsilon_1\rangle_{\text{field}}|\epsilon_1\oplus\epsilon_2\rangle_{\text{atom}}. \tag{61}$$

The initial and final states of the field in C can be mapped from and to the atomic states, respectively, by a resonant atom-field interaction (Raimond *et al.*, 1989; Parkins *et al.*, 1993; Davidovich *et al.*, 1994). This process allows the two qubits to be of the same type, i.e., two Rydberg atoms rather than a field and an atom. The practical realization can be carried out by a modification of the atomic interferometry experiments described in Davidovich *et al.* (1994), and the ratio $t_d/\tau \approx 10^4$ can be achieved with the current technology.

The cavity-QED effects can also be used to implement a different type of conditional dynamics. Turchette *et al.* (1995) have proposed the use of polarized photons as two qubits and have shown that the polarization of one photon can be conditionally rotated when the two photons travel through a high-finesse Fabry-Perot optical cavity and interact with a beam of birefringent atoms. The experiment has been performed, and the experimental data show that a rotation of the order of 12° per photon is possible. However, as the authors point out, some further experimental measurements are required to verify that the conditional dynamics is of a genuinely quantum rather than a classical nature.

B. Dipole-dipole interactions

A second proposal for the practical implementation of the quantum controlled NOT gate relies on a dipole-dipole interaction between two qubits. For the purpose of this model the qubits could be either magnetic dipoles, e.g., nuclear spins in external magnetic fields, or electric dipoles, e.g., single-electron quantum dots in static electric fields. Here we describe the model in terms of interacting quantum dots, but the basic ideas have a wider applicability.

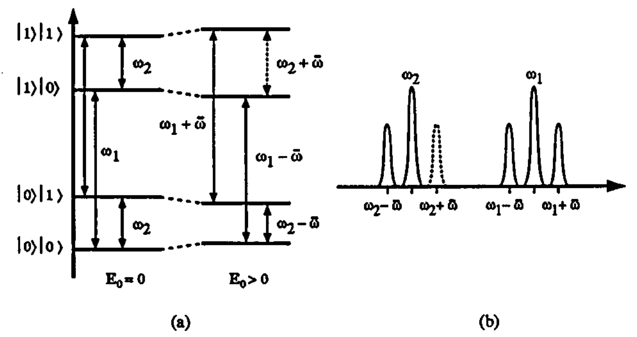

FIG. 8. Quantum dots as a controlled NOT gate: (a) Energy levels of two quantum dots without and with the coupling induced by the presence of a static electric field E_0. (b) Resonance spectrum of the two quantum dots. The dotted line shows the wavelength for which the two dots act as a controlled NOT gate, with the first dot being the control qubit and the second the target qubit.

Two single-electron quantum dots separated by a distance R are embedded in a semiconductor. The ground state and the first excited state of each dot are labeled as $|0\rangle$ and $|1\rangle$, respectively. The first quantum dot, with the resonant frequency ω_1, will act as the control qubit, and the second one, with the resonant frequency ω_2, as the target qubit. In the presence of an external static electric field, which can be turned on and off adiabatically in order to avoid transitions between the levels, the charge distribution of the ground state of each dot is shifted in the direction of the field, whilst the charge distribution of the first excited state is shifted in the opposite direction (the *quantum-confined Stark effect*). In the simple model in which the state of the qubit is encoded by a single electron per quantum dot, we can choose coordinates in which the dipole moments in state $|0\rangle$ and $|1\rangle$ are $\pm d_i$, where i=1,2 refers to the control and to the target dot respectively.

The electric field of the electron in the first quantum dot may shift the energy levels of the second one (and vice versa), but to a good approximation it does not cause transitions. This is because the total Hamiltonian

$$\hat{H} = \hat{H}_1 + \hat{H}_2 + \hat{V}_{12} \tag{62}$$

is dominated by a dipole-dipole interaction term V_{12} which is diagonal in the four-dimensional state space spanned by eigenstates $\{|\epsilon_1\rangle, |\epsilon_2\rangle\}$ of the free Hamiltonian $\hat{H}_1 + \hat{H}_2$, where ϵ_1 and ϵ_2, as before, range over 0 and 1. Specifically,

$$(\hat{H}_1 + \hat{H}_2)|\epsilon_1\rangle|\epsilon_2\rangle = \hbar(\epsilon_1\omega_1 k + \epsilon_2\omega_2)|\epsilon_1\rangle|\epsilon_2\rangle \tag{63}$$

and

$$\hat{V}_{12}|\epsilon_1\rangle|\epsilon_2\rangle = (-1)^{\epsilon_1+\epsilon_2}\hbar\bar{\omega}|\epsilon_1\rangle|\epsilon_2\rangle, \tag{64}$$

where

$$\bar{\omega} = -\frac{d_1 d_2}{4\pi\epsilon_0 R^3}. \tag{65}$$

As shown in Fig. 8, it follows that due to the dipole-

dipole interaction the resonant frequency for transitions between the states $|0\rangle$ and $|1\rangle$ of one dot *depends on the neighbor's state*. The resonant frequency for the first dot becomes $\omega_1 \pm \bar{\omega}$ depending on whether the second dot is in state $|0\rangle$ or $|1\rangle$. Similarly the second dot's resonant frequency becomes $\omega_2 \pm \bar{\omega}$, depending on the state of the first dot. Thus a pulse at frequency $\omega_2 + \bar{\omega}$ of sufficient strength to rotate the state by π (a π pulse) causes the transition $|0\rangle \leftrightarrow |1\rangle$ in the second dot only if the first dot is in $|1\rangle$ state.

A possible physical realization of the model requires the typical lifetime of the excited state $|1\rangle$ to be much greater than the time scale of the optical interaction (the length of the π pulse). For the π pulse to be monochromatic and selective enough, we must also require the length of the pulse to be greater than the inverse of the pulse carrier frequency and the inverse of the dipole-dipole interaction coupling constant.

Coupled quantum dots can, at least in principle, be conveniently integrated into a quantum network; however, the degree of control required is insurmountable. Let us mention some obvious experimental problems associated with a quantum circuit composed of single-electron quantum dots:

(i) Pulses that are not properly tuned may not be selective enough and may affect different transitions and different quantum dots. The minimal clock-cycle time τ is of the order of the typical pulse length t_p. The pulses must be approximately monochromatic (in order to be selective), which requires $t_p \gg 1/\omega$ and $t_p \gg 1/\bar{\omega}$, where ω is the carrier frequency of the pulse. Clearly higher carrier frequencies allow shorter pulses and consequently a shorter clock cycle.

(ii) The decoherence time t_d depends on the interaction with the environment. Electrons in quantum dots are coupled to the quantized electromagnetic vacuum, which leads to spontaneous emission. The typical lifetime of the excited state is of the order of

$$4\pi\epsilon_0 \frac{3\hbar c^3}{4D^2\omega^3}, \tag{66}$$

where D is the dipole moment between states $|0\rangle$ and $|1\rangle$, ω is the carrier frequency, tuned to the transition frequency between the two states. (D is the off-diagonal dipole moment generally different from the diagonal dipole moment d.) The interaction between the neighboring quantum dots will also have some off-diagonal terms, which were not included in V_{12}, and which induce the propagation of excitons in the array. This process reduces the lifetime of the excited states of the quantum dots and shortens the decoherence time.

For quantum dots of nanometer size, separated by $\approx 10^{-8}$ m, and with a resonant wavelength of the order of $\approx 10^{-6}$ m ($\approx 10^{14}$ Hz), realistic estimates could be $t_d \approx 10^{-8}$ s and $\bar{\omega} \approx 10^{12}$ Hz. Tunable vibronic solid-state lasers are available at these wavelengths (Hecht, 1992); for example, the titanium-sapphire lasers with mode locking can produce adequate short pulses $t_p \approx 10^{-13} - 10^{-10}$ s with nanosecond intervals. Assuming that the clock cycle τ is of the order 10^{-9} s, we should be able to carry out about 10 coherent steps. This is probably not enough to demonstrate the principle of "quantum parallelism." (N.B. one coherent step of the whole device implies many elementary transitions performed in parallel on many quantum dots.)

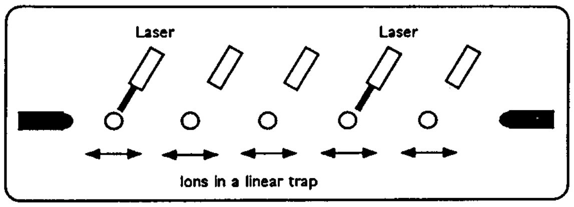

FIG. 9. Laser-cooled ions in a linear trap, well separated (by a few optical wavelengths), so that each ion can be individually addressed by a probing laser beam.

C. Selective excitation of trapped ions

Cirac and Zoller (1995) have proposed an interesting scheme for implementing quantum computation using ions (as qubits) moving in an ion trap and interacting with laser light. The computational basis states are represented by a selected pair of internal electronic states of an ion. To prepare a quantum register the ions are confined in a linear trap by an anisotropic harmonic-oscillator potential, which restricts their motion to one direction, and laser cooled (using sideband cooling techniques) so that they undergo very small oscillations around their equilibrium positions. Their collective quantized motion, which is due to electrostatic repulsion, can be analyzed in terms of phonon modes. The typical separation between the ions is of the order of a few optical wavelengths, so that each ion can be independently manipulated by a laser light, which can switch the ion-phonon interaction on and off (see Fig. 9). The conditional dynamics is then implemented as a sequence of laser-ion interactions whose effect depends on the presence or the absence of phonons.

To illustrate how this works, we describe a conditional phase shift performed on any two ions in the trap. Let $|\epsilon_1\rangle$ and $|\epsilon_2\rangle$ represent the computational basis of the first and the second ion, respectively (ϵ_i=0,1). These base states are two selected electronic states of each ion. Any ion in state $|1\rangle$, when probed with laser light of an appropriate frequency, can emit a phonon and undergo a transition from state $|1\rangle$ to state $|0\rangle$. The process can also be reversed,

$$|1\rangle|0 \text{ phonons}\rangle \overset{\text{laser light}}{\leftrightarrow} |0\rangle|\, 1 \text{ phonon}\rangle, \tag{67}$$

but only if the phonon is already present in the trap. If we probe the first ion with the laser π pulse tuned to the frequency corresponding to the energy difference between the states $|1\rangle|\,0\,\text{phonons}\rangle$ and $|0\rangle|\,1\,\text{phonon}\rangle$, then the four possible base states of the two selected ions in the absence of phonons undergo the following transitions:

$$|0\rangle|0\rangle|\,0\,\text{phonons}\rangle \rightarrow |0\rangle|0\rangle|\,0\,\text{phonons}\rangle, \tag{68}$$

$$|0\rangle|1\rangle|\,0\text{ phonons}\rangle \rightarrow |0\rangle|1\rangle|\,0\text{ phonons}\rangle, \quad (69)$$

$$|1\rangle|0\rangle|\,0\text{ phonons}\rangle \rightarrow -i|0\rangle|0\rangle|\,1\text{ phonon}\rangle, \quad (70)$$

$$|1\rangle|1\rangle|\,0\text{ phonons}\rangle \rightarrow -i|0\rangle|1\rangle|\,1\text{ phonon}\rangle. \quad (71)$$

After this operation we tune the laser to the $|0\rangle|\,1\text{ phonon}\rangle \leftrightarrow |\text{aux}\rangle|\,0\text{ phonons}\rangle$ transition and shine a 2π pulse on the second qubit. The state $|\text{aux}\rangle$ is an auxillary electronic state; by sending the 2π pulse we leave this state unexcited, but the transition from the $|0\rangle|\,1\text{ phonon}\rangle$ state back to the same state via the $|\text{aux}\rangle|\,0\text{ phonons}\rangle$ changes the sign in front of the $|0\rangle|\,1\text{ phonon}\rangle$ term resulting in the following,

$$|0\rangle|0\rangle|\,0\text{ phonons}\rangle \rightarrow |0\rangle|0\rangle|\,0\text{ phonons}\rangle, \quad (72)$$

$$|0\rangle|1\rangle|\,0\text{ phonons}\rangle \rightarrow |0\rangle|1\rangle|\,0\text{ phonons}\rangle, \quad (73)$$

$$-i|0\rangle|0\rangle|\,1\text{ phonon}\rangle \rightarrow i|0\rangle|0\rangle|\,1\text{ phonon}\rangle, \quad (74)$$

$$-i|1\rangle|1\rangle|\,1\text{ phonon}\rangle \rightarrow -i|0\rangle|1\rangle|\,1\text{ phonon}\rangle. \quad (75)$$

After this operation we return to the first qubit and repeat the first operation,

$$|0\rangle|0\rangle|\,0\text{ phonons}\rangle \rightarrow |0\rangle|0\rangle|\,0\text{ phonons}\rangle, \quad (76)$$

$$|0\rangle|1\rangle|\,0\text{ phonons}\rangle \rightarrow |0\rangle|1\rangle|\,0\text{ phonons}\rangle, \quad (77)$$

$$i|0\rangle|0\rangle|\,1\text{ phonon}\rangle \rightarrow |1\rangle|0\rangle|\,0\text{ phonons}\rangle, \quad (78)$$

$$-i|1\rangle|1\rangle|\,1\text{ phonon}\rangle \rightarrow -|1\rangle|1\rangle|\,0\text{ phonons}\rangle. \quad (79)$$

Altogether we have changed the sign in front of the $|1\rangle|1\rangle$ state, leaving all other base states unaffected.

Experiments with ion traps show that the decoherence time in such systems can be much longer than the time required to perform elementary computational steps. The preliminary experimental data indicate a decoherence time $t_d \approx 10^{-2}$ s and the switching time $\tau \approx 10^{-4}$ s, with possibilities for further improvement (Monroe *et al.*, 1995).

X. CONCLUDING REMARKS

It is not clear at present which technology, if any, will support quantum computation in the future. Nevertheless, the study of quantum computation is important both from the theoretical and the experimental perspective. Theoretically it offers a formal study of the intricate relationship between information, complexity, and the laws of physics. At the experimental level quantum computation can be viewed as a distinctive new way of harnessing nature. From the familiar harnessing of materials, forces, and energies, we have recently started to harness information in computers. The history of computer technology has involved a sequence of changes from one type of physical realization to another—from gears to relays to valves to transistors to integrated circuits and so on. The step to the molecular scale—the quantum level—will be next. Quantum theory is already important in the design of microelectronic components. Soon it will be necessary to harness quantum theory, rather than simply take it into account, giving data-processing devices a new functionality. It is important to know the limits on our experimental abilities to control nature at the quantum level, and the investigation of this fundamental issue justifies all experimental efforts in the direction of quantum computation.

ACKNOWLEDGMENTS

Thanks to David Deutsch, Peter Shor, and Umesh Vazirani for discussions and clarifications of many essential points.

APPENDIX A: NUMBER THEORY

We collect here some relevant results from number theory. Proofs and further explanations may be found in most standard texts on the subject, e.g., Hardy and Wright (1965), Schroeder (1990).

1. The Chinese remainder theorem

Consider a system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \bmod m_1 \\ &\vdots \\ x &\equiv a_k \bmod m_k. \end{aligned} \quad \text{(A1)}$$

The Chinese remainder theorem (Schroeder, 1990, Chapter 16) asserts that, if the moduli $m_1, \dots, m_k$ are coprime,

$$\gcd(m_i, m_j) = 1 \text{ for all } i \neq j, \quad \text{(A2)}$$

then Eq. (A1) has a *unique* solution for x in the range $1 \leqslant x \leqslant m_1 m_2 \cdots m_k$. Furthermore, the solution is given explicitly as follows. Set $M = m_1 m_2 \cdots m_k$ and $M_i = M/m_i$, so that $\gcd(m_i, M_i) = 1$ by Eq. (A2). Hence M_i has an inverse $N_i \equiv M_i^{-1} \bmod m_i$ (see end of Sec. A.2 below),

$$N_i M_i \equiv 1 \bmod m_i,$$

and the unique solution of Eq. (A1) is given by

$$x \equiv (a_1 N_1 M_1 + a_2 N_2 M_2 + \dots a_k N_k M_k) \bmod M. \quad \text{(A3)}$$

2. Euclid's algorithm

Given n_1 and n_2, Euclid's algorithm is an efficient method for computing the greatest common divisor $\gcd(n_1, n_2)$. Suppose that $n_1 \geqslant n_2$. Divide n_2 into n_1, giving remainder r_1:

$$n_1 = k_0 n_2 + r_1, \quad r_1 < n_2.$$

Do the same with n_2 and r_1,

$$n_2 = k_1 r_1 + r_2, \quad r_2 < r_1,$$

and then repeat with the two r's,

$$\begin{aligned} r_1 &= k_2 r_2 + r_3, \quad r_3 < r_2, \\ r_2 &= k_3 r_3 + r_4, \quad r_4 < r_3, \\ &\vdots \end{aligned}$$

until the remainder is zero (which must happen eventually, as the r's are strictly decreasing),

$$r_{l-1}=k_l r_l + r_{l+1}, \quad r_{l+1}<r_l,$$
$$r_l = k_{l+1} r_{l+1} + 0.$$

The greatest common divisor $\gcd(n_1,n_2)$ is then given (clearly!) by the last nonzero remainder,

$$\gcd(n_1,n_2)=r_{l+1},$$

By back-substituting up through the system of equations, starting with

$$r_{l+1}=r_{l-1}-k_l r_l,$$

we obtain an expression for $\gcd(n_1,n_2)$ as a linear combination of n_1 and n_2,

$$\gcd(n_1,n_2)=an_1+bn_2,$$

where a and b are integers depending on the k_i's. Hence, if n_1 and n_2 are coprime, we have

$$an_1+bn_2=1$$

so

$$an_1 \equiv 1 \bmod n_2,$$

i.e., $a \equiv n_1^{-1} \bmod n_2$. Thus coprime numbers always have multiplicative inverses modulo each other.

3. Euler's Phi function, orders modulo N, and the prime number theorem

Euler's Phi function denoted $\phi(N)$ (Hardy and Wright, 1965, Sec. 5.5; Schroeder, 1990, p. 9 and Chap. 8) is defined as

$$\phi(N)= \text{the number of integers less than } N \text{ which are coprime to } N.$$

Thus, for example, if p is prime, then $\phi(p)=p-1$ and $\phi(mn)=\phi(m)\phi(n)$, if $\gcd(m,n)=1$. Euler's Phi function is a basic ingredient in many beautiful results of number theory. Euler's theorem (Schroeder, 1990, Sec. 8.3) asserts that

$$a^{\phi(N)} \equiv 1 \bmod N \text{ if } \gcd(a,N)=1. \tag{A4}$$

Thus if $\gcd(a,N)=1$, there *exists* a power of a that is congruent to 1 mod$N$, so there is a *least* such power and we make the following definition:

Definition. Suppose that $\gcd(a,N)=1$. Then the order r of $a \bmod N$ is the least power of a congruent to 1 mod$N$.

Note that, if $\gcd(a,N) \neq 1$, then no power of a can be congruent to 1 mod$N$, since $a^m-\lambda N$ is clearly divisible by $\gcd(a,N)$ for any m and λ.

Let $\pi(N)$ denote the number of primes less than or equal to N. The prime number theorem asserts that

$$\lim_{N\to\infty} \frac{\pi(N)}{N/\log N}=1. \tag{A5}$$

This theorem is difficult to prove (Hardy and Wright, 1965, Chap. 22), and various more refined estimates for the distribution of primes are also known (see Schroeder, 1990, Chap. 4, for an informal discussion.) We use Eq. (A5) loosely in the form

$$\pi(N)=N/\log N, \text{ for all sufficiently large } N. \tag{A6}$$

The precise statement is: For any $\epsilon>0$ there is an N_0 such that

$$\frac{N}{\log N}-\epsilon \leq \phi(N) \leq \frac{N}{\log N}+\epsilon \quad \text{for all } N>N_0. \tag{A7}$$

This may be used to make our arguments based on Eq. (A6) rigorous. Now clearly $\phi(N) \geq \pi(N)$, so

$$\phi(N) \geq N/\log N. \tag{A8}$$

Thus the probability $\phi(N)/N$ that a number chosen randomly from $1,\ldots,N$ is coprime to N is greater than $1/\log N$. We make much use of this result, even though the estimate, Eq. (A8), is rather weak—there are usually very many more numbers coprime to N than just the primes. Indeed it is shown by Hardy and Wright (1965, Sec. 18.4) that

$$\lim \inf \frac{\phi(N)}{N/\log\log N}=e^{-\gamma}, \tag{A9}$$

where γ is Euler's constant, so that, in the loose sense of Eq. (A6), $\phi(N)>e^{-\gamma}N/\log\log N$, which is much better than Eq. (A8).

4. Continued fractions

A (finite, simple) continued fraction (Hardy and Wright, 1965, Chap. 10) is an expression of the form

$$a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cfrac{1}{\cdots+\cfrac{1}{a_N}}}}} \tag{A10}$$

where $a_0,\ldots,a_N$ are positive integers. We abbreviate (A10) as $[a_0,\ldots,a_N]$ and define the nth convergent to this expression as $[a_0,\ldots,a_n]$ for n in the range $0 \leq n \leq N$.

If we write the nth convergent as p_n/q_n, then there is a recurrence relation

$$p_0=a_0, \quad p_1=a_1a_0+1, \quad p_n=a_np_{n-1}+p_{n-2},$$
$$q_0=1, \quad q_1=a_1, \quad q_n=a_nq_{n-1}+q_{n-2}. \tag{A11}$$

Furthermore, the convergents p_n/q_n computed by this iteration are always in their lowest terms, i.e., $\gcd(p_n,q_n)=1$ (Hardy and Wright, 1965, theorem 157).

Any (positive) rational number can be represented by a continued fraction computed efficiently as follows. Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Then $a_0=\lfloor x \rfloor$ and $x=a_0+\xi_0$ for some $0 \leq \xi_0<1$. If $\xi_0 \neq 0$, then $a_1=\lfloor 1/\xi_0 \rfloor$ and $1/\xi_0=a_1+\xi_1$ for some $0 \leq \xi_1<1$. If $\xi_1 \neq 0$, then $a_2=\lfloor 1/\xi_1 \rfloor$, etc. This process always terminates for rational x and we get $x=[a_0,\ldots,a_N]$ (Hardy and Wright, 1965, theorem 161).

From Eq. (A10) it is easy to see that $[a_0,\ldots,a_{N-1},a_N]=[a_0,\ldots,a_{N-1},a_N-1,1]$. If we impose $a_N>1$, then the representation of x as a continued fraction is unique.

Our main interest in continued fractions lies in the following result (Hardy and Wright, 1965, theorem 184, Sec. 10.15):

Theorem. Suppose that p/q is any rational number satisfying

$$\left|\frac{p}{q}-x\right|<\frac{1}{2q^2}.$$

Then p/q is a convergent of the continued fraction of x.

Continued fractions may also be used to represent irrational numbers, in which case they have infinite length $[a_0,a_1,\ldots]$. They are associated with a rich theory and a wide variety of applications (Schroeder, 1990, Chap. 5), e.g., they can efficiently provide excellent rational approximations to irrationals. The above theorem is also true for irrational x.

APPENDIX B: PROOF OF EQUATION (30)

Theorem. Let N be odd with prime factorization

$$N=p_1^{\alpha_1}p_2^{\alpha_2}\ldots p_k^{\alpha_k}. \tag{B1}$$

Suppose y is chosen at random, satisfying $\gcd(y,N)=1$. Let r be the order of y mod$N$. Then

$$\text{Prob}\,(r\text{ is even and }y^{r/2}\not\equiv\pm1\text{ mod}N)\geqslant1-\frac{1}{2^{k-1}}. \tag{B2}$$

Proof. We have $y^r\equiv1$ mod$N$, and r is the least such value. Thus we never have $y^{r/2}\equiv1$ mod$N$. We shall prove that

$$\text{Prob}\,(r\text{ is odd or }y^{r/2}\equiv-1\text{ mod}N)\leqslant\frac{1}{2^{k-1}}. \tag{B3}$$

Now $\gcd(y,p_i^{\alpha_i})\equiv1$, so let r_i be the order of y mod$p_i^{\alpha_i}$,

$$y^{r_i}\equiv1\text{ mod}p_i^{\alpha_i},\quad i=1,\ldots,k. \tag{B4}$$

Note that

$$r=\text{lcm}(r_1,\ldots,r_k). \tag{B5}$$

This follows because $y^r\equiv1$ mod$N$ implies that $y^r\equiv1$ mod$p_i^{\alpha_i}$, hence r is a multiple of each r_i, and r is the least such value, i.e., the least common multiple, giving Eq. (B5).

Next we have

$$y^{r/2}\equiv-1\text{ mod }N\text{ iff }y^{r/2}\equiv-1\text{ mod }p_i^{\alpha_i}\quad\text{for each }i \tag{B6}$$

The forward implication is clear, since $p_i^{\alpha_i}$ divides N. The reverse implication is given by the Chinese remainder theorem, for, if we put $\xi=y^{r/2}$, then the system of congruences $\xi\equiv-1$ mod$p_i^{\alpha_i}$ has a unique solution modulo$N$, and $\xi\equiv-1$ mod$N$ is a solution (by the forward implication); hence it is the unique solution, giving Eq. (B6).

Now write

$$r_i=s_i2^{t_i},\quad s_i\text{ odd }i=1,\ldots,k, \tag{B7}$$

where

$$s=\text{lcm}(s_1,\ldots,s_k), \tag{B8}$$

$$t=\max(t_1,\ldots,t_k). \tag{B9}$$

Then by Eq. (B2) $r=s2^t$. We can thus claim that

$$y^{r/2}\equiv-1\text{ mod}p_i^{\alpha_i}\quad\text{implies }t_i=t. \tag{B10}$$

To see this, assume that $y^{r/2}\equiv-1$ mod$p_i^{\alpha_i}$, but that $t_i<t$. Then by Eqs. (B7) and (B8) r_i divides $r/2$, but this is impossible, since by Eq. (B4) $y^{r_i}\equiv1$ mod$p_i^{\alpha_i}$, and $y^{r/2}\equiv-1$ mod$p_i^{\alpha_i}$ by assumption. [Note that relation (B10) fails for the excluded case $p_i^{\alpha_i}=2$, where 1 is congruent to -1.]

Next note that r is odd if and only if each r_i is odd [by Eq. (B5)], so that in this case all the t_i's are equal (to zero). Thus, combining Eqs. (B6) and (B10), we get

$$\text{Prob}(r\text{ is odd or }y^{r/2}\equiv-1\text{ mod}N)\leqslant\text{Prob}(\text{all }t_i\text{'s are equal}). \tag{B11}$$

To estimate the latter probability we use

$$\text{Prob}(t_i=j)\leqslant\frac{1}{2}\quad\text{for all }i\text{ and }j. \tag{B12}$$

To see this we use the fact that the multiplicative group modulo $p_i^{\alpha_i}$ is cyclic. The elements of this group are the integers coprime to $p_i^{\alpha_i}$, and so the size of the group is $\phi(p_i^{\alpha_i})$. Let g be a generator, so

$$g^{\phi(p_i^{\alpha_i})}\equiv1\text{ mod}p_i^{\alpha_i}. \tag{B13}$$

Write $\phi(p_i^{\alpha_i})=\sigma2^\tau$ with σ odd. By Euler's theorem any order modulo$p_i^{\alpha_i}$ divides $\sigma2^\tau$, so $t_i\leqslant\tau$ for all i. Now for any b odd let $\tilde\sigma2^{\tilde\tau}$ be the order of g^bmod$p_i^{\alpha_i}$, so $\tilde\tau\leqslant\tau$. Then

$$(g^b)^{\tilde\sigma2^{\tilde\tau}}=g^{b\tilde\sigma2^{\tilde\tau}}=1 \tag{B14}$$

and $b\tilde\sigma$ is odd, so $\tilde\tau=\tau$. Similarly, if b is even, then g^b has order $\tilde\sigma2^{\tilde\tau}$ with $\tilde\tau\leqslant\tau-1$.

Hence [except for the trivial case of $p_i^{\alpha_i}=2,\phi(2)=1$] we see that any prescribed value of $\tilde\tau$ can occur for at most half of the numbers coprime to $p_i^{\alpha_i}$. This proves Eq. (B12).

Finally we have

$$\text{Prob (all } t_i\text{'s equal)} = \sum_j \prod_{i=1}^{k} \text{Prob } (t_i = j) \tag{B15}$$

$$= \sum_j \text{Prob } (t_1 = j) \ldots \text{Prob } (t_k = j) \tag{B16}$$

$$\leqslant \sum_j \text{Prob } (t_1 = j) \frac{1}{2^{k-1}} = \frac{1}{2^{k-1}}. \tag{B17}$$

By Eq. (B11), this proves the theorem.

REFERENCES

Barenco, A., 1995, Proc. R. Soc. London, Ser. A **449**, 679.

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

Barenco, A., D. Deutsch, A. Ekert, and R. Jozsa, 1995, Phys. Rev. Lett. **74**, 4083.

Benioff, P., 1980, J. Stat. Phys. **22**, 563.

Benioff, P., 1986, Ann. New York Acad. Sci. (USA) **480**, 475.

Bennett, C. H., 1973, IBM J. Res. Dev. **17**, 525.

Bennett, C. H., 1982, Int. J. Theor. Phys. **21**, 905.

Bennett, C. H., 1989, SIAM J. Comput. **18(4)**, 766.

Bernstein, E., and U. Vazirani, 1993, in *Proceedings of the 25th ACM Symposium on the Theory of Computation* (ACM, New York), p. 11.

Berthiaume, A., and G. Brassard, 1994, J. Mod. Opt. **41**, 2521.

Berthiaume, A., D. Deutsch, and R. Jozsa, 1994, in *Proceedings of the Workshop on Physics and Computation—PhysComp '94*, edited by (IEEE Computer Society, Dallas, Texas), p. 60.

Brune, M., P. Nussenzveig, F. Schmidt-Kaler, F. Bernardot, A. Maali, J. M. Raimond, and S. Haroche, 1994, Phys. Rev. Lett. **72**, 3339.

Chuang, I., and R. Laflamme, 1995, "Quantum error correction by coding," preprint 9511003 available at LANL quant-ph archive.

Chuang, I., and Y. Yamamoto, 1995, "A simple quantum computer," Stanford University preprint.

Cirac, J. I., and P. Zoller, 1995, Phys. Rev. Lett. **74**, 4091.

Cleve, R., 1994, "A note on computing Fourier transformation by quantum programs," University of Calgary preprint.

Coppersmith, D., 1994, "An approximate Fourier transform useful in quantum factoring," IBM Research Report No. RC19642.

Davidovich, L., N. Zagury, M. Brune, J. M. Raimond, and S. Haroche, 1994, Phys. Rev. A **50**, R895.

Deutsch, D., 1985, Proc. R. Soc. London, Ser. A **400**, 97.

Deutsch, D., 1989, Proc. R. Soc. London, Ser. A **425**, 73.

Deutsch, D., 1994, unpublished.

Deutsch, D., A. Barenco, and A. Ekert, 1995, Proc. R. Soc. London, Ser. A **449**, 669.

Deutsch, D., and R. Jozsa, 1992, Proc. R. Soc. London, Ser. A **439**, 553.

DiVincenzo, D. P., 1995a, Phys. Rev. A **51**, 1015.

DiVincenzo, D. P., 1995b, Science **270**, 255.

Dunne, P. E., 1988, *The Complexity of Boolean Networks* (Academic, New York).

Feynman, R., 1982, Int. J. Theor. Phys. **21**, 467.

Feynman, R., 1985, Opt. News **11**, 11 (reprinted in Found. Phys. 1986 **16**, 507).

Hardy, G. H., and E. M. Wright, 1965, *An Introduction to the Theory of Numbers* (4th edition Clarendon, Oxford).

Hecht, J., 1992, *The Laser Guidebook* (McGraw-Hill, New York).

Jozsa, R., 1991, Proc. R. Soc. London, Ser. A **435**, 563.

Jozsa, R., 1992, in *Proceedings of Workshop on Physics and Computation—PhysComp '92*, edited by (IEEE Computer Society, Dallas, Texas), p. 192.

Knuth, D. E., 1981, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (Addison-Wesley, Reading, MA).

Landauer, R., 1961, IBM J. Res. Dev. **5**, 183.

Landauer, R., 1987, Phys. Scr. **35**, 88.

Landauer, R., 1991, Phys. Today **44**, 23.

Lecerf, Y., 1963, C. R. Acad. Sci. Ser. A **257**, 2597.

Lenstra, A. K., H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, 1990, in *Proceedings of the 22nd ACM Symposium on the Theory of Computing* (ACM, New York), p. 564.

Lloyd, S., 1993a, Phys. Rev. Lett. **71**, 943.

Lloyd, S., 1993b, Science **261**, 1569.

Lloyd, S., 1994, J. Mod. Opt. **41**, 2503.

Lloyd, S., 1995a, Phys. Rev. Lett. **74**, 346.

Lloyd, S., 1995b, Sci. Am. **273**, 44.

Margolus, N., 1986, Ann. New York Acad. Sci. (USA) **480**, 487.

Margolus, N., 1991, in *Complexity, Entropy, and the Physics of Information*, edited by W. H. Zurek, Santa Fe Institute Series Vol. 8 (Addison-Wesley, Redwood City, CA), p. 273.

Monroe, C., D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, 1995, Phys. Rev. Lett. **50**, 4714.

Obermayer, K., W. G. Teich, and G. Mahler, 1988, Phys. Rev. B **37**, 8096.

Palma, G. M., K.-A. Suominen, and A. Ekert, 1996, Proc. R. Soc. London, Ser. A **452**, 567.

Papadimitriou, C. H., 1994, *Computational Complexity* (Addison-Wesley, Reading, MA).

Parkins, A. S., P. Marte, P. Zoller, and H. J. Kimble, 1993, Phys. Rev. Lett. **71**, 3095.

Penrose, R., 1989, *The Emperor's New Mind* (Oxford University, New York).

Rabin, M. O., 1980, J. Num. Theory **12**, 128.

Raimond, J. M., M. Brune, J. Lepape, and S. Haroche, 1989, in *Laser Spectroscopy IX*, edited by M. S. Feld, J. E. Thomas, and A. Mooradian (Academic, New York), p. 140.

Riesel, H., 1985, *Prime Numbers and Computer Methods for Factorization* (Birkhäuser, Boston, MA), p. 156.

Rivest, R., A. Shamir, and L. Adleman, January 1979, "On digital signatures and public-key cryptosystems," MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212.

Schroeder, M. R., 1990, *Number Theory in Science and Communication*, 2nd enlarged edition (Springer, New York).

Schumacher, B., 1995, Phys. Rev. A **51**, p. 2738.

Shor, P. W., 1994, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA), p. 124. Expanded version of this paper is available at LANL quant-ph archive, preprint 9508027.

Shor, P. W., 1995, Phys. Rev. A **52**, R2493.

Silverman, R. D., 1987, Math. Comput. **48**, 329.

Simon, D. S., 1994, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA), p. 116.

Sleator, T., and H. Weinfurter, 1995, Phys. Rev. Lett. **74**, 4087.

Teich, W. G., K. Obermayer, and G. Mahler, 1988, Phys. Rev. B **37**, 8111.

Turchette, Q. A., C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble, 1995, Phys. Rev. Lett. **75**, 4710.

Unruh, W., 1994, Phys. Rev. A **51**, 992.

Vedral, V., A. Barenco, and A. Ekert, 1996, Phys. Rev. A **54**, 147.

Welsh, D., 1988, *Codes and Cryptography* (Clarendon, Oxford).

Yao, A., 1993, in *Proceedings of the 34th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA), p. 352.

Zurek, W. H., 1991, Phys. Today, October, p. 36.