THE BULLETIN OF SYMBOLIC LOGIC
Volume 6, Number 3, Sept. 2000

MACHINES, LOGIC AND QUANTUM PHYSICS

DAVID DEUTSCH, ARTUR EKERT, AND ROSSELLA LUPACCHINI

§1. Mathematics and the physical world. Genuine scientific knowledge cannot be certain, nor can it be justified a priori. Instead, it must be conjectured, and then tested by experiment, and this requires it to be expressed in a language appropriate for making precise, empirically testable predictions. That language is mathematics.

This in turn constitutes a statement about what the physical world must be like if science, thus conceived, is to be possible. As Galileo put it, "the universe is written in the language of mathematics" [7]. Galileo's introduction of mathematically formulated, testable theories into physics marked the transition from the Aristotelian conception of physics, resting on supposedly necessary a priori principles, to its modern status as a theoretical, conjectural and empirical science. Instead of seeking an infallible universal mathematical design, Galilean science uses mathematics to express quantitative descriptions of an objective physical reality. Thus mathematics became the language in which we express our knowledge of the physical world — a language that is not only extraordinarily powerful and precise, but also effective in practice. Eugene Wigner referred to "the unreasonable effectiveness of mathematics in the physical sciences"[17]. But is this effectiveness really unreasonable or miraculous?

Numbers, sets, groups and algebras have an autonomous reality quite independent of what the laws of physics decree, and the properties of these mathematical structures can be just as objective as Plato believed they were (and as Roger Penrose now advocates [11]). But they are revealed to us only through the physical world. It is only physical objects, such as computers or human brains, that ever give us glimpses of the abstract world of mathematics. But how? It is a familiar fact, and has been since the prehistoric beginnings of mathematics, that simple physical systems like fingers, tally sticks and the abacus can be used to represent some mathematical entities and operations. Historically the operations of elementary arithmetic were also the first to be delegated to more complex machines. As soon as it

Received November 8, 1999; revised June 26, 2000.

became clear that additions and multiplications can be performed by a sequence of basic procedures and that these procedures are implementable as physical operations, mechanical devices designed by Blaise Pascal, Gottfried Wilhelm Leibniz and others began to relieve humans from tedious tasks such as multiplying two large integers [9]. In the twentieth century, following this conquest of arithmetic, the logical concept of computability was likewise "mechanised": Turing machines were invented in order to formalise the notion of "effectiveness" inherent in the intuitive idea of computability or calculability. Alan Turing conjectured that the theoretical machines in terms of which he defined computation are capable of performing any *finite, effective* procedure (algorithm). It is worth noting that Turing machines were intended to reproduce all operations that a *human* computer could perform, following preassigned instructions. Turing's method was to think in terms of physical operations, and imagine that every operation performed by the computer "consists of some change of the physical system consisting of the computer and his tape" [16]. The key point is that since the outcome is not affected by constructing "a machine to do the work of this computer", the effectiveness of a human computer can be mimicked by a logic machine — just as the human ability to perform arithmetic is mimicked by mechanical calculating machines.

The Turing machine was an abstract construct, but thanks to subsequent developments in computer science and engineering, algorithms can now be performed by real automatic computing machines. The natural question now arises: what, precisely, is the set of logical procedures that can be performed by a physical device? The theory of Turing machines cannot, even in principle, answer this question, nor can any approach based on formalising traditional notions of effective procedures. What we need instead is to extend Turing's idea of *mechanising* procedures, in particular, the procedures involved in the notion of derivability. This would define mathematical proofs as being mechanically reproducible and to that extent effectively verifiable. The universality and reliability of logical procedures would be guaranteed by the mechanical procedures that effectively perform logical operations — but by no more than that. But what does it mean to involve real, physical machines in the definition of a logical notion? and what might this imply in return about the 'reasonableness' or otherwise of the effectiveness of physics in the mathematical sciences?

While the abstract model of a machine, as used in the classical theory of computation, is a pure-mathematical construct to which we can attribute any consistent properties we may find convenient or pleasing, a consideration of actual computing machines as physical objects must take account of their actual physical properties, and therefore, in particular, of the laws of physics. Turing's machines (with arbitrarily long tapes) can be built, but no one would ever do so except for fun, as they would be extremely slow and cumbersome. The computers available today are much faster and more reliable. Where does this reliability come from? How do we know that the computer generates the same outputs as the appropriate abstract Turing machine, that the machinery of cog-wheels or electric currents must finally display the right answer? After all, nobody has tested the machine by following all possible logical steps, or by performing all the arithmetic it can perform. If they were able and willing to do that, there would be no need to build the computer in the first place. The reason we trust the machine cannot be based entirely on logic; it must also involve our knowledge of the physics of the machine. When relying on the machine's output, we rely on our knowledge of the laws of physics that govern the computation, i.e. the physical process that takes the machine from an initial state (input) to a final state (output). This knowledge, though formulated in mathematical terms in the tradition of Galileo, evolved by conjectures and empirical refutations. From this perspective, what Turing really asserted was that it is possible to *build* a universal computer, a machine that can be programmed to perform any computation that any other physical object can perform. That is to say, a single buildable physical object, given only proper maintenance and a supply of energy and additional memory when needed, can predict or simulate all the behavior and responses of any other physically possible object or process. In this form, Turing's conjecture (which Deutsch has called in this context the *Church-Turing Principle* [3]) can be viewed as a statement about the physical world.

Are there any limits to the computations that can be performed by computing machines? Obviously there are both logical and physical limits. Logic tells us that, for example, no machine can find more than one even prime, whilst physics tells us that, for example, no computations can violate the laws of thermodynamics. Moreover, logical and physical limitations can be intimately linked, as illustrated by the "halting problem". As a matter of logic, Turing's halting theorem says that there is no Turing-machine algorithm for deciding whether any given machine, when started from any given initial situation, eventually stops. Therefore some computational problems, such as determining whether a specified universal Turing machine, given a specified input, will halt, cannot be solved by any Turing machine. But to obtain from this theorem a statement that machines with certain properties cannot be physically built, one needs physical interpretations of the abstractions referred to by the theorem. For instance, the notion of *finiteness* in mathematics (a finite set being one that cannot be placed into one-one correspondence with part of itself) is inadequate to the task of telling us which physical processes are finite and which infinite. For example, searching a list of all the integers is a physically infinite process, yet walking across a room, thereby passing through an infinity of distinct points far more numerous than the integers, is a physically *finite* process. It is only physics that can tell the difference, and that processes of the latter type cannot ever be harnessed to perform a search of all the integers. And conversely, when we interpret Turing's theorem as a statement about what can and cannot be computed in physical fact, we are adopting some of his tacit assumptions about physical reality or equivalently about the laws of physics.

So where does mathematical effectiveness come from? It is not simply a miracle, "a wonderful gift which we neither understand nor deserve" [17] — at least, no more so than our ability to discover *empirical* knowledge, for our knowledge of mathematics and logic is inextricably entangled with our knowledge of physical reality: every mathematical proof depends for its acceptance upon our agreement about the rules that govern the behavior of physical objects such as computers or our brains. Hence when we improve our knowledge about physical reality, we may also gain new means of improving our knowledge of logic, mathematics and formal constructs. It seems that we have no choice but to recognize the dependence of our mathematical *knowledge* (though not, we stress, of mathematical truth itself) on physics, and that being so, it is time to abandon the classical view of computation as a purely logical notion independent of that of computation as a physical process. In the following we discuss how the discovery of quantum mechanics in particular has changed our understanding of the nature of computation.

§2. Quantum interference. To explain what makes quantum computers so different from their classical counterparts, we begin with the phenomenon of quantum interference. Consider the following computing machine whose input can be prepared in one of two states representing, 0 and 1.

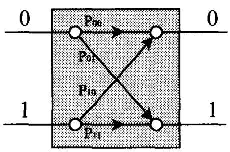

FIGURE 1. Schematic representation of the most general machine that performs a computation mapping $\{0, 1\}$ to itself. Here p_{ij} is the probability for the machine to produce the output j when presented with the input i. (The action of the machine depends on no other input or stored information.)

The machine has the property that if we prepare its input with the value a ($a = 0$ or 1) and then measure the output, we obtain, with probability p_{ab}, the value b ($b = 0$ or 1). It may seem obvious that if the p_{ab} are arbitrary apart from satisfying the standard probability conditions $\sum_b p_{ab} = 1$, Fig. (1) represents *the most general* machine whose action depends on no other input or stored information and which performs a computation mapping $\{0,1\}$ to itself. The two possible deterministic limits are obtained by setting $p_{01} = p_{10} = 0$, $p_{00} = p_{11} = 1$ (which gives a logical identity machine) or $p_{01} = p_{10} = 1$, $p_{00} = p_{11} = 0$ (which gives a negation ('**not**') machine). Otherwise we have a randomising device. Let us assume, for the sake of illustration, that $p_{01} = p_{10} = p_{00} = p_{11} = 0.5$. Again, we may be tempted to think of such a machine as a random switch which, with equal probability, transforms any input into one of the two possible outputs. However, that is not necessarily the case. When the particular machine we are thinking of is followed by another, identical, machine the output is always the negation of the input.

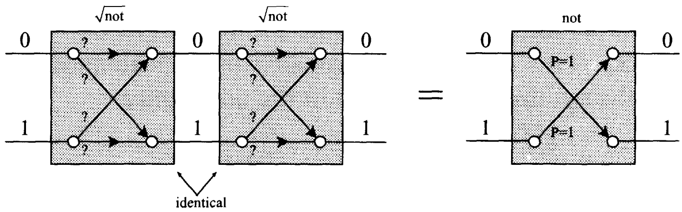

FIGURE 2. Concatenation of the two identical machines mapping $\{0,1\}$ to itself. Each machine, when tested separately, behaves as a random switch, however, when the two machines are concatenated the randomness disappears — the net effect is the logical operation **not**. This is in clear contradiction with the axiom of additivity in probability theory!

This is a very counter-intuitive claim — the machine alone outputs 0 or 1 with equal probability and independently of the input, but the two machines, one after another, acting independently, implement the logical operation **not**. That is why we call this machine $\sqrt{\textbf{\textit{not}}}$. It may seem reasonable to argue that since there is no such operation in logic, $\sqrt{\textbf{\textit{not}}}$ machines cannot exist. But they do exist! Physicists studying single-particle interference routinely construct them, and some of them are as simple as a half-silvered mirror i.e. a mirror which with probability 50% reflects a photon that impinges upon it and with probability 50% allows it to pass through. Thus the two concatenated machines are realised as a sequence of two half-silvered mirrors with a photon in each denoting 0 if it is on one of the two possible paths and 1 if it is on the other.

The reader may be wondering what has happened to the axiom of additivity in probability theory, which says that if E_1 and E_2 are mutually exclusive events then the probability of the event (E_1 **or** E_2) is the sum of the probabilities of the constituent events, E_1, E_2. We may argue that the transition

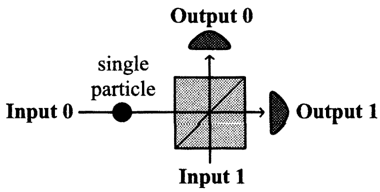

FIGURE 3. The experimental realisation of the $\sqrt{\boldsymbol{not}}$ gate. A half-silvered mirror reflects half the light that impinges upon it. But a single photon doesn't split: when we send a photon at such a mirror it is detected, with equal probability, either at Output 0 or 1. This does not, however, mean that the photon leaves the mirror in the direction of either of the two outputs at random. In fact the photon takes both paths at once! This can be demonstrated by concatenating two half-silvered mirrors as shown in the next figure.

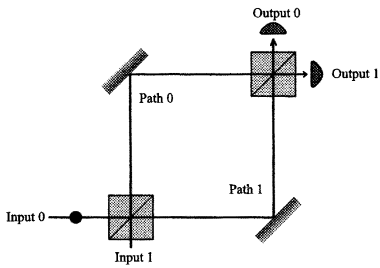

FIGURE 4. Experimental realisation of two concatenated $\sqrt{\boldsymbol{not}}$ gates, known as a single-particle interference experiment. A photon which enters the interferometer via Input 0 always strikes a detector at Output 1 and never a detector at Output 0. Any explanation which assumes that the photon takes exactly one path through the interferometer leads to the conclusion that the two detectors should on average each fire on half the occasions when the experiment is performed. But experiment shows otherwise!

$0 \to 0$ in the composite machine can happen in two mutually exclusive ways, namely, $0 \to 0 \to 0$ or $0 \to 1 \to 0$. The probabilities of the two are $p_{00}p_{00}$ and $p_{01}p_{10}$ respectively. Thus the sum $p_{00}p_{00} + p_{01}p_{10}$ represents the probability of the $0 \to 0$ transition in the new machine. Provided that p_{00} or $p_{01}p_{10}$ are different from zero, this probability should also be different from zero. Yet we can build machines in which p_{00} and $p_{01}p_{10}$ are different from zero, but the probability of the $0 \to 0$ transition in the composite machine is equal to zero. So what is wrong with the above argument?

One thing that is wrong is the assumption that the processes $0 \to 0 \to 0$ and $0 \to 1 \to 0$ are mutually exclusive. In reality, the two transitions *both occur*, simultaneously. We cannot learn about this fact from probability theory or any other a priori mathematical construct. We learn it from the best physical theory available at present, namely quantum mechanics. Quantum theory explains the behavior of $\sqrt{\boldsymbol{not}}$ and correctly predicts the probabilities of all the possible outputs no matter how we concatenate the machines. This knowledge was created as the result of conjectures, experimentation, and refutations. Hence, reassured by the physical experiments that corroborate this theory, logicians are now entitled to propose a new logical operation $\sqrt{\boldsymbol{not}}$. Why? Because a faithful physical model for it exists in nature!

Let us now introduce some of the mathematical machinery of quantum mechanics which can be used to describe quantum computing machines ranging from the simplest, such as $\sqrt{\boldsymbol{not}}$, to the quantum generalisation of the universal Turing machine. At the level of predictions, quantum mechanics introduces the concept of *probability amplitudes* — complex numbers c such that the quantities $|c|^2$ may under suitable circumstances be interpreted as probabilities. When a transition, such as "a machine composed of two identical sub-machines starts in state 0 and generates output 0, and affects nothing else in the process", can occur in several alternative ways, the overall probability amplitude for the transition is the sum, not of the probabilities, but of the probability amplitudes for each of the constituent transitions considered separately.

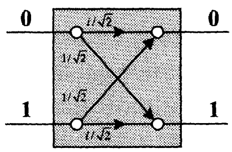

FIGURE 5. Transitions in quantum machines are described by probability amplitudes rather than probabilities. Probability amplitudes are complex numbers c such that the quantities $|c|^2$ may under suitable circumstances be interpreted as probabilities. When a transition, such as "a machine composed of two identical sub-machines starts in state 0 and generates output 0, and affects nothing else in the process", can occur in several alternative ways, the probability amplitude for the transition is the sum of the probability amplitudes for each of the constituent transitions considered separately.

In the $\sqrt{\boldsymbol{not}}$ machine, the probability amplitudes of the $0 \to 0$ and $1 \to 1$ transitions are both $i/\sqrt{2}$, and the probability amplitudes of the $0 \to 1$

and $1 \to 0$ transitions are both $1/\sqrt{2}$. This means that the $\sqrt{\textbf{\textit{not}}}$ machine preserves the bit value with probability amplitude $c_{00} = c_{11} = i/\sqrt{2}$ and negates it with probability amplitude $c_{01} = c_{10} = 1/\sqrt{2}$. In order to obtain the corresponding probabilities we have to take the modulus squared of the probability amplitudes, which gives probability $1/2$ both for preserving and swapping the bit value. This describes the behavior of the single $\sqrt{\textbf{\textit{not}}}$ machine in Fig. (1). When we concatenate the two machines, as in Fig. (2) then, in order to calculate the probability of output 0 from input 0, we have to add the probability amplitudes of all computational paths leading from input 0 to output 0. There are only two of them — $c_{00}c_{00}$ and $c_{01}c_{10}$. The first computational path has probability amplitude $i/\sqrt{2} \times i/\sqrt{2} = -1/2$ and the second one $1/\sqrt{2} \times 1/\sqrt{2} = +1/2$. We add the two probability amplitudes first and then we take the modulus squared of the sum. We find that the probability of output 0 is zero. Unlike probabilities, probability amplitudes can cancel each other out!

§3. **Quantum algorithms.** Addition of probability amplitudes, rather then probabilities, is one of the fundamental rules for prediction in quantum mechanics and applies to all physical objects, in particular quantum computing machines. If a computing machine starts in a specific initial configuration (input) then the probability that after its evolution via a sequence of intermediate configurations it ends up in a specific final configuration (output) is the squared modulus of the sum of all the probability amplitudes of the computational paths that connect the input with the output. The amplitudes are complex numbers and may cancel each other, which is referred to as *destructive interference*, or enhance each other, referred to as *constructive interference*. The basic idea of quantum computation is to use quantum interference to amplify the correct outcomes and to suppress the incorrect outcomes of computations. Let us illustrate this by describing a variant of the first quantum algorithm, proposed by David Deutsch in 1985.

Consider the Boolean functions f that map $\{0, 1\}$ to $\{0, 1\}$. There are exactly four such functions: two constant functions ($f(0) = f(1) = 0$ and $f(0) = f(1) = 1$) and two "balanced" functions ($f(0) = 0, f(1) = 1$ and $f(0) = 1, f(1) = 0$). Suppose we are allowed to evaluate the function f *only once* (given, say, a lengthy algorithm for evaluating it on a given input, or a look-up table that may be consulted only once) and asked to determine whether f is constant or balanced (in other words, whether $f(0)$ and $f(1)$ are the same or different). Note that we are not asking for the particular values $f(0)$ and $f(1)$ but for a global property of the function f. Our classical intuition insists, and the classical theory of computation confirms, that to determine this global property of f, we have to evaluate both $f(0)$ and $f(1)$, which involves evaluating f twice. Yet this is simply not true in physical reality, where quantum computation can solve Deutsch's problem with a single function evaluation. The machine that solves the problem, using quantum interference, is composed of two $\sqrt{\boldsymbol{not}}$s with the function evaluation machine in between them, as in Fig. (6).

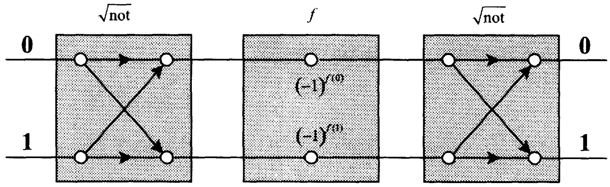

FIGURE 6. Schematic representation of a quantum machine that solves Deutsch's problem with a single function evaluation.

We need not go into the technicalities of the physical implementation of the evaluation of $f(x)$, where f is in general a Boolean function mapping $\{0,1\}^n \rightarrow \{0,1\}^m$. But generally, a machine that evaluates such a function must be capable of traversing as many computational paths as there are possible values x in the domain of f (so we can label them with x). Its effect is that the probability amplitude on path x is multiplied by the phase factor $\exp\left(\frac{2\pi i f(x)}{2^m}\right)$, where $f(x)$ is an m-bit number rather than an m-tuple [2]. In the case of Deutsch's problem the two phase factors are $(-1)^{f(0)}$ and $(-1)^{f(1)}$. Now we can calculate the probability amplitude of output 0 on input 0. The probability amplitudes on the two different computational paths are $i/\sqrt{2}\times(-1)^{f(0)}\times i/\sqrt{2} = -1/2\times(-1)^{f(0)}$ and $1/\sqrt{2}\times(-1)^{f(1)}\times 1/\sqrt{2} = 1/2 \times (-1)^{f(1)}$. Their sum is

$$\frac{1}{2}\left((-1)^{f(1)} - (-1)^{f(0)}\right), \tag{1}$$

which is 0 when f is constant and ± 1 when f is balanced. Thus the probability of output 0 on input 0 is given by the modulus squared of the expression above, which is zero when f is constant and unity when f is balanced.

Deutsch's results laid the foundation for the new field of quantum computation, and the hunt began for useful tasks for quantum computers to do. A sequence of steadily improving quantum algorithms led in 1994 to Peter Shor's discovery of a quantum algorithm that could perform efficient factorisation [13]. Since the intractability of factorisation underpins the security of many of the most secure known methods of encryption, including the most popular public key cryptosystem RSA [12] [1], Shor's algorithm

[1] In December 1997 the British Government officially confirmed that public-key cryptography was originally invented at the Government Communications Headquarters (GCHQ)

was soon hailed as the first 'killer application' for quantum computation — something very useful that only a quantum computer could do.

Shor's quantum algorithm for factoring an integer N is based on calculating the period of the function $a^x \bmod N$ where a is a randomly chosen integer that has no common factor with N. The period r is called the *order* of a modulo N. If r is even, then we have:

$$a^r = 1 \bmod N \tag{2}$$

$$\Leftrightarrow \quad (a^{r/2})^2 - 1^2 = 0 \bmod N \tag{3}$$

$$\Leftrightarrow \quad (a^{r/2} - 1)(a^{r/2} + 1) = 0 \bmod N. \tag{4}$$

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

For example, suppose we want to factor 15 using this method. Let $a = 11$. As x varies, the function $11^x \bmod 15$ forms a repeating sequence $1, 11, 1, 11, 1, 11, \ldots$, so its period is $r = 2$, and $a^{r/2} \bmod 15 = 11$. Then we take the greatest common divisor of 10 and 15, and of 12 and 15 which are respectively 5 and 3, the two factors of 15.

Obtaining r by classical computation is at least as hard as trying to factor N; the time required is thought to grow exponentially with the number of digits of N. But quantum computers can find r very efficiently. This is done, again, by a machine that can enter several (N^2 or more) computational paths simultaneously and modify the probability amplitudes of each path by a phase factor of the form $e^{2\pi i m/r}$, where m is a positive integer. Bringing the computational paths together produces interference which allows r to be estimated with a low probability of error. (Readers interested in the technical details will find a comprehensive analysis of Shor's algorithm in [13, 5, 2].)

Most complexity theorists believe the factoring problem to be outside the "**BPP**" class (where **BPP** stands for "bounded error probabilistic polynomial time"), though interestingly, this has not yet been proved. 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" (see, for example, [10]). A '**BPP** class algorithm' for solving a problem is an efficient algorithm which, for any input, provides an answer that is correct with a probability greater than some constant $\delta > 1/2$. In general we cannot check easily if the answer is correct or not but we may repeat the computation some fixed number k times and then take a majority vote of all the k answers. For sufficiently large

---

in Cheltenham. By 1975, James Ellis, Clifford Cocks, and Malcolm Williamson from GCHQ had discovered what was later re-discovered in academia and became known as RSA and Diffie-Hellman key exchange.

k the majority answer will be correct with probability as close to 1 as desired. Now, Shor's result proves that factoring is not in reality an intractable task — and we learned this by studying quantum mechanics!

Richard Feynman, in his talk during the First Conference on the Physics of Computation held at MIT in 1981, observed that it appears to be impossible to simulate a general quantum evolution on a classical probabilistic computer in an efficient way [6]. That is to say, any classical simulation of quantum evolution involves an exponential slowdown in time compared with the natural evolution, since the amount of information required to describe the evolving quantum state in classical terms generally grows exponentially with time. However, instead of viewing this fact as an obstacle, Feynman regarded it as an opportunity. If it requires so much computation to work out what will happen in a multiparticle interference experiment then, he argued, the very act of setting up such an experiment and measuring the outcome is tantamount to performing a complex computation. Thus, Feynman suggested that it might be possible to simulate a quantum evolution efficiently after all, provided that the simulator itself is a quantum mechanical device. Furthermore, he conjectured that if one wanted to simulate a different quantum evolution, it would not be necessary to construct a new simulator from scratch. It should be possible to choose the simulator so that systematic modifications of it would suffice to give it any desired interference properties. He called such a device a universal quantum simulator. In 1985 Deutsch proved that such a universal simulator or a universal quantum computer does exist and that it could perform any computation that any other quantum computer (or any Turing-type computer) could perform [3]. Moreover, it has since been shown that the time and other resources that it would need to do these things would not increase exponentially with the size or detail of the physical system being simulated, so the relevant computations would be tractable by the standards of complexity theory [1]. This illustrates the fact that the more we know about physics, the more we know about computation and mathematics. Quantum mechanics proves that factoring is tractable: without quantum mechanics we do not yet know how to settle the issue either way.

The factorisation problem is unusual among classically intractable problems in that once one has a solution, one can efficiently prove by classical means that it *is* a solution. For most intractable problems, proving that it is a solution is intractable too. A rich source of such problems has been the area of combinatorial games e.g. deciding whether or not the first player in a game such as Go or Chess on $l \times l$ board has a winning strategy.

§4. Deterministic, probabilistic, and quantum computers. Any quantum computer, including the universal one, can be described in a fashion similar to the special-purpose machines we have described above, essentially by replacing probabilities by probability amplitudes. Let us start with a classical Turing machine. This is defined by a finite set of quintuples of the form

$$(q, s, q', s', d), \tag{5}$$

where the first two characters describe the initial condition at the beginning of a computational step and the remaining three characters describe the effect of the instruction to be executed in that condition (q is the current state, s is the symbol currently scanned, q' is the state to enter next, s' is the symbol to replace s, and d indicates motion of one square to the right, or one square to the left, or stay fixed, relative to the tape). In this language a computation consists of presenting the machine with an input which is a finite string of symbols from the alphabet Σ written in the tape cells, 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 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.

During a 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, state q_0, and the head scanning the leftmost symbol from the input. There are infinitely many possible configurations of the machine but in each successful computation the machine goes through only a finite sequence of them. The transitions between configurations are completely determined by the quintuples (5).

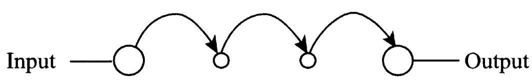

FIGURE 7. A three step deterministic computation.

Computations do not, in principle, have to be deterministic. Indeed, we can augment a Turing machine by allowing it "to toss an unbiased coin" and to choose its steps randomly. Such a probabilistic computation can be viewed as a directed, tree-like graph where each node corresponds to a configuration of the machine, and each edge represents one step of the computation. The computation starts from the root node representing the initial configuration and it subsequently branches into other nodes representing configurations reachable with non-zero probability from the initial configuration. The action of the machine is completely specified by a finite description of the form:

$$(6) \qquad \delta: \quad Q \times \Sigma \times Q \times \Sigma \times \{\text{Left}, \text{Right}, \text{Nothing}\} \longmapsto [0,1],$$

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. If we associate with each edge of the graph the probability that the computation follows that edge then we must require that the sum of the probabilities on edges leaving any single node is always equal to 1. 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, and the probability of a particular configuration 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 arbitrarily high probability less than 1) much faster than any known deterministic algorithms.

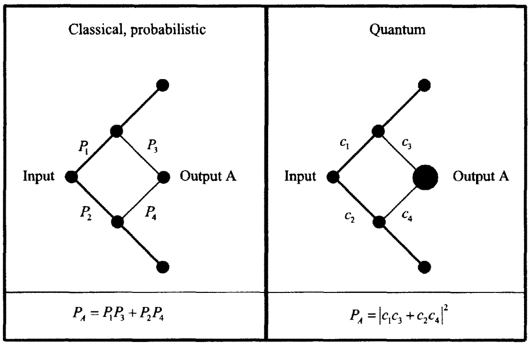

FIGURE 8. The probabilistic Turing machine (left) — the probability of output A is the sum of the probabilities of all computations leading to output A. In the quantum Turing machine (on the right) the probability of output A is obtained by adding all probability amplitudes leading from the initial state to output A and then taking the squared modulus of the sum. In the quantum case the probabilities of some outcomes can be enhanced (constructive interference) or suppressed (destructive interference) compared with what classical probability theory would permit.

The classical model described above suggests a natural quantum generalisation. A 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 being followed is the product of the probability amplitudes along the path's edges and the probability amplitude of a particular configuration is the sum of the amplitudes along all possible paths leading from the root to that configuration. If a particular final configuration can be reached via two different paths with amplitudes c and $-c$ then the probability of reaching that configuration is $|c - c|^2 = 0$ despite the fact that the probability for the computation to follow either of the two paths separately is $|c|^2$ in both cases. Furthermore a single quantum computer can follow many distinct computational paths simultaneously 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. The action of any such quantum machine is completely specified by a finite description of the form

$$(7) \qquad \delta : \quad Q \times \Sigma \times Q \times \Sigma \times \{\text{Left}, \text{Right}, \text{Nothing}\} \longmapsto \boldsymbol{C}$$

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.

§5. Qubits, superpositions and decoherence. So far we have only mentioned a possible experimental realisation of the $\sqrt{\boldsymbol{not}}$ operation, using half-silvered mirrors. That is a 'single-qubit' gate, a *qubit* being the quantum analogue of a bit, the basic unit of information. Fully-fledged quantum computers require two-qubit gates, and a means of composing them into arbitrary quantum computational networks (the analogue of classical logic networks).

In principle we know how to build such devices. From a physical point of view a bit is a two-state system: it can be prepared in one of two distinguishable states representing two logical values — yes or no, true or false, or simply 1 or 0. For example, in digital computers, the voltage between the plates of a capacitor can represent a bit of information: a charge on the capacitor denotes 1 and the absence of charge denotes 0. One bit of information can be also encoded using, for instance, two different polarisations of light or two different electronic states of an atom. Now, quantum mechanics tells us that if a bit can exist in either of two distinguishable states, it can also exist in *coherent superpositions* of them. These are further states, which in general have no classical analogues, in which the atom represents *both* values, 0 and 1, simultaneously, just as the photon may take both exits 0 and 1 from the $\sqrt{\boldsymbol{not}}$ machine.

One qubit can store both 0 and 1 in a superposition. A quantum register composed of, say, three qubits can simultaneously store up to eight numbers in a quantum superposition. It is quite remarkable that eight different numbers can be physically present in the same register; but it should be no more surprising than the numbers 0 and 1 both being present in the same qubit. If we add more qubits to the register its capacity for storing numbers in this sense increases exponentially: four qubits can store 16 different numbers at once, and in general L qubits can store up to 2^L numbers at once.

Once the register is prepared in a superposition of many different numbers, we can perform mathematical operations on all of them at once. For example, if the qubits are atoms then suitably tuned laser pulses affect their electronic states in ways that depend on those of their neighbours, thus implementing quantum computational gates. Repeated pulses have the effect of composing successive gate operations in the manner of a quantum computational network which evolves initial superpositions of encoded numbers into different superpositions. During such an evolution each number in the superposition is affected, so we are performing a massive parallel computation. Thus a quantum computer can in a single computational step perform the same mathematical operation on, say, 2^L different input numbers, and the result will be a superposition of all the corresponding outputs. In order to accomplish the same task any classical computer has to repeat the computation 2^L times, or has to use 2^L different processors working in parallel.

So if we know how to build quantum computers, why don't we have any on our desks? As the number of qubits increases, we quickly run into some serious practical problems. The more interacting qubits are involved, the harder it tends to be to engineer the interactions that would cause the necessary gate operations without introducing errors. Apart from the technical difficulties of working with individual atoms and photons, the most important problem facing all potential quantum computing technologies is that of preventing the surrounding environment from being affected by the interactions that generate quantum superpositions. The more components there are, the more likely it is that quantum information will spread outside the quantum computer and be lost into the environment, thus spoiling the computation. This process is called decoherence. Thus we need to engineer sub-microscopic systems in which qubits affect each other but not the environment. Will this be ever possible?

There are now excellent theoretical reasons for believing that it will [14]. It has been proved that if the decoherence satisfies some reasonably plausible assumptions — for instance that it occurs independently on each of the qubits, and if the performance of gate operations on some qubits does not cause decoherence in *other* qubits, then fault-tolerant quantum computation is possible. Still, no one is expecting the formidable technical problems to be solved in one fell swoop. Simple quantum logic gates involving two qubits have been realised in laboratories in Europe and U.S.A. The next decade should bring control over several qubits and we shall then begin to benefit from our new way of harnessing nature.

§6. Deeper implications. When the physics of computation was first investigated, starting in the 1960s, one of the main motivations was a fear that quantum-mechanical effects might place fundamental bounds on the accuracy with which physical objects could render the properties of the abstract entities, such as logical variables and operations, that appear in the theory of computation. Thus it was feared that the power and elegance of that theory, its most significant concepts such as computational universality, its fundamental principles such as the Church-Turing thesis and its powerful results such as the more modern theory of complexity, might all be mere figments of pure mathematics, not really relevant to anything in nature. But it turned out that quantum mechanics, far from placing restrictions on which Turing computations can be performed in nature, permits them all, and in addition provides new modes of computation such as those we have described.

The quantum analogue of the Church-Turing thesis (that all 'natural' models of computation are essentially equivalent to each other) is the Church-Turing Principle (that the universal quantum computer can simulate the behavior of any finite physical system); the latter was straightforwardly proved in Deutsch's 1985 paper [3]. A stronger result (also conjectured but never proved in the classical case), namely that such simulations can always be performed in a time that is at most a polynomial function of the time taken for the physical evolution, has since been proved in the quantum case [1].

What made these results possible was the narrowing of the Church-Turing thesis ("what is effectively calculable can be computed by a Turing machine") by interpreting "what is effectively calculable" as what may in principle be computed by a real physical system. It is interesting to contrast this with Gandy's [8] attempt to disentangle Turing's definition of computability from its appeal to a human computer. Gandy attempted to reach a *higher* level of generality. Instead of Turing machines tied to such peculiar human limitations as the ability to write only one symbol at a time, he considered a class of "discrete deterministic mechanical devices". However, in Gandy's analysis just as in the work of Turing that it refines, the computer remains a pure-mathematical construct, despite the physical intuitions it rests upon and despite the 'physical' language in which it may be described. For a "Gandy machine", although it is required to obey a principle of "local causality" whose justification lies in contemporary physics, is not a physical device, and there is no a priori reason to believe that its capabilities must coincide precisely with what is available in nature. Among other things, Gandy's machines are required to be "finite" (see also [15]), but his method provides no means of distinguishing physically finite processes from physically infinite ones, as we have discussed above. In order to capture the properties of real-life computing machines, the pivotal step is to impose a narrower, physical interpretation on the Church-Turing thesis: "what is effectively calculable can be computed by a *finite physical system*" — in particular by the "universal quantum computer" proposed by Deutsch [3].

In contrast with these abstract machines, the universal quantum computer, though still idealised in the sense that no prescription for building one is yet known, is defined in such a way that its constitution and time evolution conform strictly to the laws of quantum physics. Although it turns out that quantum computers do not improve upon Turing machines in regard to their class of effectively calculable functions, Gandy was absolutely right to take seriously the possibility that the computing power of a machine might be fundamentally increased if it were free of accidental human limitations. As we have seen, quantum interference allows quantum computers to be more efficient than classical ones because they can execute quantum algorithms with no classical analogues; and it also allows wholly new types of information processing such as quantum cryptography.

Among the ramifications of quantum computation for apparently distant fields of study are its implications for the notion of mathematical proof. Traditionally, a proof has been defined within a formal system as a sequence of propositions each of which is either an axiom of the system or follows from earlier propositions in the sequence by rules of inference. This notion translates straightforwardly into Turing-computational terms. Any classical computation may be regarded as providing such a sequence of propositions, corresponding to successive states of the tape, and those states can be recorded at will, and checked. All the intuition involved in constructing the proof refers to operations "so elementary that it is not easy to imagine them further divided." [16]. These finiteness conditions, by limiting the number and kind of possible steps, ensure that the Turing machine carries out its computations in a completely transparent way. Moreover the whole process is entirely reproducible.

With quantum computers the situation is fundamentally different. The "simple operations" they can perform are unitary transformations, and as in the classical case there is a finite number of them and they are intuitively 'elementary'. However, when a quantum computer halts, it is in general impossible to determine its final state: instead one must measure the value of some observable. Now, obtaining a particular value of that observable does indeed prove that that value was one of the possible outcomes of the computation with the given input and rules of transition. But one may run the program many more times and always obtain a different value: there is no guarantee that the computer will ever reproduce the original result. Furthermore, because a quantum algorithm depends on coherence, which is destroyed if a measurement is performed while it is still in progress, there is no way, even in principle, to record its progress, nor to check it step-by-step.

All currently known quantum algorithms have the property that their results, like those of factorisation algorithm, can be efficiently verified by classical means, even when they could not feasibly have been *constructed* by classical means. Thus if we regard these algorithms as theorem-proving processes, their conclusions, once obtained, are also provable classically; but what about the *correctness* of the proof performed by a quantum algorithm? How can we check that? Given a classical computer and unlimited time and tape, we could in principle calculate by classical means what all the probabilities of all the possible outcomes of the quantum computation were. However, even for relatively simple quantum computations involving a few hundred qubits, if the sequence of propositions corresponding to such a proof were printed out, the requisite tape would fill the observable universe many times over.

The fact is that quantum computers can prove theorems by methods that neither a human brain nor any other Turing-computational arbiter will ever be able to reproduce. What if a quantum algorithm delivered a theorem that it was infeasible to prove classically. No such algorithm is yet known, but nor is anything known to rule out such a possibility, and this raises a question of principle: should we still accept such a theorem as undoubtedly proved? We believe that the rational answer to this question is yes, for our confidence in quantum proofs rests upon the same foundation as our confidence in classical proofs: our acceptance of the physical laws underlying the computing operations.

These and other philosophical implications of quantum computing are discussed at length in [4].

§7. **Concluding remarks.** This brief discussion has merely scratched the surface of the rapidly developing field of quantum computation (see the Centre for Quantum Computation's web site at `http://www.qubit.org` for more detailed accounts and updates). We have concentrated on the fundamental issues and have minimised our discussion of physical details and technological practicalities. However, it should be mentioned that quantum computing is a serious possibility for future generations of computing devices. At present it is not clear when, how and even whether fully-fledged quantum computers will eventually be built; but notwithstanding this, the quantum theory of computation already plays a much more fundamental role in the scheme of things than its classical predecessor did. We believe that anyone who seeks a fundamental understanding of either physics, computation or logic must incorporate its new insights into his world view.

REFERENCES

[1] E. Bernstein and U. Vazirani, *Quantum complexity theory*, ***Proceedings of the 25th Annual Symposium on the Theory of Computing*** (New York), ACM, 1993, pp. 11–20.

[2] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, *Quantum Algorithms Revisited*, ***Proceedings of the RoyalSociety, A***, vol. 454, 1998, pp. 339–354.

[3] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, ***Proceedings of the Royal Society, A***, vol. 400, 1985, pp. 97–117.

[4] ———, ***The Fabric of Reality***, The Penguin Press, Allen Lane, 1997.

[5] A. Ekert and R. Jozsa, *Quantum computation and Shor's factoring algorithm*, ***Reviews of Modern Physics***, vol. 68 (1996), pp. 733–753.

[6] R.P. Feynman, *Simulating physics with computers*, ***International Journal of Theoretical Physics***, vol. 21 (1982), pp. 467–488.

[7] G. Galilei, *Saggiatore*, [1623], ***Opere*** (A. Favaro, editor), vol. 6, Edizione Nazionale, Firenze, 1896.

[8] R. Gandy, *Church's thesis and principles for mechanisms*, ***The Kleene Symposium*** (J. Barwise et al., editors), North-Holland, Amsterdam, 1980, pp. 123–148.

[9] H.H. Goldstine, ***The Computer from Pascal to von Neumann***, Princeton University Press, Princeton, 1972.

[10] C.H. Papadimitriou, ***Computational Complexity***, Addison-Wesley, Reading, 1994.

[11] R. Penrose, ***Shadows of the mind***, Oxford University Press, Oxford, 1994.

[12] R. Rivest, A. Shamir, and L. Adleman, *On Digital Signatures and Public-Key Cryptosystems*, ***Technical Report MIT/LCS/TR-212***, MIT Laboratory for Computer Science, January 1979.

[13] P. Shor, *Algorithms for quantum computation: discrete log and factoring*, ***Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*** (Los Alamitos) (S. Goldwasser, editor), IEEE Computer Society Press, 1994, pp. 124–134.

[14] ———, *Fault-Tolerant Quantum Computation*, ***Proceedings of the 37th Annual Symposium on the Foundations of Computer Science*** (Los Alamitos, CA), IEEE Computer Society, 1996.

[15] W. Sieg, *Mechanical procedures and mathematical experience*, ***Mathematics and Mind*** (A. George, editor), Oxford University Press, Oxford, 1994, pp. 71–117.

[16] A. Turing, *On computable numbers with an application to the Entscheidungsproblem*, ***Proceedings of the London Mathematical Society, 2***, vol. 42, 1936–37, pp. 230–265.

[17] E. P. Wigner, *The unreasonable effectiveness of mathematics in the natural sciences*, ***Communications on Pure and Applied Mathematics***, vol. 13, 1960, pp. 1–14.

CENTRE FOR QUANTUM COMPUTATION
CLARENDON LABORATORY
UNIVERSITY OF OXFORD
PARKS ROAD, OXFORD, OX1 3PU, U.K.
E-mail: david.deutch@qubit.org
E-mail: artur.ekert@qubit.org

DIPARTIMENTO DI FILOSOFIA
UNIVERSITÀ DI BOLOGNA
VIA ZAMBONI 38, 40126 BOLOGNA, ITALY.