Proc. R. Soc. Lond. A **400**, 97–117 (1985)
Printed in Great Britain

Quantum theory, the Church–Turing principle and the universal quantum computer

BY D. DEUTSCH
*Department of Astrophysics, South Parks Road, Oxford OX*1 3*RQ, U.K.*

(*Communicated by R. Penrose, F.R.S. – Received* 13 *July* 1984)

It is argued that underlying the Church–Turing hypothesis there is an implicit physical assertion. Here, this assertion is presented explicitly as a physical principle: 'every finitely realizible physical system can be perfectly simulated by a universal model computing machine operating by finite means'. Classical physics and the universal Turing machine, because the former is continuous and the latter discrete, do not obey the principle, at least in the strong form above. A class of model computing machines that is the quantum generalization of the class of Turing machines is described, and it is shown that quantum theory and the 'universal quantum computer' are compatible with the principle. Computing machines resembling the universal quantum computer could, in principle, be built and would have many remarkable properties not reproducible by any Turing machine. These do not include the computation of non-recursive functions, but they do include 'quantum parallelism', a method by which certain probabilistic tasks can be performed faster by a universal quantum computer than by any classical restriction of it. The intuitive explanation of these properties places an intolerable strain on all interpretations of quantum theory other than Everett's. Some of the numerous connections between the quantum theory of computation and the rest of physics are explored. Quantum complexity theory allows a physically more reasonable definition of the 'complexity' or 'knowledge' in a physical system than does classical complexity theory.

1. COMPUTING MACHINES AND THE CHURCH–TURING PRINCIPLE

The theory of computing machines has been extensively developed during the last few decades. Intuitively, a computing machine is any physical system whose dynamical evolution takes it from one of a set of 'input' states to one of a set of 'output' states. The states are labelled in some canonical way, the machine is prepared in a state with a given input label and then, following some motion, the output state is measured. For a classical deterministic system the measured output label is a definite function f of the prepared input label; moreover the value of that label can in principle be measured by an outside observer (the '*user*') and the machine is said to '*compute*' the function f.

Two classical deterministic computing machines are '*computationally equivalent*' under given labellings of their input and output states if they compute the same function under those labellings. But quantum computing machines, and indeed classical stochastic computing machines, do not 'compute functions' in the above sense: the output state of a stochastic machine is random with only the probability distribution function for the possible outputs depending on the input state. The output state of a quantum machine, although fully determined by the input state, is not an observable and so the user cannot in general discover its label. Nevertheless, the notion of computational equivalence can be generalized to apply to such machines also.

Again we define computational equivalence *under given labellings*, but it is now necessary to specify more precisely what is to be labelled. As far as the input is concerned, labels must be given for each of the possible ways of preparing the machine, which correspond, by definition, to all the possible input states. This is identical with the classical deterministic case. However, there is an asymmetry between input and output because there is an asymmetry between preparation and measurement: whereas a quantum system can be prepared in any desired permitted input state, measurement cannot in general determine its output state; instead one must measure the value of some observable. (Throughout this paper I shall be using the Schrödinger picture, in which the quantum state is a function of time but observables are constant operators.) Thus what must be labelled is the set of ordered pairs consisting of an output observable and a possible measured value of that observable (in quantum theory, a Hermitian operator and one of its eigenvalues). Such an ordered pair contains, in effect, the specification of a possible experiment that could be made on the output, together with a possible result of that experiment.

Two computing machines are computationally equivalent under given labellings if in any possible experiment or sequence of experiments in which their inputs were prepared equivalently under the input labellings, and observables corresponding to each other under the output labellings were measured, the measured values of these observables for the two machines would be statistically indistinguishable. That is, the probability distribution functions for the outputs of the two machines would be identical.

In the sense just described, a given computing machine $\mathscr{M}$ computes at most one function. However, there ought to be no fundamental difference between altering the input state in which $\mathscr{M}$ is prepared, and altering systematically the constitution of $\mathscr{M}$ so that it becomes a different machine $\mathscr{M}'$ computing a different function. To formalize such operations, it is often useful to consider machines with two inputs, the preparation of one constituting a 'program' determining which function of the other is to be computed. To each such machine $\mathscr{M}$ there corresponds a set $\mathrm{C}(\mathscr{M})$ of '$\mathscr{M}$-computable functions'. A function f is $\mathscr{M}$-computable if $\mathscr{M}$ can compute f when prepared with some program.

The set $\mathrm{C}(\mathscr{M})$ can be enlarged by enlarging the set of changes in the constitution of $\mathscr{M}$ that are labelled as possible $\mathscr{M}$-programs. Given two machines $\mathscr{M}$ and $\mathscr{M}'$ it is possible to construct a composite machine whose set of computable functions contains the union of $\mathrm{C}(\mathscr{M})$ and $\mathrm{C}(\mathscr{M}')$.

There is no purely logical reason why one could not go on *ad infinitum* building more powerful computing machines, nor why there should exist any function that is outside the computable set of every physically possible machine. Yet although logic does not forbid the physical computation of arbitrary functions, it seems that physics does. As is well known, when designing computing machines one rapidly reaches a point when adding additional hardware does not alter the machine's set of computable functions (under the idealization that the memory capacity is in effect unlimited); moreover, for functions from the integers $\mathbb{Z}$ to themselves the set $C(\mathscr{M})$ is always contained in $C(\mathscr{T})$, where $\mathscr{T}$ is Turing's universal computing machine (Turing 1936). $C(\mathscr{T})$ itself, also known as the set of recursive functions, is denumerable and therefore infinitely smaller than the set of all functions from $\mathbb{Z}$ to $\mathbb{Z}$.

Church (1936) and Turing (1936) conjectured that these limitations on what can be computed are not imposed by the state-of-the-art in designing computing machines, nor by our ingenuity in constructing models for computation, but are universal. This is called the 'Church–Turing hypothesis'; according to Turing,

Every 'function which would naturally be regarded as computable' can be computed by the universal Turing machine. (1.1)

The conventional, non-physical view of (1.1) interprets it as the quasi-mathematical conjecture that all possible formalizations of the intuitive mathematical notion of 'algorithm' or 'computation' are equivalent to each other. But we shall see that it can also be regarded as asserting a new physical principle, which I shall call the Church–Turing *principle* to distinguish it from other implications and connotations of the conjecture (1.1).

Hypothesis (1.1) and other formulations that exist in the literature (see Hofstadter (1979) for an interesting discussion of various versions) are very vague by comparison with physical principles such as the laws of thermodynamics or the gravitational equivalence principle. But it will be seen below that my statement of the Church–Turing principle (1.2) is manifestly physical, and unambiguous. I shall show that it has the same epistemological status as other physical principles.

I propose to reinterpret Turing's 'functions which would naturally be regarded as computable' as the functions which may in principle be computed by a real physical system. For it would surely be hard to regard a function 'naturally' as computable if it could not be computed in Nature, and conversely. To this end I shall define the notion of '*perfect simulation*'. A computing machine $\mathscr{M}$ is capable of perfectly simulating a physical system $\mathscr{S}$, under a given labelling of their inputs and outputs, if there exists a program $\pi(\mathscr{S})$ for $\mathscr{M}$ that renders $\mathscr{M}$ computationally equivalent to $\mathscr{S}$ under that labelling. In other words, $\pi(\mathscr{S})$ converts $\mathscr{M}$ into a 'black box' functionally indistinguishable from $\mathscr{S}$.

I can now state the physical version of the Church–Turing principle:

'Every finitely realizible physical system can be perfectly simulated by a universal model computing machine operating by finite means'. (1.2)

This formulation is both better defined and more physical than Turing's own way of expressing it (1.1), because it refers exclusively to objective concepts such as 'measurement', 'preparation' and 'physical system', which are already present in measurement theory. It avoids terminology like 'would naturally be regarded', which does not fit well into the existing structure of physics.

The 'finitely realizible physical systems' referred to in (1.2) must include any physical object upon which experimentation is possible. The 'universal computing machine', on the other hand, need only be an idealized (but theoretically permitted) finitely specifiable model. The labellings implicitly referred to in (1.2) must also be finitely specifiable.

The reference in (1.1) to a specific universal computing machine (Turing's) has of necessity been replaced in (1.2) by the more general requirement that this machine operate 'by finite means'. 'Finite means' can be defined axiomatically, without restrictive assumptions about the form of physical laws (cf. Gandy 1980). It we think of a computing machine as proceeding in a sequence of steps whose duration has a non-zero lower bound, then it operates 'by finite means' if (i) only a finite subsystem (though not always the same one) is in motion during any one step, and (ii) the motion depends only on the state of a finite subsystem, and (iii) the rule that specifies the motion can be given finitely in the mathematical sense (for example as an integer). Turing machines satisfy these conditions, and so does the universal quantum computer $\mathscr{Q}$ (see §II).

The statement of the Church–Turing principle (1.2) is stronger than what is strictly necessitated by (1.1). Indeed it is so strong that it is *not* satisfied by Turing's machine in classical physics. Owing to the continuity of classical dynamics, the possible states of a classical system necessarily form a continuum. Yet there are only countably many ways of preparing a finite input for $\mathscr{T}$. Consequently $\mathscr{T}$ cannot perfectly simulate any classical dynamical system. (The well studied theory of the 'simulation' of continuous systems by $\mathscr{T}$ concerns itself not with perfect simulation in my sense but with successive discrete approximation.) In §III, I shall show that it is consistent with our present knowledge of the interactions present in Nature that every real (dissipative) finite physical system can be perfectly simulated by the universal quantum computer $\mathscr{Q}$. Thus quantum theory is compatible with the strong form (1.2) of the Church–Turing principle.

I now return to my argument that (1.2) is an empirical assertion. The usual criterion for the empirical status of a theory is that it be experimentally falsifiable (Popper 1959), i.e. that there exist potential observations that would contradict it. However, since the deeper theories we call 'principles' make reference to experiment only *via* other theories, the criterion of falsifiability must be applied indirectly in their case. The principle of conservation of energy, for example, is not in itself contradicted by any conceivable observation because it contains no specification of how to measure energy. The third law of thermodynamics whose form

'*No finite process can reduce the entropy or temperature of a finitely realizible physical system to zero*' (1.3)

bears a certain resemblance to that of the Church–Turing principle, is likewise not directly refutable: no temperature measurement of finite accuracy could distinguish absolute zero from an arbitrarily small positive temperature. Similarly, since the number of possible programs for a universal computer is infinite, no experiment could in general verify that none of them can simulate a system that is thought to be a counter-example to (1.2).

But all this does not place 'principles' outside the realm of empirical science.

On the contrary, they are essential frameworks within which directly testable theories are formulated. Whether or not a given physical theory contradicts a principle is first determined by logic alone. Then, if the directly testable theory survives crucial tests but contradicts the principle, that principle is deemed to be refuted, albeit indirectly. If all known experimentally corroborated theories satisfy a restrictive principle, then that principle is corroborated and becomes, on the one hand, a guide in the construction of new theories, and on the other, a means of understanding more deeply the content of existing theories.

It is often claimed that every 'reasonable' *physical* (as opposed to mathematical) model for computation, at least for the deterministic computation of functions from $\mathbb{Z}$ to $\mathbb{Z}$, is equivalent to Turing's. But this is not so; there is no *a priori* reason why physical laws should respect the limitations of the mathematical processes we call 'algorithms' (i.e. the functions $C(\mathscr{T})$). Although I shall not in this paper find it necessary to do so, there is nothing paradoxical or inconsistent in postulating physical systems which compute functions not in $C(\mathscr{T})$. There could be experimentally testable theories to that effect: e.g. consider any recursively enumerable non-recursive set (such as the set of integers representing programs for terminating algorithms on a given Turing machine). In principle, a physical theory might have among its implications that a certain physical device $\mathscr{F}$ could compute in a specified time whether or not an arbitrary integer in its input belonged to that set. This theory would be experimentally refuted if a more pedestrian Turing-type computer, programmed to enumerate the set, ever disagreed with $\mathscr{F}$. (Of course the theory would have to make other predictions as well, otherwise it could never be non-trivially *corroborated*, and its structure would have to be such that its exotic predictions about $\mathscr{F}$ could not naturally be severed from its other physical content. All this is logically possible.)

Nor, conversely, is it obvious *a priori* that any of the familiar recursive functions is in physical reality computable. The reason why we find it possible to construct, say, electronic calculators, and indeed why we can perform mental arithmetic, cannot be found in mathematics or logic. *The reason is that the laws of physics 'happen to' permit the existence of physical models for the operations of arithmetic* such as addition, subtraction and multiplication. If they did not, these familiar operations would be non-computable functions. We might still know *of* them and invoke them in mathematical proofs (which would presumably be called 'non-constructive') but we could not perform them.

If the dynamics of some physical system did depend on a function not in $C(\mathscr{T})$, then that system could in principle be used to compute the function. Chaitin (1977) has shown how the truth values of all 'interesting' non-Turing decidable propositions of a given formal system might be tabulated very efficiently in the first few significant digits of a single physical constant.

But if they were, it might be argued, we could never know because we could not check the accuracy of the 'table' provided by Nature. This is a fallacy. The reason why we are confident that the machines we call calculators do indeed compute the arithmetic functions they claim to compute is not that we can 'check' their answers, for this is ultimately a futile process of comparing one machine with another: *Quis custodiet custodios ipsos?* The real reason is that we believe the detailed physical theory that was used in their design. That theory, including its assertion that the abstract functions of arithmetic are realized in Nature, is empirical.

II. Quantum computers

Every existing general model of computation is effectively classical. That is, a full specification of its state at any instant is equivalent to the specification of a set of numbers, all of which are in principle measurable. Yet according to quantum theory there exist no physical systems with this property. The fact that classical physics and the classical universal Turing machine do not obey the Church–Turing principle in the strong physical form (1.2) is one motivation for seeking a truly quantum model. The more urgent motivation is, of course, that classical physics is false.

Benioff (1982) has constructed a model for computation within quantum kinematics and dynamics, but it is still effectively classical in the above sense. It is constructed so that at the end of each elementary computational step, no characteristically quantum property of the model – interference, non-separability, or indeterminism – can be detected. Its computations can be perfectly simulated by a Turing machine.

Feynman (1982) went one step closer to a true quantum computer with his 'universal quantum simulator'. This consists of a lattice of spin systems with nearest-neighbour interactions that are freely specifiable. Although it can surely simulate any system with a finite-dimensional state space (I do not understand why Feynman doubts that it can simulate fermion systems), it is not a computing machine in the sense of this article. 'Programming' the simulator consists of endowing it by *fiat* with the desired dynamical laws, and then placing it in a desired initial state. But the mechanism that allows one to select arbitrary dynamical laws is not modelled. The dynamics of a true 'computer' in my sense must be given once and for all, and programming it must consist entirely of preparing it in a suitable *state* (or mixed case).

Albert (1983) has described a quantum mechanical measurement 'automaton' and has remarked that its properties on being set to measure itself have no analogue among classical automata. Albert's automata, though they are not general purpose computing machines, are true quantum computers, members of the general class that I shall study in this section.

In this section I present a general, fully quantum model for computation. I then describe the universal quantum computer $\mathscr{Q}$, which is capable of perfectly simulating every finite, realizible physical system. It can simulate ideal closed (zero temperature) systems, including all other instances of quantum computers and quantum simulators, with arbitrarily high but not perfect accuracy. In computing strict functions from $\mathbb{Z}$ to $\mathbb{Z}$ it generates precisely the classical recursive functions $C(\mathscr{T})$ (a manifestation of the correspondence principle). Unlike $\mathscr{T}$, it can simulate any finite classical discrete stochastic process perfectly. Furthermore, as we shall see in §III, it as many remarkable and potentially useful capabilities that have no classical analogues.

Like a Turing machine, a model quantum computer $\mathscr{Q}$ consists of two components, a finite *processor* and an infinite *memory*, of which only a finite portion is ever used. The computation proceeds in steps of fixed duration T, and during each step only the processor and a finite part of the memory interact, the rest of the memory remaining static.

The processor consists of M 2-state observables

$$\{\hat{n}_i\} \quad (i \in \mathbb{Z}_M), \tag{2.1}$$

where $\mathbb{Z}_M$ is the set of integers from 0 to $M-1$. The memory consists of an infinite sequence

$$\{\hat{m}_i\} \quad (i \in \mathbb{Z}) \tag{2.2}$$

of 2-state observables. This corresponds to the infinitely long memory 'tape' in a Turing machine. I shall refer to the $\{\hat{n}_i\}$ collectively as $\hat{\boldsymbol{n}}$, and to the $\{\hat{m}_i\}$ as $\hat{\boldsymbol{m}}$. Corresponding to Turing's 'tape position' is another observable $\hat{x}$, which has the whole of $\mathbb{Z}$ as its spectrum. The observable $\hat{x}$ is the 'address' number of the currently scanned tape location. Since the 'tape' is infinitely long, but will be in motion during computations, it must not be rigid or it could not be made to move 'by finite means'. A mechanism that moved the tape according to signals transmitted at finite speed between adjacent segments only would satisfy the 'finite means' requirement and would be sufficient to implement what follows. Having satisfied ourselves that such a mechanism is possible, we shall not need to model it explicitly. Thus the state of $\mathcal{Q}$ is a unit vector in the space $\mathcal{H}$ spanned by the simultaneous eigenvectors

$$|x; \boldsymbol{n}; \boldsymbol{m}\rangle \equiv |x; n_0, n_1 \ldots n_{M-1}; \ldots m_{-1}, m_0, m_1 \ldots\rangle \tag{2.3}$$

of $\hat{x}$, $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{m}}$, labelled by the corresponding eigenvalues x, $\boldsymbol{n}$ and $\boldsymbol{m}$. I call (2.3) the '*computational basis states*'. It is convenient to take the spectrum of our 2-state observables to be $\mathbb{Z}_2$, i.e. the set $\{0, 1\}$, rather than $\{-\frac{1}{2}, +\frac{1}{2}\}$ as is customary in physics. An observable with spectrum $\{0, 1\}$ has a natural interpretation as a 'one-bit' memory element.

The dynamics of $\mathcal{Q}$ are summarized by a constant unitary operator U on $\mathcal{H}$. U specifies the evolution of any state $|\psi(t)\rangle \in \mathcal{H}$ (in the Schrödinger picture at time t) during a single computation step

$$|\psi(nT)\rangle = \mathsf{U}^n |\psi(0)\rangle \quad (n \in \mathbb{Z}^+), \tag{2.4}$$

$$\mathsf{U}^\dagger \mathsf{U} = \mathsf{U}\mathsf{U}^\dagger = \hat{1}. \tag{2.5}$$

We shall not need to specify the state at times other than non-negative integer multiples of T. The computation begins at $t = 0$. At this time $\hat{x}$ and $\hat{n}$ are prepared with the value zero, the state of a finite number of the $\hat{\boldsymbol{m}}$ is prepared as the 'program' and 'input' in the sense of §I and the rest are set to zero. Thus

$$\left.\begin{aligned} |\psi(0)\rangle &= \sum_{\boldsymbol{m}} \lambda_{\boldsymbol{m}} |0; \boldsymbol{0}; \boldsymbol{m}\rangle, \\ \sum_{\boldsymbol{m}} |\lambda_{\boldsymbol{m}}|^2 &= 1, \end{aligned}\right\} \tag{2.6}$$

where only a finite number of the $\lambda_{\boldsymbol{m}}$ are non-zero and $\lambda_{\boldsymbol{m}}$ vanishes whenever an infinite number of the $\boldsymbol{m}$ are non-zero.

To satisfy the requirement that $\mathscr{Q}$ operate 'by finite means', the matrix elements of U take the following form:

$$\begin{aligned}&\langle x'; \boldsymbol{n}'; \boldsymbol{m}' \,|\, \mathsf{U} \,|\, x; \boldsymbol{n}; \boldsymbol{m}\rangle \\ &\quad = [\delta_{x'}^{x+1}\, \mathsf{U}^{+}(\boldsymbol{n}', m_x' \,|\, \boldsymbol{n}, m_x) + \delta_{x'}^{x-1}\, \mathsf{U}^{-}(\boldsymbol{n}', m_x' \,|\, \boldsymbol{n}, m_x)] \prod_{y \neq x} \delta_{m_y'}^{m_y}. \qquad (2.7)\end{aligned}$$

The continued product on the right ensures that only one memory bit, the xth, participates in a single computational step. The terms $\delta_{x'}^{x\pm 1}$ ensure that during each step the tape position x cannot change by more than one unit, forwards or backwards, or both. The functions $\mathsf{U}^{\pm}(\boldsymbol{n}', m' \,|\, \boldsymbol{n}, m)$, which represent a dynamical motion depending only on the 'local' observables $\hat{\boldsymbol{n}}$ and $\hat{m}_x$, are arbitrary except for the requirement (2.5) that U be unitary. Each choice defines a different quantum computer, $\mathscr{Q}[\mathsf{U}^{+}, \mathsf{U}^{-}]$.

Turing machines are said to '*halt*', signalling the end of the computation, when two consecutive states are identical. A 'valid' program is one that causes the machine to halt after a finite number of steps. However, (2.4) shows that two consecutive states of a quantum computer $\mathscr{Q}$ can never be identical after a non-trivial computation. (This is true of any reversible computer.)

Moreover, $\mathscr{Q}$ must not be observed before the computation has ended since this would, in general, alter its relative state. Therefore, quantum computers need to signal actively that they have halted. One of the processor's internal bits, say $\hat{n}_0$, must be set aside for this purpose. Every valid $\mathscr{Q}$-program sets n_0 to 1 when it terminates but does not interact with $\hat{n}_0$ otherwise. The observable $\hat{n}_0$ can then be periodically observed from the outside without affecting the operation of $\mathscr{Q}$. The analogue of the classical condition for a program to be valid would be that the expectation value of $\hat{n}_0$ must go to one in a finite time. However, it is physically reasonable to allow a wider class of $\mathscr{Q}$-programs. A $\mathscr{Q}$-program is valid if the expectation value of its *running time* is finite.

Because of unitarity, the dynamics of $\mathscr{Q}$, as of any closed quantum system, are necessarily reversible. Turing machines, on the other hand, undergo irreversible changes during computations, and indeed it was, until recently, widely held that irreversibility is an essential feature of computation. However, Bennett (1973) proved that this is not the case by constructing explicitly a reversible classical model computing machine equivalent to (i.e. generating the same computable function as) $\mathscr{T}$ (see also Toffoli 1979). (Benioff's machines are equivalent to Bennett's but use quantum dynamics.)

Quantum computers $\mathscr{Q}[\mathsf{U}^{+}, \mathsf{U}^{-}]$ equivalent to any reversible Turing machine may be obtained by taking

$$\mathsf{U}^{\pm}(\boldsymbol{n}', m' \,|\, \boldsymbol{n}, m) = \tfrac{1}{2}\delta_{\boldsymbol{n}'}^{\boldsymbol{A}(\boldsymbol{n},\, m)}\, \delta_{m'}^{B(\boldsymbol{n},\, m)}[1 \pm \mathrm{C}(\boldsymbol{n}, m)], \qquad (2.8)$$

where $\boldsymbol{A}$, B and C are functions with ranges $(\mathbb{Z}_2)^M$, $\mathbb{Z}_2$ and $\{-1, 1\}$ respectively. Turing machines, in other words, are those quantum computers whose dynamics ensure that they remain in a computational basis state at the end of each step, given that they start in one. To ensure unitarity it is necessary and sufficient that the mapping

$$\{(\boldsymbol{n}, m)\} \leftrightarrow \{(\boldsymbol{A}(\boldsymbol{n}, m), B(\boldsymbol{n}, m), C(\boldsymbol{n}, m))\} \tag{2.9}$$

be bijective. Since the constitutive functions $\boldsymbol{A}$, B and C are otherwise arbitrary there must, in particular, exist choices that make $\mathcal{Q}$ equivalent to a universal Turing machine $\mathcal{T}$.

To describe the universal quantum computer $\mathcal{Q}$ directly in terms of its constitutive transformations $\mathsf{U}^{\pm}$ would be possible, but unnecessarily tedious. The properties of $\mathcal{Q}$ are better defined by resorting to a higher level description, leaving the explicit construction of $\mathsf{U}^{\pm}$ as an exercise for the reader. In the following I repeatedly invoke the 'universal' property of $\mathcal{T}$.

For every recursive function f there exists a program $\pi(f)$ for $\mathcal{T}$ such that when the image of $\pi(f)$ is followed by the image of any integer i in the input of $\mathcal{T}$, $\mathcal{T}$ eventually halts with $\pi(f)$ and i themselves followed by the image of $f(i)$, with all other bits still (or again) set to zero. That is, for some positive integer n

$$\mathsf{U}^{n}|0; \mathbf{0}; \pi(f), i, \mathbf{0}\rangle = |0; 1, \mathbf{0}; \pi(f), i, f(i), \mathbf{0}\rangle. \tag{2.10}$$

Here $\mathbf{0}$ denotes a sequence of zeros, and the zero eigenvalues of $\hat{m}_i$ $(i < 0)$ are not shown explicitly. $\mathcal{T}$ loses no generality if it is required that every program allocate the memory as an infinite sequence of 'slots', each capable of holding an arbitrary integer. (For example, the ath slot might consist of the bits labelled by successive powers of the ath prime.) For each recursive function f and integers a, b there exists a program $\pi(f, a, b)$, which computes the function f on the contents of slot a and places the result in slot b, leaving slot a unchanged. If slot b does not initially contain zero, reversibility requires that its old value be not overwritten but combined in some reversible way with the value of the function. Thus, omitting explicit mention of everything unnecessary, we may represent the effect of the program π by

$$|\overset{\substack{\text{slot 1}\\ \downarrow}}{\pi(f, 2, 3)}, \overset{\substack{\text{slot 2}\\ \downarrow}}{i}, \overset{\substack{\text{slot 3}\\ \downarrow}}{j}\rangle \rightarrow |\pi(f, 2, 3), i, j \oplus f(i)\rangle, \tag{2.11}$$

where $\oplus$ is any associative, commutative operator with the properties

$$\left.\begin{aligned} i \oplus i &= 0, \\ i \oplus 0 &= i, \end{aligned}\right\} \tag{2.12}$$

(the exclusive-or function, for example, would be satisfactory). I denote by $\pi_1 \cdot \pi_2$ the *concatenation* of two programs π_1 and π_2, which always exists when π_1 and π_2 are valid programs; $\pi_1 \cdot \pi_2$ is a program whose effect is that of π_1 followed by π_2.

For any bijective recursive function g there exists a program $\phi(g, a)$ whose sole effect is to replace any integer i in slot a by $g(i)$. The proof is immediate, for if some slot b initially contains zero,

$$\phi(g, a) = \pi(g, a, b) \cdot \pi(g^{-1}, b, a) \cdot \pi(I, b, a) \cdot \pi(I, a, b). \tag{2.13}$$

Here I is the 'perfect measurement' function (Deutsch 1985)

$$|\pi(I,2;3), i, j\rangle \to |\pi(I,2,3), i, j \oplus i\rangle. \tag{2.14}$$

The universal quantum computer $\mathcal{Q}$ has all the properties of $\mathcal{T}$ just described, as summarized in (2.10) to (2.14). But $\mathcal{Q}$ admits a further class of programs which evolve computational basis states into linear superpositions of each other. All programs for $\mathcal{Q}$ can be expressed in terms of the ordinary Turing operations and just eight further operations. These are unitary transformations confined to a single two-dimensional Hilbert space $\mathcal{H}$, the state space of a single bit. Such transformations form a four (real) parameter family. Let α be any irrational multiple of π. Then the four transformations

$$\left.\begin{aligned} \mathsf{V}_0 &= \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}, & \mathsf{V}_1 &= \begin{bmatrix} \cos\alpha & \mathrm{i}\sin\alpha \\ \mathrm{i}\sin\alpha & \cos\alpha \end{bmatrix}, \\ \mathsf{V}_2 &= \begin{bmatrix} \mathrm{e}^{\mathrm{i}\alpha} & 0 \\ 0 & 1 \end{bmatrix}, & \mathsf{V}_3 &= \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\alpha} \end{bmatrix}, \end{aligned}\right\} \tag{2.15}$$

and their inverses V_4, V_5, V_6, V_7, generate, under composition, a group dense in the group of all unitary transformations on $\mathcal{H}$. It is convenient, though not essential, to add two more generators

$$\mathsf{V}_8 = 2^{-\frac{1}{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad \mathsf{V}_9 = 2^{-\frac{1}{2}}\begin{bmatrix} 1 & \mathrm{i} \\ \mathrm{i} & 1 \end{bmatrix}, \tag{2.16}$$

which corresponds to 90° 'spin rotations'. To each generator V_i there correspond computational basis elements representing programs $\phi(\mathsf{V}_i, a)$, which perform V_i upon the least significant bit of the ath slot. Thus if j is zero or one, these basis elements evolve according to

$$|\phi(\mathsf{V}_i,2), j\rangle \to \sum_{k=0}^{1} \langle k|\mathsf{V}_i|j\rangle\, |\phi(\mathsf{V}_i,2), k\rangle. \tag{2.17}$$

Composition of the V_i may be effected by concatenation of the $\phi(\mathsf{V}_i, a)$. Thus there exist programs that effect upon the state of any one bit a unitary transformation arbitrarily close to any desired one.

Analogous conclusions hold for the joint state of any finite number L of specified bits. This is not a trivial observation since such a state is not necessarily a direct product of states confined to the Hilbert spaces of the individual bits, but is in general a linear superposition of such products. However, I shall now sketch a proof of the existence of a program that effects a unitary transformation on L bits, arbitrarily close to any desired unitary transformation. In what follows, 'accurate' means 'arbitrarily accurate with respect to the inner product norm'. The case $L = 1$ is trivial. The proof for L bits is by induction.

First note that the $(2^L)!$ possible permutations of the 2^L computational basis states of L bits are all invertible recursive functions, and so can be effected by programs for $\mathcal{T}$, and hence for $\mathcal{Q}$.

Next we show that it is possible for $\mathcal{Q}$ to generate 2^L-dimensional unitary transformations diagonal in the computation basis, arbitrarily close to any transformation diagonal in that basis. The $(L-1)$-bit diagonal transformations, which are accurately $\mathcal{Q}$-computable by the inductive hypothesis, are generated by certain 2^L-dimensional diagonal unitary matrices whose eigenvalues all have even degeneracy. The permutations of basis states allow $\mathcal{Q}$ accurately to effect every diagonal unitary transformation with this degeneracy. The closure of this set of degenerate transformations under multiplications is a group of diagonal transformations dense in the group of all 2^L-dimensional diagonal unitary transformations.

Next we show that for each L-bit state $|\psi\rangle$ there exists a $\mathcal{Q}$-program $\rho(|\psi\rangle)$ which accurately evolves $|\psi\rangle$ to the basis state $|0_L\rangle$ in which all L bits are zero. Write

$$|\psi\rangle = c_0|0\rangle|\psi_0\rangle + c_1|1\rangle|\psi_1\rangle, \tag{2.18}$$

where $|\psi_0\rangle$ and $|\psi_1\rangle$ are states of the $L-1$ bits numbered 2 to L. By the inductive hypothesis there exist $\mathcal{Q}$-programs ρ_0 and ρ_i which accurately evolve $|\psi_0\rangle$ and $|\psi_i\rangle$, respectively, to the $(L-1)$-fold product $|0_{L-1}\rangle$. Therefore there exists a $\mathcal{Q}$-program with the following effect. If bit no. 1 is a zero, execute ρ_0, otherwise execute ρ_1. This converts (2.18) accurately to

$$(c_0|0\rangle + c_1|1\rangle)|0_{L-1}\rangle. \tag{2.19}$$

Then (2.19) can be evolved accurately to $|0_L\rangle$ by a transformation of bit no. 1.

Finally, an arbitrary 2^L-dimensional transformation U is accurately effected by successively transforming each eigenvector $|\psi\rangle$ of U accurately into $|0_L\rangle$ (by executing the program $\rho^{-1}(|\psi\rangle)$), then performing a diagonal unitary transformation which accurately multiplies $|0_L\rangle$ by the eigenvalue (a phase factor) corresponding to $|\psi\rangle$, but has arbitrarily little effect on any other computational basis state, and then executing $\rho(|\psi\rangle)$.

This establishes the sense in which $\mathcal{Q}$ is a *universal* quantum computer. It can simulate with arbitrary precision any other quantum computer $\mathcal{Q}[\mathsf{U}^+, \mathsf{U}^-]$. For although a quantum computer has an infinite-dimensional state space, only a finite-dimensional unitary transformation need be effected at every step to simulate its evolution.

III. Properties of the universal quantum computer

We have already seen that the universal quantum computer $\mathcal{Q}$ can perfectly simulate any Turing machine and can simulate with arbitrary precision any quantum computer or simulator. I shall now show how $\mathcal{Q}$ can simulate various physical systems, real and theoretical, which are beyond the scope of the universal Turing machine $\mathcal{T}$.

Random numbers and discrete stochastic systems

As is to be expected, there exist programs for $\mathcal{Q}$ which generate true random numbers. For example, when the program

$$\phi(\mathsf{V}_8, 2)\cdot\pi(I, 2, a) \tag{3.1}$$

halts, slot a contains with probability $\frac{1}{2}$ either a zero or a one. Iterative programs incorporating (3.1) can generate other probabilities, including any probability that is a recursive real. However, this does not exhaust the abilities of $\mathcal{Q}$. So far, all our programs have been, *per se*, classical, though they may cause the 'output' part of the memory to enter non-computational basis states. We now encounter our first quantum program. The execution of

$$2^{-\frac{1}{2}}\underset{\text{slot 1}}{|\pi(I,2,a)\rangle}\,\underset{\text{slot 2}}{(\cos\theta|0\rangle+\sin\theta|1\rangle)} \tag{3.2}$$

yields in slot a, a bit that is zero with probability $\cos^2\theta$. All $\aleph_1$ states of the form (3.2) are valid programs for $\mathcal{Q}$. In particular, valid programs exist with arbitrary irrational probabilities $\cos^2\theta$ and $\sin^2\theta$. It follows that every discrete finite stochastic system, whether or not its probability distribution function is $\mathcal{T}$-computable, can be perfectly simulated by $\mathcal{Q}$. Even if $\mathcal{T}$ were given access to a 'hardware random number generator' (which cannot really exist classically) or a 'random oracle' (Bennett 1981) it could not match this. However, it could get arbitrarily close to doing so. But neither $\mathcal{T}$ nor any classical system whatever, including stochastic ones, can even approximately simulate the next property of $\mathcal{Q}$.

Quantum correlations

The random number generators (3.1) and (3.2) differ slightly from the other programs I have so far considered in that they necessarily produce 'waste' output. The bit in slot a is, strictly speaking, perfectly random only if the contents of slot 2 are hidden from the user and never again participate in computations. The quantum program (3.2) can be used only once to generate a single random bit. If it were re-used the output would contain non-random correlations.

However, in some applications, such correlations are precisely what is required. The state of slots 2 and a after the execution of (3.1) is the 'non-separable' (d'Espagnat 1976) state

$$2^{-\frac{1}{2}}(|0\rangle|0)+|1\rangle|1\rangle). \tag{3.3}$$

Consider a pair of programs that swap these slots into an output region of the tape *one at a time*. That is, if the output is at first blank,

$$2^{-\frac{1}{2}}(|0\rangle|0)+|1\rangle|1\rangle)\overset{\text{output}}{|0\rangle|0\rangle}, \tag{3.4}$$

execution of the first program halts with

$$2^{-\frac{1}{2}}|0\rangle(|0\rangle|0\rangle+|1\rangle|1\rangle)|0\rangle, \tag{3.5}$$

and, execution of the second program halts with

$$2^{-\frac{1}{2}}|0\rangle|0\rangle(|0\rangle|0\rangle+|1\rangle|1\rangle). \tag{3.6}$$

An equivalent program is shown explicitly at the end of §4. Bell's (1964) theorem tells us that no classical system can reproduce the statistical results of consecutive measurements made on the output slots at times (3.5) and (3.6). (Causing the output to appear in two steps with an opportunity for the user to perform an experiment after each step is sufficient to satisfy the locality requirement in Bell's theorem.)

The two bits in (3.3) can also be used as 'keys' for performing 'quantum cryptography' (Bennett *et al.* 1983).

Perfect simulation of arbitrary finite physical systems

The dynamics of quantum computers, though by construction 'finite', are still unphysical in one important respect: the evolution is strictly unitary. However, the third law of thermodynamics (1.3) implies that no realizible physical system can be prepared in a state uncorrelated with systems outside itself, because its entropy would then be zero. Therefore, every realizible physical system interacts with other systems, in certain states. But the effect of its dynamical coupling to systems outside itself cannot be reduced to zero by a finite process because the temperature of the correlation degrees of freedom would then have been reduced to zero. Therefore there can be no realizible way of placing the system in states on which the components of the time evolution operator which mix internal and external degrees of freedom have no effect.

A faithful description of a finitely realizible physical system with an L-dimensional state space $\mathscr{H}$ cannot therefore be made *via* state vectors in $\mathscr{H}$ but must use density matrices $\rho_a{}^b$. Indeed, all density matrices are in principle allowed except (thanks to the 'entropy' half of the third law (1.3)) pure cases. The dynamics of such a system are generated not by a unitary operator but by a superscattering matrix \$:

$$\rho_a{}^b(T) = \sum_{c,\,d} \$_a{}^{bc}{}_d\, \rho_c{}^d(0). \tag{3.7}$$

It is worth stressing that I am not advocating non-unitary dynamics for the universe as a whole, which would be a heresy contrary to quantum theory. Equation (3.7) is, of course, merely the projection into $\mathscr{H}$ of unitary evolution in a higher state space $\mathscr{H} \times \mathscr{H}'$, where $\mathscr{H}'$ represents as much of the rest of the universe as necessary. Roughly speaking (the systems are far from equilibrium) $\mathscr{H}'$ plays the role of a 'heat bath'.

Thus the general superscattering operator has the form

$$\$_a{}^{bc}{}_d = \sum_{e',\,f',\,g'} \mathsf{U}_{ae'}{}^{cf'}\, \mathsf{U}^{bc'}{}_{dg'}\, \bar{\rho}_{f'}{}^{g'}, \tag{3.8}$$

where $\mathsf{U}_{ab'}{}^{cd'}$ is a unitary operator on $\mathscr{H} \times \mathscr{H}'$, that is

$$\sum_{c,\,d'} \mathsf{U}_{ab'}{}^{cd'}\, \mathsf{U}^{ef'}{}_{cd'} = \delta_a{}^e\, \delta_{b'}{}^{f'}, \tag{3.9}$$

which does not decompose into a product of operators on $\mathscr{H}$ and $\mathscr{H}'$. (Raising and lowering of indices denotes complex conjugation.) The term $\bar{\rho}_{a'}{}^{b'}$ has an approximate interpretation as the initial density matrix of the 'heat bath', which would be strictly true if the system, the heat bath, and the entity preparing the system in its initial state were all uncorrelated initially. Let us rewrite (3.8) in the $\mathscr{H}'$-basis in which $\bar{\rho}$ is diagonal:

$$\$_a{}^{bc}{}_d = \sum_{e', f'} P_{f'} \mathsf{U}_{ae'}{}^{cf'} \mathsf{U}^{be'}{}_{df'},$$

$$\sum_{a'} P_{a'} = 1, \tag{3.10}$$

where the probabilities $P_{a'}$ are the eigenvalues of $\bar{\rho}$. The set $\mathfrak{S}$ of all superscattering matrices (3.8) or (3.10) lies in a subspace $\mathscr{J}$ of $\mathscr{H} \times \mathscr{H}^* \times \mathscr{H}^* \times \mathscr{H}$, namely the subspace whose elements satisfy

$$\sum_a \$_a{}^{ab}{}_c = \delta^b{}_c. \tag{3.11}$$

Every element of $\mathfrak{S}$ satisfies the constraints

$$0 \leqslant \sum_{a, b, c, d} \rho^{(1)}{}_b \$_a{}^{bc}{}_d \rho^{(2)}{}_c{}^d \leqslant 1 \tag{3.12}$$

for arbitrary density matrices $\rho^{(1)}$ and $\rho^{(2)}$.

The inequality on the left in (3.12) can be an equality only if the states of $\mathscr{H}$ form disjoint subsets with strictly zero probability so that thermal noise can effect a transition between them. This is impossible unless there are superselection rules forbidding such transitions, a possibility that we lose no generality by excluding because only one superselected sector at a time can be realized as a physical system. The inequality on the right becomes an equality precisely in the unitary case

$$\$_a{}^{bc}{}_d = \mathsf{U}_a{}^c \mathsf{U}^b{}_d, \tag{3.13}$$

which is unphysical because it represents perfectly non-dissipative evolution. Thus the set of physically realizible elements of $\mathfrak{S}$ is an open set in $\mathscr{J}$. Moreover, for any $\$^{(1)}$ and $\$^{(2)}$ that are $\mathscr{Q}$-computable the convex linear combination

$$p_1 \$^{(1)} + p_2 \$^{(2)}, \tag{3.14}$$

where p_1 and p_2 are arbitrary probabilities, is also computable, thanks to the random number generator (3.2). By computing unitary transformations as in (3.10), every element of a certain countable dense subset of $\mathfrak{S}$ can be computed. But every point in any open region of a finite-dimensional vector space can be represented as a finite convex linear combination of elements of any dense subset of that space. It follows that $\mathscr{Q}$ can perfectly simulate any physical system with a finite-dimensional state space. Therefore quantum theory is compatible with the Church–Turing principle (1.2).

The question whether all finite systems in the physical universe can likewise be simulated by $\mathscr{Q}$ – i.e. whether (1.2) is satisfied in Nature – must remain open until the state space and dynamics of the universe are understood better. What little is known seems to bear out the principle. If the theory of the thermodynamics of black holes is trustworthy, no system enclosed by a surface with an appropriately defined area A can have more than a finite number (Bekenstein 1981)

$$N(A) = \exp(Ac^3/4\hbar G) \tag{3.15}$$

of distinguishable accessible states ($\hbar$ is the Planck reduced constant, G is the gravitational constant and c is the speed of light). That is, in a suitable basis the system can be perfectly described by using an $N(A)$-dimensional state space, and hence perfectly simulated by $\mathcal{Q}$.

Parallel processing on a serial computer

Quantum theory is a theory of parallel interfering universes. There are circumstances under which different computations performed in different universes can be combined by $\mathcal{Q}$, giving it a limited capacity for parallel processing. Consider the quantum program

$$N^{-\frac{1}{2}} \sum_{i=1}^{N} |\pi(f,2,3),\, i,\, 0\rangle, \tag{3.16}$$

which instructs $\mathcal{Q}$ in each of N universes to compute $f(i)$, for i from 1 to N. Linearity and (2.11) imply that after executing (3.16) $\mathcal{Q}$ halts in the state

$$N^{-\frac{1}{2}} \sum_{i=1}^{N} |\pi(f,2,3),\, i,\, f(i)\rangle. \tag{3.17}$$

Although this computation requires exactly the same time, memory space and hardware as (2.11), the state (3.17) contains the results of an arbitrarily large number N of separate computations. Unfortunately, at most one of these results is accessible in each universe. If (3.16) is executed many times, the mean time required to compute all N values $f(i)$, which I shall refer to collectively as $\boldsymbol{f}$, is at least that required for (2.11) to compute all of them serially. I shall now show that the expectation value of the time to compute any non-trivial N-fold parallelizable function $G(\boldsymbol{f})$ of all N values $\boldsymbol{f}$ via quantum parallelism such as (3.16) cannot be less than the time required to compute it serially via (2.11).

For simplicity assume that τ, the running time of (2.11), is independent of i and that the time taken to combine all the $\boldsymbol{f}$ to form $G(\boldsymbol{f})$ is negligible compared with τ. Now suppose that there exists a program ζ, which for any function f extracts the value of $G(\boldsymbol{f})$ from (3.17) in a negligible time and with probability $|\beta|^2$. That is, ζ has the effect

$$N^{-\frac{1}{2}} \sum_{i=1}^{N} |i, f(i)\rangle \rightarrow \beta\,|0, G(\boldsymbol{f})\rangle + (1-|\beta|^2)^{\frac{1}{2}}\,|1\rangle\,|\lambda(\boldsymbol{f})\rangle, \tag{3.18}$$

where the states $|\lambda(\boldsymbol{f})\rangle$ contain no information about $G(\boldsymbol{f})$. Then the first slot could be measured. If it contained zero, the second slot would contain $G(\boldsymbol{f})$. Otherwise the information in (3.17) would have been lost and it would have to be recomputed. Unitarity implies

$$N^{-1} \sum_{i=1}^{N} \delta(f(i), g(i)) = |\beta|^2\,\delta(G(\boldsymbol{f}), G(\boldsymbol{g})) + (1-|\beta|^2)\,\langle\lambda(\boldsymbol{f})\,|\,\lambda(\boldsymbol{g})\rangle \tag{3.19}$$

for any functions $g(i)$ and $f(i)$.

If $G(\boldsymbol{f})$ is not a constant function then for each function $f(i)$ there exists another function $g(i)$ such that $G(\boldsymbol{g}) \neq G(\boldsymbol{f})$, but $g(i) = f(i)$ for all but one value of i between 1 and N. For this choice

$$1 - N^{-1} = (1-|\beta|^2)\,\langle\lambda(\boldsymbol{f})\,|\,\lambda(\boldsymbol{g})\rangle, \tag{3.20}$$

whence it follows that $|\beta|^2 < N^{-1}$. Thus the mean time to compute $G(\boldsymbol{f})$ must be at least $\tau/|\beta|^2 = N\tau$. This establishes that quantum parallelism cannot be used to improve the mean running time of parallelizable algorithms.

As an example of quantum parallelism for $N = 2$, let

$$G(\boldsymbol{f}) \equiv f(0) \oplus f(1), \tag{3.21}$$

(see equations (2.12)). Then the state (3.17) following the quantum parallel computation has

$$2^{-\frac{1}{2}}(|0, f(0)\rangle + |1, f(1)\rangle) \tag{3.22}$$

as a factor. A suitable program ζ to 'decode' this is one that effects a measurement of any non-degenerate observable with eigenstates

$$\left.\begin{aligned} |\text{zero}\rangle &\equiv \tfrac{1}{2}(|0,0\rangle - |0,1\rangle + |1,0\rangle - |1,1\rangle),\\ |\text{one}\rangle &\equiv \tfrac{1}{2}(|0,0\rangle - |0,1\rangle - |1,0\rangle + |1,1\rangle),\\ |\text{fail}\rangle &\equiv \tfrac{1}{2}(|0,0\rangle + |0,1\rangle + |1,0\rangle + |1,1\rangle),\\ |\text{error}\rangle &\equiv \tfrac{1}{2}(|0,0\rangle + |0,1\rangle - |1,0\rangle - |1,1\rangle).\end{aligned}\right\} \tag{3.23}$$

Such an observable exists, since the states (3.23) form an orthonormal set. Furthermore, the measurement can be made in a fixed time independent of the execution time of the algorithm computing f. If the outcome of the measurement is 'zero' (i.e. the eigenvalue corresponding to the state $|\text{zero}\rangle$) or 'one' then it can be inferred that $f(0) \oplus f(1)$ is zero or one respectively. Whatever the form of the function f, there will be a probability $\frac{1}{2}$ that the outcome will be 'fail', in which case nothing can be inferred about the value of $f(0) \oplus f(1)$. The probability of the outcome 'error' can be made arbitrarily small with a computational effort independent of the nature of f.

In this example the bound $N\tau$ for the running time has been attained. However, for $N > 2$ I have been unable to construct examples where the mean running time is less than $(N^2 - 2N + 2)\tau$, and I conjecture that this is the optimal lower bound. Also, although there exist non-trivial examples of quantum parallelizable algorithms for all N, when $N > 2$ there are none for which the function $G(\boldsymbol{f})$ has the set of all 2^N possible graphs of f as its domain.

In practical computing problems, especially in real time applications, one may not be concerned with minimizing specifically the *mean* running time of a program: often it is required that the minimum or maximum time or some more complicated measure be minimized. In such cases quantum parallelism may come into its own. I shall give two examples.

(1) Suppose that (3.17) is a program to estimate tomorrow's Stock Exchange movements given today's, and $G(\boldsymbol{f})$ specifies the best investment strategy. If τ were one day and $N = 2$, the classical version of this program would take two days to run and would therefore be useless. If the quantum version was executed every day, then on one day in two on average slot 1 would contain the measured value '1', indicating a failure. On such days one would make no investment. But with equal average frequency a zero would appear, indicating that slot 2 contained the correct value of the investment strategy $G(\boldsymbol{f})$. $G(\boldsymbol{f})$, which incorporates the result

of two classical processor-days of computation, would on such occasions have been performed by one processor in one day.

One physical way of describing this effect is that when the subtasks of an N-fold parallel task are delegated to N^2-2N+2 universes, at most one of them can acquire the overall result.

(2) Now consider the problem of the design of parallel information-processing systems which are subject to noise. For example, suppose that it is required, within a fixed time τ, to compute a certain N-fold parallelizable function $G(\boldsymbol{f})$. NR processors are available, each of which may fail for reasons of thermal noise, etc. with probability p. For simplicity assume that such a hardware error can be reliably detected. The problem is to minimize the overall failure rate q. 'Classically' (i.e. without using quantum parallelism) one minimizes q by means of an R-fold redundancy: R processors are instructed to perform each of the N parallel subtasks. The machine as a whole will therefore fail to compute the result in time only when all R processors assigned to any one subtask fail, and this occurs with probability

$$q_{\text{classical}} = 1-(1-p^R)^N. \tag{3.24}$$

Using quantum parallelism, however, each of the NR available processors may be given all N tasks. Each is subject to two independent causes of failure, (i) the probability p that it will fail for hardware reasons, and (ii) the probability, which as I have indicated will for certain $G(\boldsymbol{f})$ be $1-(N-2N+2)^{-1}$, that it will end up in a different universe from the answer. It takes only one of the NR processors to succeed, so the failure rate is

$$q_{\text{quantum}} = [1-(N^2-2N+2)^{-1}(1-p)]^{NR}, \tag{3.25}$$

a number which, for suitable values of p, N and R, can be smaller than (3.24).

Faster computers

One day it will become technologically possible to build quantum computers, perhaps using flux quanta (Likharev 1982; Leggett 1985) as the fundamental components. It is to be expected that such computers could operate at effective computational speeds in excess of Turing-type machines built with the same technology. This may seem surprising since I have established that no recursive function can be computed by $\mathcal{Q}$ on average more rapidly with the help of quantum programs than without. However, the idealizations in $\mathcal{Q}$ take no account of the purely technological fact that it is always easier in practice to prepare a very large number of identical systems in the same state than to prepare each in a different state. It will therefore be possible to use a far higher degree of redundancy R for parallel quantum programs than for classical ones running on the same basic hardware.

Interpretational implications

I have described elsewhere (Deutsch 1985; cf. also Albert 1983) how it would be possible to make a crucial experimental test of the Everett ('many-universes') interpretation of quantum theory by using a quantum computer (thus contradicting the widely held belief that it is not experimentally distinguishable from other interpretations). However, the performance of such experiments must await both the construction of quantum computers and the development of true artificial-intelligence programs. In explaining the operation of quantum computers I have, where necessary, assumed Everett's ontology. Of course the explanations could always be 'translated' into the conventional interpretation, but not without entirely losing their explanatory power. Suppose, for example, a quantum computer were programmed as in the Stock Exchange problem described. Each day it is given different data. The Everett interpretation explains well how the computer's behaviour follows from its having delegated subtasks to copies of itself in other universes. On the days when the computer succeeds in performing two processor-days of computation, how would the conventional interpretations explain the presence of the correct answer? *Where was it computed?*

IV. Further connections between physics and computer science

Quantum complexity theory

Complexity theory has been mainly concerned with constraints upon the computation of functions: which functions can be computed, how fast, and with use of how much memory. With quantum computers, as with classical stochastic computers, one must also ask 'and with what probability?'. We have seen that the minimum computation time for certain tasks can be lower for $\mathcal{Q}$ than for $\mathcal{T}$. Complexity theory for $\mathcal{Q}$ deserves further investigation.

The less immediately applicable but potentially more important application of complexity theory has been in the attempt to understand the spontaneous growth of complexity in physical systems, for example the evolution of life, and the growth of knowledge in human minds. Bennett (1983) reviewed several different measures of complexity (or 'depth', or 'knowledge') that have been proposed. Most suffer from the fatal disadvantage that they assign a high 'complexity' to a purely random state. Thus they do not distinguish true knowledge from mere information content. Bennett has overcome this problem. His 'logical depth' is roughly the running time of the shortest $\mathcal{T}$-program that would compute a given state ψ from a blank input. Logical depth is at a minimum for random states. Its intuitive physical justification is that the 'likeliest explanation' why a physical system might be found to be in the state ψ is that ψ was indeed 'computed' from that shortest $\mathcal{T}$-program. In biological terminology, logical depth measures the amount of evolution that was needed to evolve ψ from the simplest possible precursors.

At first sight Bennett's construction seems to lose this physical justification when it is extended beyond the strictly deterministic physics of Turing machines. In physical reality most random states are not generated by 'long programs' (i.e. precursors whose complexity is near to their own), but by short programs relying on indeterministic hardware. However, there is a quantum analogue of Bennett's idea which solves this problem. Let us define the Q-logical depth of a quantum state as the running time of the shortest $\mathcal{Q}$-program that would generate the state from a blank input (or, perhaps, as Bennett would have it, the harmonic mean of the running times of all such programs). Random numbers can be rapidly generated by *short* $\mathcal{Q}$-programs.

Notice that the Q-logical depth is not even in principle an observable, because it contains information about all universes at once. But this makes sense physically: the Q-logical depth is a good measure of knowledge in that it gives weight only to complexity that is present in all universes, and can therefore be assumed to have been put there 'deliberately' by a deep process. Observationally complex states that are different in different universes are not truly deep but just random. Since the Q-logical depth is a property of the quantum *state* (vector), a quantum subsystem need not necessarily have a well defined Q-logical depth (though often it will to a good degree of approximation). This is again to be expected since the knowledge in a system may reside entirely in its correlations with other systems. A spectacular example of this is quantum cryptography.

Connections between the Church–Turing principle and other parts of physics

We have seen that quantum theory obeys the strong form (1.2) of the Church–Turing principle only on the assumption that the third law of thermodynamics (1.3) is true. This relation is probably better understood by considering the Church–Turing principle as more fundamental and deriving the third law from it and quantum theory.

The fact that classical physics does not obey (1.2) tempts one to go further. Some of the features that distinguish quantum theory from classical physics (for example the discreteness of observables?) can evidently be derived from (1.2) and the laws of thermodynamics alone. The new principle has therefore given us at least part of the solution to Wheeler's problem 'Why did quantum theory have to be?' (see, for example, Wheeler 1985).

Various 'arrows of time' that exist in different areas of physics have by now been connected and shown to be different manifestations of the same effect. But, contrary to what is often asserted, the 'psychological' or 'epistemological' arrow of time is an exception. Before Bennett (1973) it could be maintained that computation is intrinsically irreversible, and since psychological processes such as the growth of knowledge are computations, the psychological arrow of time is necessarily aligned with the direction in which entropy increases. This view is now untenable, the alleged connection fallacious.

One way of reincorporating the psychological arrow of time into physics is to postulate another new principle of Nature which refers directly to the Q-logical depth. It seems reasonable to assert, for example, that the Q-logical depth of the universe is at a minimum initially. More optimistically the new principle might require the Q-logical depth to be non-decreasing. It is perhaps not unreasonable to hope that the second law of thermodynamics might be derivable from a constraint of this sort on the Q-logical depth. This would establish a valid connection between the psychological (or epistemological, or evolutionary) and thermodynamic 'arrows of time'.

Programming physics

To view the Church–Turing hypothesis as a physical principle does not merely make computer science a branch of physics. It also makes part of experimental physics into a branch of computer science.

The existence of a universal quantum computer $\mathcal{Q}$ implies that there exists a program for each physical process. In particular, $\mathcal{Q}$ can perform any physical experiment. In some cases (for example measurement of coupling constants or the form of interactions) this is not useful because the result must be known to write the program. But, for example, when testing quantum theory itself, every experiment is genuinely just the running of a $\mathcal{Q}$-program. The execution on $\mathcal{Q}$ of the following ALGOL 68 program *is* a performance of the Einstein–Podolski–Rosen experiment:

```
begin
  int n = 8 * random;      ¢ random integer from 0 to 7 ¢
  bool x, y;               ¢ bools are 2-state memory elements ¢
  x := y := false;         ¢ an irreversible preparation ¢
  V(8, y);                 ¢ see equation (2.15) ¢
  x eorab y;               ¢ perfect measurement (2.14) ¢
  if V(n, y) ≠             ¢ measure y in random direction ¢
    V(n, x)                ¢ and x in the parallel direction ¢
    then print (("Quantum theory refuted."))
    else print (("Quantum theory corroborated."))
  fi
end
```

Quantum computers raise interesting problems for the design of programming languages, which I shall not go into here. From what I have said, programs exist that would (in order of increasing difficulty) test the Bell inequality, test the linearity of quantum dynamics, and test the Everett interpretation. I leave it to the reader to write them.

I wish to thank Dr C. H. Bennett for pointing out to me that the Church–Turing hypothesis has physical significance, C. Penrose and K. Wolf for interesting discussions about quantum computers, and Professor R. Penrose, F.R.S., for reading an earlier draft of the article and suggesting many improvements.

This work was supported in part by N.S.F. grant no. PHY 8205717.

References

Albert, D. Z. 1983 *Phys. Lett.* A **98**, 249.
Bekenstein, J. D. 1973 *Phys. Rev.* D **7**, 2333.
Bekenstein, J. D. 1981 *Phys. Rev.* D **23**, 287.
Bell, J. S. 1964 *Physica* **1**, 195.
Benioff, P. A. 1982 *Int. J. theor. Phys.* **21**, 177.
Bennett, C. H. 1973 *IBM Jl Res. Dev.* **17**, 525.
Bennett, C. H. 1981 *SIAM Jl Comput.* **10**, 96.
Bennett, C. H. 1983 On various measures of complexity, especially 'logical depth'. Lecture at Aspen. IBM Report.
Bennett, C. H., Brassard, G., Breidbart, S. & Wiesner, S. 1983 Advances in cryptography. In *Procedings of Crypto 82.* New York: Plenum.
Chaitin, G. J. 1977 *IBM Jl Res. Dev.* **21**, 350.

Church, J. 1936 *Am. J. Math.* **58**, 435.
Deutsch, D. 1985 *Int. J. theor. Phys.* **24**, 1.
d'Espagnat, B. 1976 *Conceptual foundations of quantum mechanics* (second edn). Reading, Massachusetts: W. A. Benjamin.
Feynman, R. P. 1982 *Int. J. theor. Phys.* **21**, 467.
Gandy, R. 1980 In *The Kleene symposium* (ed. J. Barwise, H. J. Keisler & K. Kunen), pp. 123–148. Amsterdam: North Holland.
Hofstadter, D. R. 1979 *Gödel, Escher, Bach: an eternal golden braid.* New York: Random House.
Leggett, A. J. 1985 In *Quantum discussions, proceedings of the Oxford quantum gravity conference 1984* (ed. R. Penrose & C. Isham). Oxford University Press.
Likharev, K. K. 1982 *Int. J. theor. Phys.* **21**, 311.
Popper, K. R. 1959 *The logic of scientific discovery.* London: Hutchinson.
Toffoli, T. J. 1979 *J. Comput. Syst. Sci.* **15**, 213.
Turing, A. M. 1936 *Proc. Lond. math. Soc. Ser.* 2, **442**, 230.
Wheeler, J. A. 1985 In *NATO Advanced Study Institute Workshop on Frontiers of Nonequilibrium Physics 1984.* New York: Plenum.