Proc. R. Soc. Lond. A **425**, 73–90 (1989)
Printed in Great Britain

Quantum computational networks

By D. Deutsch

Oxford University Mathematical Institute, 24–29 *St Giles*, *Oxford OX*1 3*LB*, *U.K.*

(*Communicated by R. Penrose, F.R.S. – Received* 8 *July* 1988)

The theory of quantum computational networks is the quantum generalization of the theory of logic circuits used in classical computing machines. Quantum gates are the generalization of classical logic gates. A single type of gate, the univeral quantum gate, together with quantum 'unit wires', is adequate for constructing networks with any possible quantum computational property.

Computations

A *computation* is a process that produces *outputs* that depend in some desired way on given *inputs*. In one sense, inputs and outputs are abstract symbols that may or may not refer to anything concrete. Some computations refer to purely mathematical objects like prime numbers, whereas others refer to physical systems like tomorrow's weather. But in another sense, in any computation that is actually performed the inputs and outputs must be concrete. Regardless of what they are interpreted as referring to, they themselves are states of physical objects, and computations are physical processes. A *computing machine* is a physical object whose motion can be regarded as the performance of a computation. The inputs are identified with possible ways in which the machine can be prepared before the motion, and the outputs with possible results of a fixed measurement performed after the motion.

A physical object $\mathscr{S}$ is used as a computing machine in the following way (this description is phrased in quantum terminology but applies also to classical computers).

A complete set $\mathbf{C}$ of compatible observables for $\mathscr{S}$ is chosen, including an input observable $\hat{I}$ and an observable $\hat{h}$ that is independent of $\hat{I}$ and has spectrum $\{0, 1\}$, the *halt flag*. A distinct eigenvalue of $\hat{I}$ is designated to represent each of the possible inputs for the computation that $\mathscr{S}$ is required to perform.

An output observable $\hat{O}$ is chosen usually, but not necessarily, from $\mathbf{C}$, though it must at least be compatible with the halt flag, $\hat{h}$. A distinct eigenvalue of $\hat{O}$ is designated to represent each of possible outputs of the computation.

For $\mathscr{S}$ to perform its computation for a specific input, $\hat{I}$ is prepared with the eigenvalue corresponding to that input, $\hat{h}$ is prepared with the value '0', meaning 'not halted', and the remaining observables in $\mathbf{C}$ are prepared with some fixed set of standard values. $\mathscr{S}$ then evolves according to its internal dynamics, remaining isolated, except that $\hat{h}$ is measured at suitable intervals by an external non-perturbing measuring instrument. When the result of one of these measurements is '**1**', meaning 'halted', $\hat{O}$ is immediately measured and the result of that measurement is interpreted as the output of the computation.

Bits

A *bit* is the smallest possible quantity of non-probabilistic information. Bits arise naturally in both the theory and the practice of computation. This is already true in the classical case, even though discrete-spectrum observables are quite alien to classical physics. In quantum theory, discrete spectra are the rule. This is one of several ways in which the quantum theory of computation is more natural than the classical theory. The very word 'quantum' means 'indivisible chunk', which is exactly what 'bit' means too. The simplest possible quantum system, a *2-state system* such as the spin of a spin-$\frac{1}{2}$ particle, can hold exactly one bit of information.

For this reason, the term 'bit' is also used, by extension, to denote a 2-state physical system. A larger object, such as a spin-$\frac{1}{2}$ atom or even an electronic circuit element, that contains a 2-state subsystem may with a further extension of terminology also be called a 'bit'. Where it is necessary to be more precise, I shall refer to the larger object as the *carrier* of the bit.

It is convenient to restrict our attention to computations that are discrete in another sense also, namely those that proceed in a sequence of *computational steps* of fixed duration. Input and output, and the measurement of the halt flag, occur only at the beginning or end of computational steps, and never during one. Thus the duration of a computation is always a whole number of computational steps. No relevant generality is lost by this idealization.

Logic gates and quantum gates

In the classical theory of computation a *logic gate* is a computing machine whose input and output consist of fixed numbers of bits and which performs a fixed computation in a fixed time that is independent of the input. Thus the halt flag for a gate need not have any dynamical connection with the rest of the gate; it might be an external 'clock', for example.

A *quantum gate* is defined in a similar way. The only difference is that the states of the input and output of a quantum gate are not required to be eigenstates of $\hat{I}$ and $\hat{O}$, but may be arbitrary states of the input and output bits (or quantum mixtures of states). The eigenstates of $\hat{I}$ form a basis, the *input basis*, spanning the space of all possible (pure) input states of the gate. Similarly, the eigenstates of $\hat{O}$ form the *output basis*, spanning the space of output states. In general the output of a quantum gate will be a superposition of output basis states even if the input is prepared in an input basis state, though for a given input state the output state will always be the same superposition.

The computation performed by a logic gate may be shown on a diagram in which the gate is represented by a rectangle, each input bit by a line attached to the rectangle on the left, and each output bit by a line attached to the rectangle on the right. The bits are labelled to show how the value of each output bit depends on the values of the input bits. The halt flag is not shown explicitly. For example, figure 1 indicates that the *exclusive-or* gate has two input bits a and b, and one output bit which the gate sets to the value $a \oplus b$, where '$\oplus$' denotes the 'exclusive-or' operation, or addition modulo 2.

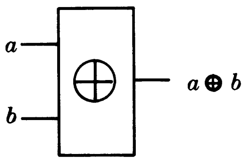

FIGURE 1. *Exclusive-or* gate.

A logic gate G is said to be *reversible* if its inputs and corresponding outputs are related by an invertible function. Thus the *exclusive-or* gate is *irreversible*. If a gate G is reversible then there exists another gate G^{-1}, the *inverse gate* of G, which places bits in any state that can be an output of G into the corresponding input state.

Most of my discussion concerns reversible gates with the same number of output bits as input bits. Any gate can be regarded as a restriction of such a gate, where some of the output bits are discarded and/or some of the input bits are given fixed values. For example, the *exclusive-or* gate is a restriction of the *measurement gate* (figure 2). The measurement gate is reversible, being its own inverse, and has two inputs and two outputs.

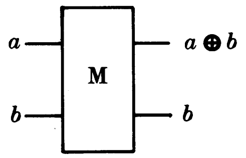

FIGURE 2. Measurement gate.

It is so called because if the value a of the upper input bit is zero, the upper bit is in effect an apparatus that performs a perfectly accurate non-perturbing measurement of the lower one. Bennett (1973) has shown that both the classical theory of computability and classical complexity theory are essentially the same for reversible computing machines as for the more familiar irreversible ones. In the quantum case some computations (those relying on quantum coherence) can be performed only by machines all of whose components are reversible.

When a gate has the same number of output bits as input bits it is possible for the output bits to be the same physical objects as the input bits, but at a later time, and for the input and output observables to be the same (in the Schrödinger picture). The input and output bases then coincide and can be called the *computational basis*. It is convenient to speak of the bits as 'moving through' the gate, though it is really the carriers of the bits that move. The motion of a bit, *qua* 2-state system, is confined to its two basis states and superpositions and mixtures thereof.

The effect of an n-bit reversible logic gate is to perform a permutation on the 2^n possible states of the bits that pass through it. Thus an alternative way of specifying a reversible logic gate is to specify the permutation that it performs. For example, let

$$\{|a,b\rangle\} \quad (a,b \in \{0,1\}) \tag{1}$$

be the four computational basis states of the two input (or output) bits of the measurement gate. Here the labels in the ket, reading from left to right, correspond to the bits reading from top to bottom in the diagram. Then the gate performs the permutation

$$\left.\begin{array}{l}|0,0\rangle \Rightarrow |0,0\rangle \\ |0,1\rangle \Rightarrow |1,1\rangle \\ |1,0\rangle \Rightarrow |1,0\rangle \\ |1,1\rangle \Rightarrow |0,1\rangle\end{array}\right\}. \tag{2}$$

The symbol '$\Rightarrow$' may be read as 'evolves into'. It means that if the bits are prepared in the state on the left of the symbol, then after a fixed time they will be in the state on the right.

This leads us to a third way of specifying a gate. Consider the 4×4 matrix S with components

$$\mathsf{S}^{ab}_{a'b'} = \delta^{a\oplus b}_{a'}\,\delta^{b}_{b'} \tag{3}$$

(where '$a'b'$' and 'ab' are 'clumped indices' for S). The effect of the measurement gate on bits in the input state $|a, b\rangle$ is

$$|a,b\rangle \Rightarrow \sum_{a',b'\in\{0,1\}} \mathsf{S}^{ab}_{a'b'}\,|a',b'\rangle \equiv \mathsf{S}|a,b\rangle. \tag{4}$$

I shall refer to S as a 'matrix' even in cases, such as the right-hand side of the identity in (4), where it appears as a linear operator on a quantum state space and we are not considering its components in a particular basis.

Specifying S would be an extravagant way of defining the measurement gate, or any logic gate. It would be a very sparse and simple matrix. For S to describe a permutation, every element would have to be a '1' or a '0', with exactly one '1' in each row and column. But for S to describe the computation performed by a reversible *quantum* gate it need only satisfy a far less restrictive condition, namely that it be *unitary*. It is the *S-matrix* for the computation. The computation performed by an n-bit reversible quantum gate is an n-particle elastic quantum scattering process described by a $2^n \times 2^n$ S-matrix. Of the three methods I have given for specifying logic gates, only the third, the S-matrix method, is applicable to general quantum gates.

If a gate G has the S-matrix S_{G}, then the inverse gate G^{-1} has the S-matrix $\mathsf{S}^{\dagger}_{\mathrm{G}}$, the hermitian conjugate of S_{G}. Irreversible gates are described by superscattering matrices rather than by S-matrices, but they will not concern us in this paper.

I shall call two gates *computationally distinct* whenever their S-matrices, referred to bases of correspondingly labelled states of the gates' inputs and outputs, are distinct up to multiplication by a phase factor, and I shall call them *computationally equivalent* otherwise. This would not be a sufficiently discriminating definition of computational equivalence for general computing machines, because it depends only on the relations between the inputs and outputs and not on other important properties such as the time required to perform computations. But it is an appropriate definition in the case of gates. For example, the speeds of two gates that are computationally equivalent by this definition can at worst be different by a fixed factor independent of the input.

There are only four computationally distinct logic gates with one input bit and one output bit, and only two of those are reversible. One of them is the unit gate, or *unit wire* I, whose S-matrix is the unit matrix, I. The other is the *not* gate, N (figure 3), which has the S-matrix

$$S_N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{5}$$

By contrast, there are many computationally distinct 1-bit reversible quantum gates (one for every element of the continuous group SU(2), to be exact). Consider, for example, the family of gates $\{N^\alpha\}$ whose S-matrices are the αth powers of S_N

$$S_{N^\alpha} = S_N^\alpha = \frac{1}{2}\begin{pmatrix} 1+e^{i\pi\alpha}, & 1-e^{i\pi\alpha} \\ 1-e^{i\pi\alpha}, & 1+e^{i\pi\alpha} \end{pmatrix}, \tag{6}$$

where α is a real parameter. When α is an integer, N^α is a logic gate, namely either a unit wire or a *not* gate. For all other α, the $\{N^\alpha\}$ are quantum gates with no classical analogue. They perform computations that are powers and roots of '*not*'.

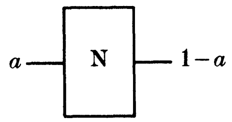

FIGURE 3. *Not* gate.

QUANTUM NETWORKS

Complex gates can be constructed out of simple ones by using the outputs of some gates as the inputs of others. Physically, this could be done by moving the bits from one gate to another without changing their state. More precisely, they must be moved without changing their *computational* state; their carriers can of course be made to undergo any convenient motion so long as it is independent of the computational state. The resulting connections between gates are called unit wires because they compute the identity function. As remarked above a unit wire is also a gate, I. It is represented diagramatically as a line, usually connecting the output and input of other gates.

Sometimes it is necessary for the proper operation of a network that two or more unit wires be connected in series to delay bits travelling down them (even though this is 'computationally equivalent' to one unit wire). The symbol shown in figure 4 represents n unit wires in series.

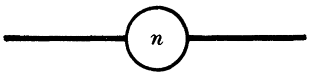

FIGURE 4. n unit wires in series.

Bits move from left to right along unit wires and through gates unless arrows indicate otherwise.

A *logic circuit* is a computing machine consisting of logic gates whose computational steps are synchronized (that is, they all begin and end at the same instants). The outputs of some of the gates are connected by unit wires to the inputs of others. Some or all of the inputs that are not connected to outputs are used as the input of the network. The remaining inputs are connected to *sources*.

Sources of '0' bits or of '1' bits (figure 5) are gates with no input and one output which, once every computational step, emit a bit with value '0' or '1', respectively. Some of the outputs that are not connected to inputs are used as the output of the network. If a halt flag is necessary, one of the outputs is used for that purpose. The remaining outputs are connected to *sinks* (figure 6), where any arriving bits are discarded (or perhaps reprepared with values '0' or '1' and recycled). Sources are *reversible* gates and are not the inverses of sinks, which are irreversible. This is because a sink can absorb both '0' bits and '1' bits, whereas the inverse of a source of '0' bits is a gate whose only permitted inputs are '0' bits.

FIGURE 5. (*a*) Source of '0' bits, (*b*) source of '1' bits.

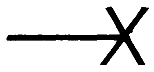

FIGURE 6. Sink.

A *quantum computational network* is the quantum generalization of a logic circuit. It consists of quantum gates, sources, sinks and unit wires. A quantum network may or may not itself be a gate. If its component gates can be numbered so that no input of any gate with a given number is connected to any ouput from any gate with an equal or higher number, then the network, possibly with its output bits delayed so that they emerge simultaneously, is necessarily a gate. If this condition is not satisfied, and some of the outputs of gates 'loop back', then the network may or may not have properties that disqualify it from being a gate. For example, the time taken before the computation halts, if it ever does halt, may depend on the value of the input; or after the gate has performed a computation on one input its effect on the identical input presented to it again may be different. That is to say, in computing terminology, that a network will in general have a memory but a gate does not.

At the beginning of each computation performed by the network, the bits constituting the input observable $\hat{I}$ are prepared appropriately, and a '0' bit is placed at every remaining input of every gate in the network, including unit wires.

Unit wires do not bifurcate. If two or more input bits of gates need to be set to the value of a single bit, then this value must be physically copied first. A bifurcation point where one unit wire splits into two is a non-trivial gate, the *fan-out* gate F, which might be constructed as in figure 7.

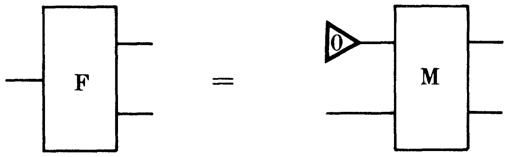

FIGURE 7. *Fan-out* gate.

Some computations can be performed by unit wires alone. These computations are *bit permutations.* For example, the permutation

$$|a,b\rangle \Rightarrow |b,a\rangle \tag{7}$$

is performed by the *swap* gate (figure 8).

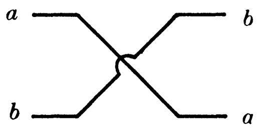

FIGURE 8. *Swap* gate.

Mere permutations of *bits* must not be confused with permutations of states, of which they form a proper subset. There are only $n!$ distinct permutations of n bits, but there are $(2^n)!$ distinct permutations of the 2^n classical states of n bits. As mentioned above, the state permutations correspond precisely to the classical reversible computations.

On emerging from a gate (other than a unit wire) a bit must either leave the network or enter one or more consecutive unit wires *en route* to another gate. Moreover, the route that a particular bit takes through a network is independent of what computation is being performed. Thus a bit in a computational network undergoes two different sorts of motion in alternation: when it is in a unit wire its computational degree of freedom (i.e. the bit proper) is inert and only the carrier moves. When it is inside any other gate it interacts with the other bits that are there at the same time, in a scattering process in which only the computational degrees of freedom participate. Such a scattering process lasts at most one computational step and is always followed by another inert period of at least one step.

ADEQUATE SETS OF GATES

In the propositional calculus a set of logical connectives is said to be *adequate* if any truth function is expressible using connectives only from that set. The operations implicit in the term 'expressible' are, in our terminology, those performed by unit wires (function composition), bifurcating wires (repeated use of a propositional variable), and sources (truth constants). Logical connectives correspond to 1- or 2-input, 1-output logic gates.

This definition of adequate sets can be adapted to arbitrary logic gates as follows: a set **G** of logic gates is *adequate* relative to another set **H** of gates if for every gate $\mathrm{H} \in \mathbf{H}$ there exists a logic network computationally equivalent to H that can be constructed entirely out of elements of **G**.

For example, the set consisting of a source of '1' bits, a unit wire and a measurement gate, is adequate relative to the set of all 2-bit reversible logic gates.

This definition of 'adequate' is not useful, as it stands, for sets of quantum gates. The set of all logic gates is a discrete set, whereas the set of quantum gates is a continuum†. It is possible for every member of a set **H** of gates to be approximated arbitrarily closely by quantum computational networks constructed out of elements of some set **G** of quantum gates, and yet for no such network to be exactly computationally equivalent to any member of **H**. This is the case, for example, if

$$\mathbf{H} = \{\mathrm{N}\} \quad \text{and} \quad \mathbf{G} = \{\mathrm{N}^{\alpha}, \mathrm{I}\}, \tag{8}$$

where α is irrational. The only way in which two or more N^{α} gates can be connected together by unit wires to form a 1-input, 1-output network is in series. From (6), the S-matrix of a network consisting of m copies of the N^{α} gate connected in series is

$$(S_{\mathrm{N}^{\alpha}})^{m} = S_{\mathrm{N}}^{m\alpha} = S_{\mathrm{N}}^{m\alpha - 2\lfloor\frac{1}{2}m\alpha\rfloor}, \tag{9}$$

because $S_{\mathrm{N}}^{2} = I$. Now, although this shows that no network of elements of **G** can be computationally equivalent to N, because the exponent $m\alpha - 2\lfloor\frac{1}{2}m\alpha\rfloor$ can never be 1 if α is irrational, it also shows that such networks can be arbitrarily close to being computationally equivalent to N. Given any $\epsilon > 0$, there exists an integer m such that $m\alpha - 2\lfloor\frac{1}{2}m\alpha\rfloor$ is between 1 and $1+\epsilon$. For that m, the probability of detecting experimentally, by preparing the input bit in some state $|\psi\rangle$ and measuring the output, that the network of m N^{α} gates is not in fact an N gate, is at most

$$\begin{aligned}\max_{|\psi\rangle}(1 - |\langle\psi|\, S_{\mathrm{N}}^{\dagger}(S_{\mathrm{N}^{\alpha}})^{m}\,|\psi\rangle|^{2}) &= \max_{|\psi\rangle}(1 - |\langle\psi|\, S_{\mathrm{N}}^{\epsilon}|\psi\rangle|^{2}) \\ &= \sin^{2}\tfrac{1}{2}\pi\epsilon \end{aligned} \tag{10}$$

on each occasion that the experiment is performed. If the network were used in place of a *not* gate in some larger network, then the mean time between failures caused by this substitution, in units of the larger network's computational steps, would be $\operatorname{cosec}^{2}\frac{1}{2}\pi\epsilon$, which is arbitrarily large. For sufficiently small ϵ this will be an improvement, in tems of reliability, on any other fixed way of constructing a *not* gate. It is true that a *not* gate cannot be constructed out of elements of **G**. But this is true in exactly the same unphysical sense as the proposition that the *not* gate cannot be constructed at all. It is after all impossible, because of the third law of thermodynamics, to construct a perfectly reliable gate.

For this reason it is desirable to classify the set $\{\mathrm{N}^{\alpha}, \mathrm{I}\}$ as adequate relative to $\{\mathrm{N}\}$. In general I shall classify a set **G** of quantum gates as adequate relative to a set **H** of gates if for every $\mathrm{H} \in \mathbf{H}$ there exists a sequence $\{\mathrm{K}_{n}\}$ of gates, each of

† It is curious that in this one respect, passing from the classical to the quantum theory of computation takes us from the discrete to the continuous, rather than vice versa as is usual. The reason is that we are not comparing like with like. It is not customary, in the classical theory of computation, actually to state what sort of processes within classical physics might correspond to discrete computations. When this is done, as, for example, by Toffoli (1981), the classical continuum reappears.

which is a quantum network that can be constructed entirely out of elements of **G**, and a sequence $\{\phi_n\}$ of phase angles, such that

$$\lim_{n\to\infty} \mathrm{e}^{\mathrm{i}\phi_n} S_{\mathrm{K}_n} = S_{\mathrm{H}}. \tag{11}$$

Universal gates

If a **G** is an adequate set of gates relative to **H**, and the elements of **G** are precisely the unit wire, sources of '0' and '1' bits, and one other gate U, then U is a *universal gate* relative to **H**.

For example, we have just seen that N^α is universal relative to $\{\mathrm{N}\}$. The *nand/not* gate (figure 9), a 2-bit irreversible logic gate, is universal relative to the set of all logic gates. The *nor* operation, provided by the first output alone, is a universal *connective*, but is not a universal logic gate because it cannot be used to construct bifurcating wires.

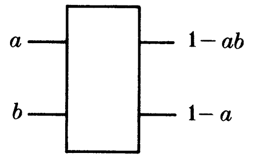

FIGURE 9. *Nand/not* gate.

Toffoli (1981) has shown that the 3-bit reversible logic gate T, defined in figure 10, is also universal relative to the set of all logic gates. In terms of state permutations the effect of the Toffoli gate is as simple as can be. Six of the eight computational basis states $\{|a,b,c\rangle\}$ evolve into themselves, and the other two, $|1,1,0\rangle$ and $|1,1,1\rangle$, evolve into each other. The S-matrix is correspondingly simple:

$$S^{abc}_{\mathrm{T}a'b'c'} = \delta^a_{a'}\,\delta^b_{b'}\,\delta^{(c\oplus ab)}_{c'}. \tag{12}$$

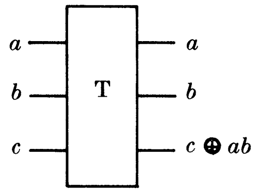

FIGURE 10. Toffoli gate.

It is convenient to express this in the form

$$S^{abc}_{\mathrm{T}a'b'c'} = \delta^a_{a'}\,\delta^b_{b'}[(1-ab)\,\delta^c_{c'} + ab\mathrm{S}^c_{\mathrm{N}c'}], \tag{13}$$

where $\mathrm{S_N}$ is the S-matrix of the *not* gate, defined above. I now show that the 3-bit quantum gate **Q**, whose S-matrix is

$$S^{abc}_{\mathrm{Q}a'b'c'} = \delta^a_{a'}\,\delta^b_{b'}[(1-ab)\,\delta^c_{c'} + \mathrm{i}ab\,\mathrm{e}^{-\frac{1}{2}\mathrm{i}\pi\alpha}(\mathrm{S}^\alpha_{\mathrm{N}})^c_{c'}], \tag{14}$$

where α is any irrational number, is a universal quantum gate, i.e. it is universal relative to the set of all quantum computational gates.

Proof that Q is a universal quantum gate

The method of proof is to build a repertoire of networks for which **Q** is adequate, showing successively that this includes the Toffoli gate T, all logic networks, all 3-bit quantum gates, all n-bit quantum gates, and finally all quantum computational networks.

Let us relabel the elements of the computational basis $\mathscr{B}$ for **Q** with integers from 0 to 7, interpreting the values of the three bits as digits of a binary number so that $|1,1,1\rangle \equiv |7\rangle$, $|1,1,0\rangle \equiv |6\rangle$, etc. In this basis, if n is an integer,

$$\mathsf{S}_{\mathrm{Q}}^{4n+1} = \begin{pmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&& \mathrm{i}\cos[(2n+\frac{1}{2})\pi\alpha] & \sin[(2n+\frac{1}{2})\pi\alpha] \\ &&&&&& \sin[(2n+\frac{1}{2})\pi\alpha] & \mathrm{i}\cos[(2n+\frac{1}{2})\pi\alpha] \end{pmatrix}. \quad (15)$$

Given any $\epsilon > 0$, there exist positive integers n and m such that $(2n+\frac{1}{2})\pi\alpha$ is within ϵ of $(2m+\frac{1}{2})\pi$, the values for which $\mathsf{S}_{\mathrm{Q}}^{4n+1}$ would be equal to S_{T}. An argument similar to that which showed that N^{α} is a universal gate relative to $\{\mathrm{N}\}$ now shows that the Toffoli gate T is in our repertoire, and it follows immediately that all logic gates are in the repertoire too.

Similarly because

$$\mathsf{S}_{\mathrm{Q}}^{4n} = \begin{pmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&& \cos(2n\pi\alpha) & -\mathrm{i}\sin(2n\pi\alpha) \\ &&&&&& -\mathrm{i}\sin(2n\pi\alpha) & \cos(2n\pi\alpha) \end{pmatrix}, \quad (16)$$

all gates whose S-matrices are of the form,

$$U_{\lambda} \equiv \begin{pmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&& \cos\lambda & \mathrm{i}\sin\lambda \\ &&&&&& \mathrm{i}\sin\lambda & \cos\lambda \end{pmatrix}, \quad (17)$$

for any real λ, are in the repertoire.

Let P_{mn} be the permutation of $\mathscr{B}$ that interchanges $|m\rangle$ and $|n\rangle$ and leaves the other six elements of $\mathscr{B}$ unchanged. The $\{P_{mn}\}$ are S-matrices of logic gates, and are therefore in the repertoire. Now, for small λ,

$$P_{56}(U_\lambda P_{57})^2 (U_{-\lambda} P_{57})^2 P_{56} = I + \begin{pmatrix} 0 &&&&&&& \\ & 0 &&&&&& \\ && 0 &&&&& \\ &&& 0 &&&& \\ &&&& 0 &&& \\ &&&&& 0 && \\ &&&&&& 0 & \lambda^2 \\ &&&&&& -\lambda^2 & 0 \end{pmatrix} + O(\lambda^3). \quad (18)$$

Therefore, the repertoire must also include all gates with S-matrices of the form

$$\lim_{n\to\infty} \{P_{56}(U_{\surd(\lambda/n)} P_{57})^2 (U_{-\surd(\lambda/n)} P_{57})^2 P_{56}\}^n = \begin{pmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&& \cos\lambda & \sin\lambda \\ &&&&&& -\sin\lambda & \cos\lambda \end{pmatrix} \equiv V_\lambda. \quad (19)$$

Similarly, it must include all gates with S-matrices of the form

$$\lim_{n\to\infty} (U_{\surd(\lambda/2n)} V_{\surd(\lambda/2n)} U_{-\surd(\lambda/2n)} V_{-\surd(\lambda/2n)})^n = \mathrm{diag}\,(1,1,1,1,1,1,\mathrm{e}^{-\mathrm{i}\lambda},\mathrm{e}^{\mathrm{i}\lambda}) \equiv W_\lambda. \quad (20)$$

Matrices of the form $P_{ab} W_\lambda P_{ab}$ are also diagonal and have the same components as W_λ, but in a different order. Therefore, all S-matrices that are diagonal in the computational basis can be represented, up to a phase factor, as products of such permuted versions of W_λ. The corresponding gates, and in particular gates with S-matrices

$$X_\lambda \equiv \mathrm{diag}\,(1,1,1,1,1,1,1,\mathrm{e}^{\mathrm{i}\lambda}), \quad (21)$$

which change the phase of only one of the basis states, are in the repertoire.

Now consider a general state $|\psi\rangle$ of three bits. This can be expanded in terms of the computational basis:

$$|\psi\rangle = \sum_{n=0}^{7} c_n|n\rangle, \quad \text{where} \quad \sum_{n=0}^{7} |c_n|^2 = 1. \quad (22)$$

If bits in the state $|\psi\rangle$ with $c_6 \neq 0$ pass through a gate with S-matrix

$$Z_6[|\psi\rangle] = X_{-\frac{1}{2}\mathrm{Arg}\,(c_6 c_7)} V_{-\arctan|c_6/c_7|} W_{-\frac{1}{2}\mathrm{Arg}\,(c_7/c_6)}, \quad (23)$$

which is manifestly in the repertoire, they undergo the evolution

$$|\psi\rangle \Rightarrow \sum_{n=0}^{5} c_n|n\rangle + \surd(|c_6|^2+|c_7|^2)\,|7\rangle, \quad (24)$$

after which the coefficient of the basis state $|6\rangle$ has gone to zero.

Gates $Z_5[|\psi\rangle], \ldots, Z_0[|\psi\rangle]$, which succesively evolve to zero the coefficients of the states $|5\rangle, \ldots, |0\rangle$, can be constructed analogously within the repertoire. Therefore, the gate $G[|\psi\rangle]$ with S-matrix

$$S_{G[|\psi\rangle]} = \prod_{n=0}^{6} Z_n[|\psi\rangle] \tag{25}$$

evolves to zero the coefficients of the seven kets $|0\rangle$ to $|6\rangle$, and evolves to 1 the coefficient of $|7\rangle$. Thus for any state $|\psi\rangle$ there is a gate $G[|\psi\rangle]$ in the repertoire that will cause three bits initially in the state $|\psi\rangle$ to undergo the evolution

$$|\psi\rangle \Rightarrow |7\rangle. \tag{26}$$

Similarly, the gate $G^{-1}[|\psi\rangle]$ is also in the repertoire. Now, a general 8×8 unitary operator S can be expanded in the form

$$S = \sum_{n=0}^{7} e^{i\sigma_n} |\psi_n\rangle \langle \psi_n|, \tag{27}$$

where $\{|\psi_n\rangle\}$ are the eigenstates and $\{e^{i\sigma_n}\}$ the eigenvalues of S. But

$$S = \prod_{n=0}^{7} S_{G^{-1}[|\psi_n\rangle]} X_{\sigma_n} S_{G[|\psi_n\rangle]}, \tag{28}$$

and each of the 24 terms in this product is the S-matrix of a gate in the repertoire. Hence S is also, and we have shown that $\mathbf{Q}$ is universal relative to the set of all 3-bit quantum gates (and hence also relative to the set of all 2- and 1-bit gates).

Now consider the quantum network $\mathbf{Q}_4$ that is shown in figure 11. In spite of the unit wire that loops backwards, $\mathbf{Q}_4$ is a gate, and a reversible one at that: Suppose that the four input bits and the looping bit are initially in the state $|a, b, 0, c, d\rangle$ (the third label describes the looping bit which, according to the rules laid down above, is initially prepared with the value '0'). Because of the various delaying unit wires, these five bits move from left to right in the network at equal speeds for six computational steps. As they do so, the first Toffoli gate, the $\mathbf{Q}$ gate, and the second Toffoli gate in turn cause them to undergo the evolution

$$\left.\begin{aligned} &|a,b,0,c,d\rangle \Rightarrow |a,b,ab,c,d\rangle \\ &\quad \Rightarrow [1+abc(\mathrm{i}\cos\tfrac{1}{2}\pi\alpha - 1)]\,|a,b,ab,c,d\rangle + abc\sin\tfrac{1}{2}\pi\alpha|a,b,ab,c,1-d\rangle \\ &\quad \Rightarrow [1+abc(\mathrm{i}\cos\tfrac{1}{2}\pi\alpha - 1)]\,|a,b,0,c,d\rangle + abc\sin\tfrac{1}{2}\pi\alpha|a,b,0,c,1-d\rangle. \end{aligned}\right\} \tag{29}$$

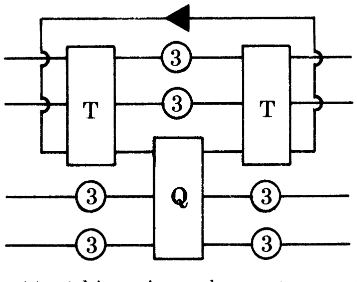

FIGURE 11. 4-bit universal quantum gate $\mathbf{Q}_4$.

After this evolution, the looping bit is again in a pure $|0\rangle$ state, ready to be reused one computational step later. The four output bits are also, jointly, in a pure state, albeit a linear superposition of computational basis states. Because this is true when the four input bits are in an arbitrary computational basis state, it must also be true when they are in a general state. Thus the $\mathbf{Q}_4$ network has an S-matrix

$$S^{abcd}_{\mathbf{Q}_4 a'b'c'd'} = \delta^a_{a'}\,\delta^b_{b'}\,\delta^c_{c'}[(1-abc)\,\delta^d_{d'} + \mathrm{i}abc\,\mathrm{e}^{-\frac{1}{2}\mathrm{i}\pi\alpha}(S^{\alpha}_{\mathrm{N}})^{\,d}_{\,d'}], \tag{30}$$

and always requires exactly five computational steps (not counting the input and output unit wires) to perform its computation. It follows that $\mathbf{Q}_4$ is a reversible gate.

Comparing (14) and (30) reveals a close similarity between $\mathbf{Q}_4$ and $\mathbf{Q}$. The 3-bit gate $\mathbf{Q}$ performs an operation something like this: 'if the first two bits are both "1", then perform the αth power of *not* on the third bit, and change the phase by $\frac{1}{2}\pi(1-\alpha)$; otherwise leave all three bits unchanged'. The 4-bit gate $\mathbf{Q}_4$ performs a 4-bit analogue of this: 'if the first three bits are all "1", then perform the αth power of *not* on the fourth bit, and change the phase by $\frac{1}{2}\pi(1-\alpha)$; otherwise leave all four bits unchanged'. The proof that $\mathbf{Q}$ is universal relative to the set of all 3-bit gates applies step by step, *mutatis mutandis*, to $\mathbf{Q}_4$ (for example equations (15)–(21) are unchanged except that there are fourteen '1's on the leading diagonals instead of six, etc.). This shows that $\mathbf{Q}_4$ is universal relative to the set of all 4-bit gates, and hence that $\mathbf{Q}$ is too.

Moreover, the construction of figure 11 can be extended indefinitely to construct n-bit analogues of $\mathbf{Q}$ for arbitrarily large n. Hence $\mathbf{Q}$ is universal relative to the set of all n-bit gates for every n, from which we deduce that it is universal relative to the set of all gates.

The universal quantum computer as a quantum computational network

So far, I have spoken of computing machines as performing one computation on a class of possible inputs. But it is often useful to regard part of the input of a computing machine as a *program* determining which of a class of computations is to be performed on the remainder of the input or on subsequent inputs. A machine that can perform a large and diverse class of computations depending on the program with which it is prepared is called a *general purpose computer*. Present-day general purpose computers are, as regards the computations they perform, approximations to the most general possible classical computer, the universal Turing machine $\mathscr{T}$ (Turing 1936). This is a theoretical model that is idealized in that (i) it has an unlimited memory, (ii) it can perform arbitrarily many computational steps with perfect reliability, and (iii) it is describable within the kinematics of classical physics.

In a previous paper (Deutsch 1985) I described a *universal quantum computer* $\mathscr{Q}$ that is idealized only in the first two of these three respects. In every other way it is a physically realizable machine, and machines resembling it arbitrarily closely can in principle be built. $\mathscr{Q}$ plays the same role in the quantum theory of computation as the universal Turing machine does in the classical theory. It can be programmed to perform with arbitrary reliability any computation that can be performed by any physically possible computing machine. $\mathscr{Q}$, like $\mathscr{T}$, consists of a processor with a finite-dimensional state space, and an unlimited memory tape. The memory tape consists of a sequence of bits, and in any computational basis state one and only one of the bits interacts with the processor during any given computational step. During the step the tape may move, or rather the carriers of the bits in the tape move, by exactly one bit in either direction, or (and here it differs of course from its classical counterpart) the tape may enter a linear superposition of states corresponding to having moved in one of those two ways.

The machine $\mathscr{Q}$ is 'universal' and the gate **Q** is 'universal'. I must now relate these two senses of the term 'universal', because *a priori* $\mathscr{Q}$ is universal in a stronger sense.

In both cases, for any computing machine $\mathscr{M}$ one can obtain machines that perform the same computation as $\mathscr{M}$ with arbitrarily high reliability; in one case by assembling **Q** gates into a network, and in the other by preparing part of the input of $\mathscr{Q}$ with a program. But the process of assembling a network and the process of programming a computer *are themselves computations*. When we say that a computation can be performed 'using' only **Q** gates or 'using' only $\mathscr{Q}$, we tacitly assume that these preparatory computations can also be performed. Now, the preparation of the input of the machine $\mathscr{Q}$ with a program is a task that could be performed by another machine such as $\mathscr{Q}$; thus a question-begging description is avoided and the theory of $\mathscr{Q}$ indeed provides a complete model of computation within quantum theory. But the task of assembling gates into a network could not be performed by another network. Therefore, the theory of quantum computational networks as described up to this point is incomplete as a model of computation. There are computations that the model relies upon but does not, even in principle, describe.

This deficiency can also be expressed in practical terms. Approximations to $\mathscr{Q}$ may be technologically difficult to build at present. But at least we know that when they are, we shall not have to solve any further problem in physics, engineering or the quantum theory of computation before being able to use them to perform general quantum computations. By contrast, even if a supply of working **Q** gates were readily available, it has not yet been shown that they would be useful to anyone who wanted to perform computations with them. For the gates would first have to be assembled into the network appropriate for the desired computation. And nothing I have said so far shows that performing this assembly is not tantamount to performing the desired computation.

This gap is bridged by showing that (i) the computation performed by any finite quantum computational network can be performed with arbitrary reliability by $\mathscr{Q}$, and (ii) $\mathscr{Q}$ can be approximated arbitrarily well by networks consisting entirely of **Q** gates, sources, and unit wires. Because a finite quantum network is a quantum system with a finite-dimensional state space, point (i) follows directly from the universality of $\mathscr{Q}$. Point (ii) is not immediately obvious. $\mathscr{Q}$ as described is certainly not a quantum network because the motion of the carriers of its bits (i.e. the motion of its tape) is affected by the computation, and indeed $\mathscr{Q}$ must be capable of being in coherent superpositions of carrier-position states. The processor of $\mathscr{Q}$

can straightforwardly be a quantum network. But replacing the whole of $\mathcal{Q}$ by a computationally equivalent quantum network involves representing the coherent motion of the carrier positions of $\mathcal{Q}$'s tape by network observables in a way that does not require quantum coherence in the motion of the network's carriers.

Because $\mathcal{Q}$ has an unlimited memory tape, there must be an observable of $\mathcal{Q}$, such as the 'tape position', that has an infinite spectrum, and any network simulating $\mathcal{Q}$ must have a corresponding observable. I am not concerned here with sterile questions about approximation and limits. Arbitrarily long tapes can certainly be constructed out of **Q** gates, and so can 'cursors' or other equivalent addressing mechanisms with any given finite range. The significant question raised by considering arbitrarily long tapes is whether in constructing a tape long enough for $\mathcal{Q}$ to perform a given computation, and in constructing the appropriate addressing mechanism, one has not in effect performed that computation oneself.

That question is settled by showing that in principle the task of assembling a tape of length n, with its addressing mechanism, can be accomplished by performing a certain subsidiary task n times. Thus the computational resources (principally the time and the number of gates) required to assemble an approximation to $\mathcal{Q}$ whose tape is of length n bits are bounded by linear functions of n, whereas of course the resources required by a general computation using a tape of length n bits are not bounded by any polynomial in n. I leave the details as an exercise for the reader, but one way (probably not the most efficient in practice) of ensuring that the task of assembling the tape satisfies this condition is to let each tape bit be accompanied by a complete copy of the processor, and by a 'cursor' bit. This is the quantum analogue of the standard embedding of Turing machines in one-dimensional cellular automata. The logic of the network may then be arranged so that each copy of the processor has an internal state consisting entirely of '0's, and so that it repeatedly computes the identity function unless its accompanying cursor bit is a '1', in which case it performs the computation that $\mathcal{Q}$'s processor would, on its own tape bit and internal state. It then sets its cursor bit to zero, places the two adjacent cursor bits in some state, possibly a superposition, in which exactly one of them is a '1', and swaps its own internal state with that of the adjacent processor whose cursor bit has just been set to '1'. Each of these operations is a reversible computation.

Dynamics of quantum computations

In my previous paper (Deutsch 1985) I described quantum computations entirely in terms of the relations between the states of the computing machine immediately before and immediately after each computational step. I did not enquire what dynamical processes actually relate the output of each computational step to its input, except to require (i) that they not be forbidden by the principles of quantum theory, or by other fundamental physical principles, and (ii) that they be, in a suitable sense 'local'. So far I have adopted the same approach in this paper. Requirement (i) was met in both cases by describing all the computational processes in terms of explicit unitary transformations. Requirement (ii) is built into the definition of a quantum computational network, because a bit interacts only with the finite number of other bits that are inside the same gate at the same time. It was met in the case of the universal quantum computer by ensuring that the processor, which has a finite-dimensional state space, interacts with only one tape bit at a time, all other bits remaining inert during that computational step.

However, as Benioff (1982) and others have pointed out, an S-matrix that is local in this sense may not be generated by an interaction hamiltonian that is local in the same sense. Indeed it cannot be so generated unless the hamiltonian has explicit time dependence. For example, if the motion during one computational step of duration T is described by the S-matrix S, then the time-independent hamiltonian that would generate this motion would be

$$\hat{H} = (\mathrm{i}/T) \ln \mathsf{S}. \tag{31}$$

If S has support only at matrix elements relating adjacent tape bits, then $\hat{H}$ will in general have matrix elements relating arbitrarily distant bits, because the expansion of the logarithm in (31) contains arbitrarily high powers of $\mathsf{S}-\mathsf{I}$. Similarly, if $\hat{H}$ is local, then S will not be.

Computing machines of the type suggested by Feynman (1985) have time-independent local hamiltonians, but do not complete their computations in a definite time. They go into superpositions of states in which different numbers of computational steps have been performed, and they have to be measured periodically to determine when they have finished. Thus if they were used as gates, additional components, with time-dependent hamiltonians, would be required.

We infer that any quantum computational network, and presumably any general-purpose computing machine that can actually be built, must have an explicitly time-dependent hamiltonian. This raises no problem of principle. Although it is true that we have excellent reasons for being reluctant to consider explicit time-dependence in the fundamental interactions of Nature, there is no reason why this reluctance should extend to finite machines, even those that are required to display quantum coherence.

Consider first the motion of the carriers. As I have stressed above, this is fixed and independent of the network's computational state. Moreover, the operation of the network does not rely on quantum interference between different states of the carriers, nor on quantum correlations between them, as it does for the bits proper; on the contrary, in an ideal network each carrier would have a sharply defined position at all times. But that means that each carrier would have a sharp trajectory, which raises the question whether such a thing is compatible with the quantum uncertainty principle.

If, for example, the carriers were particles constrained to move along frictionless wires but otherwise free, it would not be long before the quantum dispersion in their positions ruined the accuracy of the computations. However, there is no reason why the carriers should not be subjected to forces other than holonomic constraints. In particular there is no reason why ordinary methods of ensuring stability could not be applied to give the carriers arbitrarily sharp trajectories. For example, the carriers' arrival times at the inputs of each gate (or also at intermediate positions if necessary) could be measured, and appropriate delays introduced to put them back on their unvarying schedule. This would be an irreversible process, as is all error correction, but it would not affect the reversibility of the computation being performed.

That brings us to the bits proper. How would they need to interact? We need only consider the interaction of the three bits inside a single **Q** gate. Let $|\psi(t)\rangle$ be the state of these bits at a time t after they enter the gate. The hamiltonian operator that describes any such interaction is the 8×8 time-dependent hermitian matrix $\hat{H}(t)$ that appears in the Schrödinger equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = -\mathrm{i}\hat{H}(t)\,|\psi(t)\rangle. \tag{32}$$

If the duration of a computational step is T, then because $\hat{H}(t)$ presumably has support only during the period $0 < t < T$, one class of hamiltonians that satisfy (32) is

$$\hat{H}(t) = \frac{\mathrm{i}f(t)}{T}\ln S_{\mathrm{Q}} = \frac{-\pi f(t)}{2T}\begin{bmatrix} 0 &&&&&&& \\ & 0 &&&&&& \\ && 0 &&&&& \\ &&& 0 &&&& \\ &&&& 0 &&& \\ &&&&& 0 && \\ &&&&&& 1 & -\alpha \\ &&&&&& -\alpha & 1 \end{bmatrix}, \tag{33}$$

where f is any function such that

$$\frac{1}{T}\int_0^T f(t)\,\mathrm{d}t = 1. \tag{34}$$

and a physically attainable f would presumably go smoothly to zero when $t \to 0$ and when $t \to T$.

Such hamiltonians describe interactions that are both local and very simple.

How many universal gates are there?

Three-input, three-output reversible quantum gates are specified by 8×8 S-matrices. We have seen that (14) is the S-matrix of a universal gate, but it is certainly not the only such matrix. Because α was unspecified except for the condition that it be irrational, the gates whose S-matrices are the xth powers of that of **Q** describe universal gates when x is any non-zero rational number or almost any real number. Indeed, raising the S-matrix of any universal gate to almost any real power yields the S-matrix of another universal gate. If $\boldsymbol{P}$ and $\boldsymbol{Q}$ are bit permutations and $\boldsymbol{S}$ is the S-matrix of any universal gate, then $\boldsymbol{PSQ}$ is obviously the S-matrix of another universal gate. Furthermore, if $\boldsymbol{U}$ is any 2×2 unitary matrix, then

$$(\boldsymbol{U}^{\dagger} \otimes \boldsymbol{U}^{\dagger} \otimes \boldsymbol{U}^{\dagger})\boldsymbol{S}(\boldsymbol{U} \otimes \boldsymbol{U} \otimes \boldsymbol{U}) \tag{35}$$

describes a universal gate too.

The question arises, how many universal gates are there? Clearly not all 8×8

unitary matrices describe universal gates. The unit matrix, for instance, is one that does not. But I conjecture that for all integers $n \geqslant 3$ almost all $2^n \times 2^n$ unitary matrices describe n-bit universal gates.

Conclusions

The existence of a universal quantum gate simplifies the theory of quantum computational networks and of quantum computation generally. At least as far as the relation between output and input is concerned, it is sufficient to study only those networks that consist entirely of **Q** gates, unit wires, sources and sinks. The simplicity of the theory also bodes well for the possibility of actually constructing quantum computers in the future. They are likely to be realized as quantum computational networks.

I thank Dr Keith Hannabuss for improvements to the proof of the universality of the **Q** gate, and Rolf Landauer for drawing my attention to the importance of providing a simple hamiltonian description of computing machines.

References

Benioff, P. A. 1982 *Int. J. theor. Phys.* **21**, 177.
Bennett, C. H. 1973 *IBM Jl Res. Dev.* **6**, 525–532.
Deutsch, D. 1985 *Proc. R. Soc. Lond.* A **400**, 97–117.
Feynman, R. P. 1985 *Optics News* **11**, 11–20.
Toffoli, T. 1981 *Math. Systems Theory* **14**, 13–23.
Turing, A. M. 1936 *Proc. Lond. math. Soc.* ser. 2 **43**, 544.