Journal of the Nigerian Association of Mathematical Physics
***Volume 10 (November 2006),* 433 - 446**
© J of NAMP

Quantum computer gate simulations

Adetunmise C. Dada
Department of Physics, Obafemi Awolowo University,
Ile-Ife, Nigeria
email: techada@yahoo.com

Abstract

A new interactive simulator for Quantum Computation has been developed for simulation of the universal set of quantum gates and for construction of new gates of up to 3 qubits. The simulator also automatically generates an equivalent quantum circuit for *any* arbitrary unitary transformation on a qubit. Available quantum computer simulators attempt to emulate the various physical realisations of quantum computation, simulate existing quantum algorithms or are aimed at facilitating the development of new algorithms. However, because of the level of advancement and complexity of quantum computation algorithms, these simulators tend to be quite complex, at least from a novice's point of view. As a result of this, beginners are often at a loss when trying to interact with them. The simulator here proposed therefore is aimed at bridging the gap somewhat, making quantum computer simulation more accessible to novices in the field.

1.0 Introduction

The field of quantum computation, which deals with the harnessing of physical phenomena unique to quantum mechanical systems to realise a fundamentally new mode of information processing, has come quite a long way; why, its basic ideas were formulated well over two decades ago! [1, 2]. However, progress in the software/algorithm related aspect of quantum computation has been relatively slow. This can be deduced from the fact that, from its incipience, only a small number of quantum computer algorithms have been discovered. The simple explanation for this is: "coming up with good quantum algorithms seems to be *hard*" [3]. Well, what more do we expect when Richard Feynman, a Nobel Laureate in quantum mechanics—the very basis of quantum computation—said, "I think I can safely say that nobody understands quantum mechanics". It is little wonder then that developing computational algorithms which are based on such a theory is difficult.
However, we seek to go beyond this 'simple explanation' because as scientists, inquisitiveness is our defining characteristic, as asking questions and finding answers is the very essence of science. Therefore, the question which inspires the research work presented in this paper is: *Why is coming up with quantum algorithms hard?* This question was addressed by Michael Nielsen and Isaac Chuang [3]. According to them, algorithm design for quantum computers is hard because designers face two difficult problems not faced in the construction of algorithms for classical computers. First, our human intuition is rooted in the classical world and naturally, we come up with classical algorithms when that intuition is used as an aid to the construction of algorithms. Second, it is not enough to design an algorithm which is merely quantum mechanical. To be truly interesting, the algorithm must out-perform any existing classical algorithm!
Here, the following are added to the line of questions on the progress of quantum computation:
(1) *Are there other reasons why it seems to be hard to come up with quantum algorithms?*
(2) *While it is true that our intuition is rooted in the classical world; what can be done practically, to found and substantiate our intuitive powers in the quantum domain?*

To question (1) above, the answer is “yes”. Another possible reason which is hereby identified is that the number of scientists involved in active research on the algorithm related aspect of quantum computation is relatively few. There is no doubt that the chances of getting major breakthroughs in quantum computation will increase if more individuals are involved. However, many researchers and scientists hesitate to get involved in quantum computation for a number of reasons. Some are reluctant because of sheer pessimism; they feel that the field will develop no further. Others are taken aback because they feel quantum computation is just not for them to peer into. Even though they get curious about the possibilities that it offers, the formalism in which it is presented is not palatable for them and thus, they lose the courage to delve deeper.

To address question (2) we ask another question: Why is our intuition rooted in the classical world? The answer is obvious. It is because in the world around us and in our everyday activities, things behave according to the laws of classical physics. This means that if everyday objects behaved quantum mechanically, we likely would more easily be able to intuit in that way. For that reason, creating ‘objects’ which behave according to the laws of quantum mechanics and making such available for everyday interaction would no doubt help more individuals develop their intuitive powers in the direction of quantum computation. This *can* be done through quantum computer simulation. The new simulator presented herein is aimed at doing just that.

Since quantum computer hardware systems which have been constructed are insufficient for detailed exploration of some of the algorithms that have been proposed, and even are currently not available outside research laboratories; quantum computer simulators have been useful for exploring quantum algorithms thereby enabling investigations on such to take place, which would otherwise be impossible given the current state of quantum computer hardware. In fact, a lot of excellent simulators have been designed to simulate various aspects of quantum computation, to verify the feasibility of quantum computers, to investigate the effect of errors on quantum computation, etc. Brilliant reviews have been written on these as well [4, 5, 6].
However, because these simulators tend to concentrate on emulating various physical realizations of quantum computation, simulating the available quantum algorithms, demonstrating new theories of quantum computation, there are hardly any simulators which lend themselves to the easy understanding of beginners in the field. In fact, there is no Windows® based simulator available which concentrates on simulating the behavior of a single qubit as a quantum mechanical system, thereby enhancing appreciation for what a quantum computer actually does at the most fundamental level, in terms of unitary gate operations etc, and which, for instance, enables a user to start computation on a qubit which is in *any* arbitrary initial state as may be specified by the user (all known simulators[i] [4, 5, 6] only allow a user to start computations on basis states). Many simulators are used to perform quantum gate operations on qubits; however, none automatically generates a single quantum circuit, composed of only two types of gates, which performs any specified transformation of the qubit (that is with only the initial and final states specified). Moreover, there is a need for simulators aimed at helping fledglings in quantum computation gain some familiarity, and aiding their understanding of the workings of more advanced simulators, thereby instilling in them a measure of confidence which will encourage them to delve deeper into the field. The simulator here proposed is targeted at filling these needs. It presents itself as a practical tool for addressing question (2) raised above, serving as a contribution in its own little way to the field.
The following gives a summary of basic principles of quantum computation on which the machinery of the new simulator is based.

2.0 Basic ideas in quantum computation

A quantum bit (***qubit***[ii]) is the basic unit of information in quantum computation. The qubit can exist in states 0 or 1 like its classical counterpart (i.e. the bit), but is different in the sense that it can also exist in a coherent superposition state $\alpha|0\rangle + \beta|1\rangle$ in which it can have both values at the same time, with $|\alpha|^2$ and $|\beta|^2$ representing the probability that it will be ‘found’ in each state, α and β being complex numbers. This strange property of the qubit arises as a direct consequence of its strict adherence

[1]This refers to software that was accessible via the web at the time of writing
[2]The term was coined by Schumacher [8]

to the laws of quantum mechanics which differ profoundly from the laws of classical physics. In fact, quantum computation is based on the following three postulates of quantum mechanics: (1) Superposition (2) Coherence, and (3) Entanglement.

2.1 ***Superposition*** expresses the ability of a quantum mechanical system to be in two classical states at the same time as mentioned above. However, there have been generalisations in which the basic unit of computation would then be a quantum system which could have more than two basis states as done in the Quantum Qudit Theory [7]. However, to keep things simple, this discussion and hence the simulator here proposed will be based on *qubits*, that is computational units which can be found in only two classical states. The qubit can be in a linear superposition of the states $|0\rangle$ and $|1\rangle$ —it is in the state $|0\rangle$ with probability amplitude $\alpha \in \mathbb{C}$ and in the state $|1\rangle$ with probability amplitude $\beta \in \mathbb{C}$. It is as though the qubit "does not make up its mind" as to which of the 2 classical states it is in. Such a '2-state' quantum system is said to be in a *superposition* of the two classical states, and its state can be written as a unit (column) vector $\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \in \mathbb{C}^2$.

In Dirac notation, this may be written as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \alpha,\beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \qquad (2,1)$$

The Dirac notation has the advantage that it labels the basis vectors explicitly. This is very convenient because the notation expresses both that the state of the qubit is a vector, and that it is data (0 or 1) to be processed. (The $\{|0\rangle, |1\rangle\}$ basis is called the *standard* or *computational basis*.)

This linear superposition $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is part of the 'private world' of the qubit. For us to know its state, we must make a measurement. Measuring $|\psi\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis yields $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with the probability $|\beta|^2$.

One important aspect of the measurement is that it alters the state of the qubit: the effect of the measurement is that the new state is exactly the outcome of the measurement, i.e. if the outcome of the measurement of $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ yields $|0\rangle$, then following the measurement, the qubit is in state $|0\rangle$. This means that one cannot collect any additional information about α and β by repeating the measurement.

2.2 ***Coherence*** refers to the ability of a system to remain quantum rather than classical over a period of time. *Decoherence* occurs when the quantum state degenerates into a classical state upon being measured. Here, 'measurement' refers to events both intentional and accidental, in which information is being transferred out the quantum system. In the absence of such events, the system maintains its quantum coherence.

2.3 ***Entanglement*** effects come to light in the case of systems of more than one qubit. It is a strange phenomenon in which, if the quantum state of one of the qubits changes, then the state(s) of the other(s) change(s) in a particular way, even if there is no interaction between them. The kinds of such correlations are much more than one would expect classically.

Consider the case of two qubits. Classically speaking, each qubit can be either in the 0 or 1 state. Therefore, the two qubits are in one of the four states –00, 01, 10, 11– and this represents two bits of classical information. Quantum mechanically, they are in a superposition of those four states:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \qquad (2.2)$$

where $\sum_{ij} |\alpha_{ij}|^2 = 1$.

Again, this is just Dirac notation for the unit vector in

$$\mathbb{C}^4 : \begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix}$$

where $\alpha_{ij} \in \mathbb{C}$, $\sum_{ij} |\alpha_{ij}|^2 = 1$.

Now consider the state of a two qubit system given by $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. This state[iii] is a very popular example of an entangled state. Notice that the state cannot be represented as $(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle)$[iv] for any complex numbers $\alpha_0, \alpha_1, \beta_0$ or β_1. We cannot analyze the state of each individual qubit in this system, because the states of the two qubits are entangled. If we take a measurement on the first qubit, then the state of the other qubit is determined by the outcome of the measurement. With probability ½ we see $|0\rangle$ as the outcome of the measurement, and in this case, we know that the state of the system must be $|00\rangle$.

Quantum computers depend on the principles expressed above. That is, they exploit the ability of a quantum particle to be in two states at the same time, such as the spin of a nucleus pointing simultaneously up and down relative to an applied magnetic field. With the two states representing a 'one' and a 'zero', N such particles, each representing a qubit, can then be combined or entangled so that they represent 2^N values simultaneously. A quantum computer would then, in principle, be able to process these values at the same time, making it exponentially faster than a classical computer.

3.0 Simple Operations on Qubits

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

As expounded by Ekert *et al*. [9], if we assume that information is stored in the register in binary form, the number 6 for instance will be represented by a register in the state $|1\rangle \otimes |1\rangle \otimes |0\rangle$. In more compact notation, $|a\rangle$ stands for the tensor product $|a_{n-1}\rangle \otimes |a_{n-2}\rangle \ldots |a_1\rangle \otimes |a_0\rangle$, where $|a_i\rangle \in \{0,1\}$, and it represents a quantum register prepared with the value $a = 2^0 a_0 + 2^1 a_1 + \ldots 2^{n-1} a_{n-1}$. There are 2^n states of this kind, representing all binary strings of length n or numbers from 0 to $2^n - 1$ and they form a convenient computational basis. In the following $|a\rangle \in \{0,1\}^n$ (a is a binary string of length n) implies that $|a\rangle$ belongs to the computational basis.

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

$$|0\rangle \otimes |1\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix} \equiv |011\rangle \equiv |3\rangle, \qquad (3.1)$$

Similarly, $$|1\rangle \otimes |1\rangle \otimes |1\rangle \equiv |111\rangle \equiv |7\rangle, \qquad (3.2)$$

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

$$\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)\otimes|1\rangle\otimes|1\rangle\equiv\frac{1}{\sqrt{2}}\left(|011\rangle+|111\rangle\right) \tag{3.3}$$

$$\equiv\frac{1}{\sqrt{2}}\left(|3\rangle+|7\rangle\right) \tag{3.4}$$

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

$$\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right), \tag{3.5}$$

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

$$|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle \tag{3.6}$$

or in decimal notation as $|0\rangle+|1\rangle+|2\rangle+|3\rangle+|4\rangle+|5\rangle+|6\rangle+|7\rangle$, (3.7)

or simply as $\sum_{x=0}^{7}|x\rangle$.

4.0 Quantum gates and networks

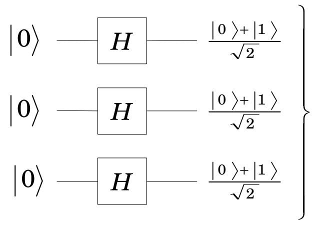
These preparations, and any other manipulations on qubits, have to be performed by unitary operations. A ***quantum logic gate***$^{\mathrm{v}}$ is a device which performs a fixed unitary operation on selected qubits in a fixed period of time and a ***quantum network*** is a device consisting of quantum logic gates whose computational steps are synchronised in time [13]. The outputs of some of the gates are connected to the inputs of others. The *size* of the network is the number of gates it contains.

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

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

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

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



$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
$|0\rangle$
H
$\frac{|0\rangle+|1\rangle}{\sqrt{2}}$

IN BINARY

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

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

IN DECIMAL

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

$$|101\rangle\mapsto\frac{1}{2^{2/3}}\begin{Bmatrix}|000\rangle-|001\rangle+|010\rangle-|011\rangle+\\ -|100\rangle+|101\rangle-|110\rangle+|111\rangle\end{Bmatrix} \tag{4.1}$$

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

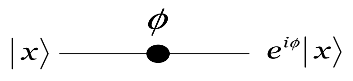
$$|y\rangle\mapsto 2^{-n/2}\sum_{x\in\{0,1\}^n}(-1)^{y\cdot x}|x\rangle \tag{4.2}$$

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

$$y\cdot x=(y_{n-1}x_{n-1}+\dots y_1x_1+y_0x_0) \tag{4.3}$$

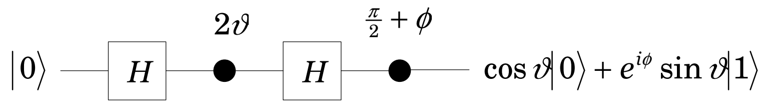
We will need another single qubit gate – the ***phase shift gate*** ϕ defined as $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto e^{i\phi}|1\rangle$, or, in matrix notation,

$$\phi = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$



(4.4)

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



(4.5)

The above is obvious, *in fact*:
Performing the Hadamard transform on the basis ket $|0\rangle$, we have

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

A subsequent of phase shift of 2ϑ gives

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \mapsto \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\vartheta i}|1\rangle\right).$$

Another Hadamard transform now yields

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\vartheta i}|1\rangle\right) \mapsto \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{e^{2\vartheta i}}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\right) = \frac{1}{2}\left(\left(1+e^{2\vartheta i}\right)|0\rangle + \left(1-e^{2\vartheta i}\right)|1\rangle\right).$$

Finally, phase-shifting this resultant state by $\dfrac{\pi}{2}+\phi$ gives

$$\frac{1}{2}\left(\left(1+e^{2\vartheta i}\right)|0\rangle + \left(1-e^{2\vartheta i}\right)|1\rangle\right) \mapsto \frac{1}{2}\left(\left(1+e^{2\vartheta i}\right)|0\rangle + e^{\left(\frac{\pi}{2}+\phi\right)i}\left(1-e^{2\vartheta i}\right)|1\rangle\right) = \frac{1}{2}e^{i\vartheta}\left(\left(e^{-i\vartheta}+e^{i\vartheta}\right)|0\rangle + e^{\left(\frac{\pi}{2}+\phi\right)i}\left(e^{-i\vartheta}-e^{i\vartheta}\right)|1\rangle\right)$$

$$= e^{i\vartheta}\left(\frac{1}{2}\left(e^{-i\vartheta}+e^{i\vartheta}\right)|0\rangle + \frac{i}{2}e^{\phi i}\left(e^{-i\vartheta}-e^{i\vartheta}\right)|1\rangle\right) = e^{i\vartheta}\left(\frac{1}{2}\left(e^{-i\vartheta}+e^{i\vartheta}\right)|0\rangle + \frac{-i}{2}e^{\phi i}\left(e^{i\vartheta}-e^{-i\vartheta}\right)|1\rangle\right)$$

$$= e^{i\vartheta}\left(\frac{1}{2}\left(e^{-i\vartheta}+e^{i\vartheta}\right)|0\rangle + \frac{1}{2i}e^{\phi i}\left(e^{i\vartheta}-e^{-i\vartheta}\right)|1\rangle\right) = e^{i\vartheta}\left(\cos\vartheta|0\rangle + e^{\phi i}\sin\vartheta|1\rangle\right)$$

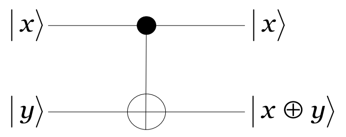
We then neglect the phase factor $e^{i\vartheta}$ to obtain Equation (4.5).
Consequently, the Hadamard gates and the phase gates are sufficient to construct any unitary operation on a single cubit.

Thus they can be used to transform the input state $|0\rangle|0\rangle\ldots|0\rangle$ of the n cubit register into any state of the type $|\psi_1\rangle|\psi_2\rangle\ldots|\psi_n\rangle$, where $|\psi_i\rangle$ is an arbitrary superposition of $|0\rangle$ and $|1\rangle$. These are rather special *n*-qubit states, called the product states or the separable states. In general a quantum register of size $n>1$ can be prepared into states which are not separable i.e. entangled states. For example, for two qubits ($n = 2$), the state

$$\alpha|00\rangle + \beta|10\rangle = |0\rangle \otimes \left(\alpha|0\rangle + \beta|1\rangle\right) \qquad (4.6)$$

is separable into, $|\psi_1\rangle = |0\rangle$ and $|\psi_2\rangle = \alpha|0\rangle + \beta|1\rangle$, whilst the state
$\alpha|00\rangle + \beta|11\rangle$ $(\alpha, \beta \neq 0)$ cannot be separated because it cannot be written as a tensor product.
In order to entangle two (or more) qubits we have to extend our repertoire of quantum gates to two-qubit gates. The most popular two-qubit gate is the ***controlled-NOT*** gate (c-NOT), also known as the XOR or the measurement gate. It flips the second (target) qubit if the first (control) qubit is $|1\rangle$ and does nothing if the control qubit is $|0\rangle$. The gate is represented by the unitary matrix

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



(4.7)

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

$$|x\rangle|0\rangle \mapsto |x\rangle|x\rangle \qquad x = 0,1 \tag{4.8}$$

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

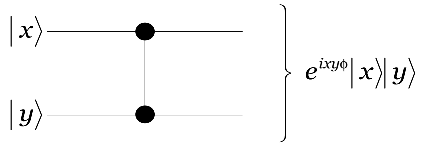
$$|\Psi\rangle|0\rangle \mapsto |\Psi\rangle|\Psi\rangle \tag{4.9}$$

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

$$(\alpha|0\rangle + \beta|1\rangle)|0\rangle \mapsto \alpha|00\rangle + \beta|11\rangle\ ) \tag{4.10}$$

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

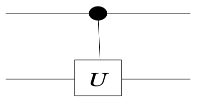
$$B(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$$



(4.11)

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

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



The Hadamard gate, all phase gates, the C-NOT, form an infinite universal set of gates i.e. if the C-NOT gate as well as the Hadamard and all phase gates are available then any n -qubit unitary operation can be simulated exactly with $O(4^n n)$ such gates [10]. (Here the asymptotic notation - $O(T(n))$ is used, which means bounded above by $cT(n)$ for some constant *c* for sufficiently large *n*.) This is not the only universal set of gates. In fact, almost any gate which can entangle two qubits can be used as a universal gate [11, 12]. Mathematically, an elegant choice is a pair of the Hadamard and the controlled-*V* (*c* - *V*) where *V* is described by the unitary matrix

$$V = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{4.12}$$

The two gates form a finite universal set of gates –networks. Networks containing only a finite number of these gates can approximate any unitary transformation on two (and more) qubits.

A *quantum computer* will be viewed here as a quantum network (or a family of quantum networks) and *quantum computation* is defined as a unitary evolution of the network which takes its initial "input" into some final state "output".

5.0 Quantum Computer Simulation

As Feynman said [2], the only way to effectively model a quantum mechanical system is by using another quantum mechanical system. Therefore, a quantum computer simulator is merely an attempt to model a quantum mechanical system[vi] on a classical system and the simulator must keep track of exponentially many computations in order to do this accurately. If it were possible to construct an efficient quantum computer simulator, then it would no longer be necessary to construct a quantum computer via production of quantum computer hardware (except perhaps for Moore's law). The simulator itself would effectively be a quantum computer.

Therefore quantum computer simulation remains only an attempt to model the operations of a quantum computer. In fact, every simulator has this as its primary objective. However, quite a number of quantum computer simulators have been developed for secondary aims, as reviews on simulators show [3, 4, & 5].

6.0 Q~*GATE* Simulator

Q-GATE, the quantum computer simulator presented in this paper was designed with the following objectives:

1. To simulate the most fundamental quantum logic gate operations, applying them to *any* input qubit state.
(2) To carry out its operations while preserving the quantum mechanical notations as well as the displaying the gate operations in matrix form.
(3) To emphasise the matrix manipulation carried out in the various gate operations.
(4) To automate the creation of quantum circuits.
(5) To provide a software tool that may serve as a teaching aid for the basic concepts of quantum logic.
(6) To provide simulator that mimics all the basic properties of a quantum computer.
(7) Ultimately, to increase the number of individuals involved in quantum computation.

6.1 Brief Description

Q~*GATE* has a very user-friendly GUI (graphical user interface). It is lavished with *tool tip texts*[vii] which tell the user what to do with each graphical component and it also uses different forms of user interactivity: from drag-and-drop to clicking, scrolling, e. t. c.
It makes use of the following matrix representations of the computational basis vectors

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \qquad (6.1)$$

So that

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad (6.2$$

and so on.
Moreover, Q~*GATE* presents the input and output qubits both in Dirac notation and matrix form. It simulates gate operations on quantum registers of up to size 3. It adopts the matrix formulation of quantum computation, in which the gates are actually unitary matrices, and the gate operations are essentially matrix multiplication operations. Hence, if we apply the Hadamard gate (for instance) to the state of a qubit, which itself is a column matrix, we are in effect, carrying out the following:

For $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ in Dirac notation, or $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ in matrix form,

$H|\psi_1\rangle = |\psi_2\rangle$, i.e.

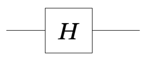
$$H|\psi_1\rangle \equiv \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = |\psi_2\rangle \qquad (6.3)$$

6.2 Gate Simulations

1-qubit, 2-qubit, and 3-qubit gate operations are simulated and these simulations are very useful in the understanding and design of quantum circuits because they contain a universal set of gates [10, 11, 12]. The rows and columns of the unitary matrices are labelled from left to right and top to bottom as $00\ldots0$, $00\ldots1$ to $11\ldots1$ with the bottom-most wire being the least significant bit.
The circuit symbols as well as matrix representations of the gates simulated are shown below.

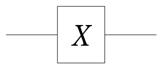
<u>1-qubit gates</u>

The Hadamard gate



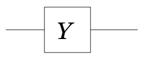
$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

The Pauli-X gate



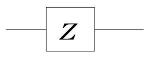
$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

The Pauli-Y gate



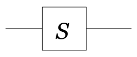
$$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$$

The Pauli-Z gate



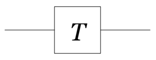
$$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

The Phase gate (*S* or *V* gate)



$$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$$

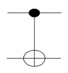
The $\frac{\pi}{8}$ gate



$$\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$$

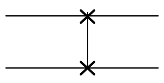
<u>2-qubit gates</u>

The controlled-NOT gate



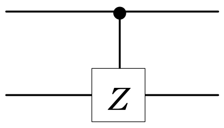
$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

The swap gate



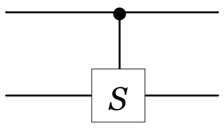
$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

The controlled-Z gate



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

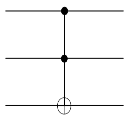
The controlled-phase gate



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$$

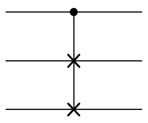
<u>3-qubit gates</u>

Toffoli gate (controlled-controlled-NOT)



$$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

The Fredkin gate (controlled-swap gate)



$$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

Note that $e^{i\pi/4}$ is the square root of i, so that the $\frac{\pi}{8}$ gate is the square root of the phase gate, which itself is the square root of the Pauli-Z gate.

Another interesting feature of Q~*GATE* is that it allows the user to create a custom (user-defined) gate. This can be done either by applying a succession of the predefined unitary gates or by manually entering the complex number components of the gate operator matrix, in which case the user has to test and confirm the unitarity of the specified gate, upon which the gate is then accepted.

Q~*GATE* also captures what happens when a qubit is measured. By modelling the effect of measurement so well, it enhances the understanding of the meaning of the probability amplitudes and the quantum states.

6.3 Gate construction example –An interactive session

Let us assume we want to simulate the application of a 2-qubit unitary operation defined by

$$\hat{U} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \qquad (6.4)$$

to a quantum register of size 2 (this is a special case of the Quantum Fourier Transform [3]).
The first thing to do is to run the executable file Q~*GATE*.exe from the programs menu. (That should be after Q~*GATE* has been installed of course!). This loads the main program window which has tabs for toggling between one, two, three, and higher qubit operations. For the purpose of the problem at hand, we click on the 2 qubit tab. The tab page for two qubit operations comes up, and we then click on the "User defined gate" command button in the frame with the caption "Gate". A window then pops up, which allows us to specify values for each matrix element for the gate. After entering the elements of the gate according to equation (25), and running the unitarity test, we are then ready to use this gate on the input register. If, for instance, we specify the state of the input register to be $|00\rangle$, which is

$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ in matrix form, then Q~GATE basically performs the following.

$$\hat{U}|\psi\rangle=\frac{1}{2}\begin{pmatrix}1&1&1&1\\1&i&-1&-i\\1&-1&1&-1\\1&-i&-1&i\end{pmatrix}\times\begin{pmatrix}1\\0\\0\\0\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\equiv\frac{1}{2}\left(|00\rangle+|01\rangle+|10\rangle+|11\rangle\right).$$

The simulator is thus very simple to use. In fact, its elegant simplicity and presentation of the mathematical as well quantum mechanical notations constitute a part of the features which make Q~*GATE* stand out as a simulator.

6.4 Circuit Generation

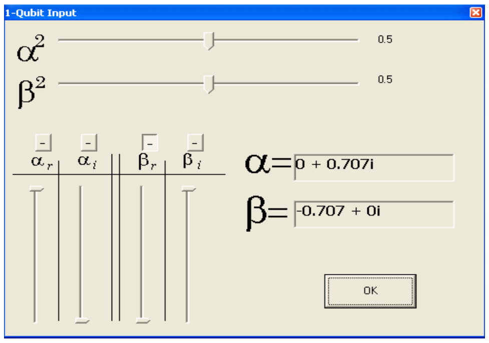
Q~*GATE* automates the generation of circuits which carry out a specified transformation on a qubit. Its automatic generation of an equivalent circuit is based on the discussions in section 4.0 above, and it particularly makes use of equation (14) i.e.

$$|\psi_1\rangle\mapsto|\psi_2\rangle=e^{i\vartheta}\left(\cos\vartheta|0\rangle+e^{i\phi}\sin\vartheta|1\rangle\right) \qquad (6.5)$$

Recall that this represents a most general state of the qubit. Therefore, to specify any desired rotation[viii] or transformation of the qubit, we only need to specify θ and ϕ in the equation.

Q~*GATE* allows the user to specify an arbitrary final state of a qubit, not only by means of a sequence of gate operations, but also by another interactive and perhaps more user friendly means (especially from a novice's point of view) –sliders. There is no known simulator which makes use of this. The sliders are used to specify the values of $|\alpha|^2$ and $|\beta|^2$ and subsequently the complex number values of α, β; and hence the state of the qubit $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$.

An instance of this is shown below.



$\alpha=$ 0 + 0.707i
$\beta=$ -0.707 + 0i

Figure 1: Specifying a qubit state in Q~GATE by means of sliders.

The sliders are coded in such a way that the restriction imposed by the normalization condition $|\alpha|^2 + |\beta|^2 = 1$ is enforced. That way, the user can select a truly 'arbitrary' state.

More to that, a user may choose to generate a quantum circuit which rotates a qubit in the computational basis state $|0\rangle$ to the state $\alpha|0\rangle + \beta|1\rangle$ specified by the user. This makes Q~*GATE* a useful tool for grasping the fundamental concepts of building a quantum circuit which performs a particular transformation on a qubit.

In obtaining the values of θ and ϕ, Q~*GATE* employs the well known Newton's Method. The values are used to construct the phase-shift gates in equation (4.5). Q~*GATE* then couples these phase shift gates with the Hadamard gate as shown in the diagram. The user can of course, also use Q~*GATE* to verify that this network indeed performs that transformation.

The method employed by Q~*GATE* in generating an equivalent circuit to perform a specified transformation on a single qubit is the following:

We can recast equation (6.5) as, $|\psi\rangle = e^{i\vartheta}\left(\cos\vartheta|0\rangle + e^{\phi i}\sin\vartheta|1\rangle\right)$. (6.6)

To build the circuit in equation (4.5) it is sufficient to obtain θ and ϕ. Since the specified state $|\psi\rangle$ will be in the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with $\alpha = \alpha_r + \alpha_i i$, and $\beta = \beta_r + \beta_i i$, (6.7)

expanding equation (6.6) and equating the real and complex parts of the coefficients of the basis states yields

$$\begin{aligned}\alpha_r &= \cos^2\theta\\ \alpha_i &= \sin\theta\cos\theta\\ \beta_r &= \sin\theta\cos\theta\cos\phi - \sin^2\theta\sin\phi\\ \beta_i &= \sin^2\theta\cos\theta + \sin\theta\cos\theta\sin\phi\end{aligned} \qquad (6.8$$

In equation (6.8) we obtain θ easily from the value of α_r, $\theta = \cos^{-1}\left(\pm\sqrt{\alpha_r}\right)$ (6.9)

From equation (3.1), we obtain two values of θ, one corresponding to $+\sqrt{\alpha_r}$, and the other to $-\sqrt{\alpha_r}$. We can find which of these values to select by trying them out in equation (29b). Thus we obtain θ. Next, we use the value of θ in equation (6.6) to obtain ϕ given the value of β_r.

However, because of the implicit nature of the resulting function of ϕ i.e. $g(\phi) = \beta_r - \sin\theta\cos\theta\cos\phi + \sin^2\theta\sin\phi$, Newton's method is used to obtain ϕ to a fairly good degree of accuracy. Recall that according to Newton's method, if we make a first approximation $\phi = \phi_0$, however bad, to the root of $g(\phi)$, then subsequent ones $\phi = \phi_1, \phi_2, \phi_3, \phi_4$ …, such that

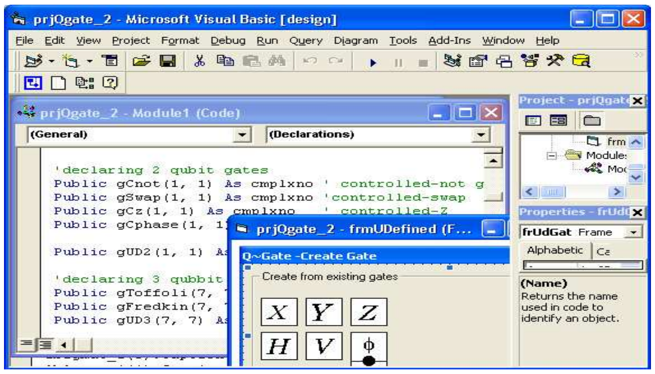
$$\phi_{n+1} = \phi_n - \frac{g(\phi_n)}{g'(\phi_n)}; \qquad (6.10)$$

after a number of iterations, we can obtain a fairly accurate ϕ, since each approximation is always better than the preceding one. Q~GATE iterates this procedure programmatically (with $\phi_0 = 0$ radians) to give us the value of ϕ. The values of θ and ϕ are then used to construct the quantum circuit as in equations (4.5) and (6.5)

7.0 Simulator Design

The simulator was designed with the Visual Basic 6.0 programming language. One advantage of using visual basic is that Any PC running an MS Windows OS and a VB500.DLL can run VB applications. This is 75 % of all the world's PCs. I therefore found VB very suitable, especially with the aim of the Q~*GATE* simulator in mind –reaching out to beginners in quantum computation. Links to the program files and source codes can be accessed through http://tunmise.4t.com. he screenshots of the simulator during design and run-time are shown below (Figures 3, 4).

Q~*GATE* takes an input of qubits specified by the user and allows the user to perform a simulation by clicking on a gate. The result of the operation is seen immediately. There is also a status/log window which keeps record of all operations carried out in a simulation session, allowing the user to save this for future reference.


prjQgate_2 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
prjQgate_2 - Module1 (Code)
(General)
(Declarations)
Project - prjQgate
frm
Modules
Moc
Properties - frUd
frUdGat Frame
Alphabetic
Ca
prjQgate_2 - frmUDefined (F...
Q~Gate -Create Gate
Create from existing gates
X
Y
Z
H
V
ϕ

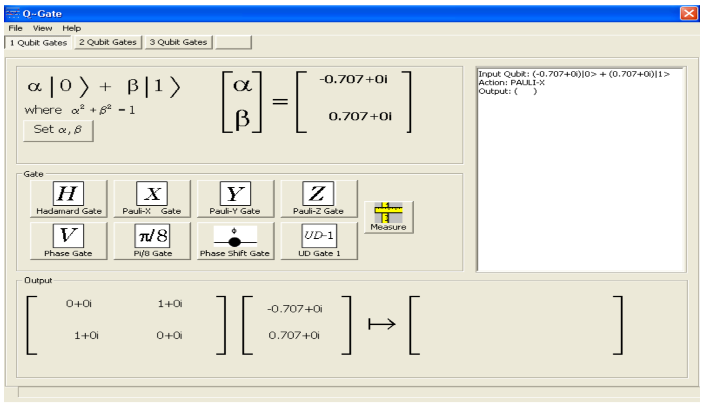
```
'declaring 2 qubit gates
Public gCnot(1, 1) As cmplxno ' controlled-not g
Public gSwap(1, 1) As cmplxno 'controlled-swap
Public gCz(1, 1) As cmplxno   ' controlled-Z
Public gCphase(1, 1

Public gUD2(1, 1) As

'declaring 3 qubbit
Public gToffoli(7, 7
Public gFredkin(7, 7
Public gUD3(7, 7) As
```

(Name)
Returns the name used in code to identify an object.

Figure 3:Q-GATE simulator (Visual Basic design)



```
Input Qubit: (-0.707+0i)|0> + (0.707+0i)|1>
Action: PAULI-X
Output: (    )
```

Figure 4: Q-GATE simulator (Runtime)

8.0 Limitations of Q-GATE

In addition to the general limitations of quantum computer simulators [18], Q~*GATE*

- ♦ does not include simulation of errors,
- ♦ cannot be used for carrying out simulations on a large number of qubits.

These however help to keep Q~*GATE* simple, so as not to defeat its purpose.

9.0 Conclusions

Q~GATE gives a useful addition to the available quantum computer simulators. It increases our appreciation for quantum logic and basic quantum computational operations in a unique way, providing a teaching aid for introductory aspects of quantum computation. The method employed in the generation of circuits which rotate qubits from a basis state to any specified arbitrary state, can be generalised and may find application in the modelling of quantum mechanical systems.

10.0 Acknowledgement

I gratefully acknowledge the enlightenments received from Dr. O. Odundun. I thank Dr. J. Idiodi for motivating and encouraging discussions. I would also like to thank Hans De Raedt for sending me the updated article of ref. [6].

Notations

The Bell state

⊗ Denotes the outer tensor product

Of course quantum logic gates are different from their classical counterparts because they can *create* and *perform operations on* superpositions

which in this case, is the quantum computer
small pop-up balloons
A qubit is seen as a vector in Hilbert space

References

[1] Benioff. The Computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as presented by Turing machines. *J. Stat. Phys. 22(5), 563-591, 1980.*

[2] R. P. Feynman. Simulating physics with computers. *Int. J. Theor. Phys., 21:467, 1982*

[3] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, 2000.

[4] J. Wallace. *A collection Quantum Computer Simulators,* www.dcs.ex.ac.uk/~jwallace/simtable.htm, 2000

[5] D. Fitzgerald. *Known quantum simulators and their attributes.* www.compsoc.nuigalway.ie/~damo624, 2004

[6] H. De Raedt and K. Michielsen. Department of Applied Physics, Materials Science Centre, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands. *Computational Methods for Simulating Quantum Computers*, 2005

[7] D. Fitzgerald. National University of Ireland Galway. *Quantum Qudit Simulation* Master Thesis, 2004

[8] B. Schumacher. See, for example, *Phys. Rev. A* 51 2738 (1995).

[9] A. Ekert, Patrick Hayden and Hitoshi Inamori. Centre for Quantum Computation Oxford OX1 3PU, United Kingdom. *Basic concepts in quantum computation,* 2000

[10] A. Barenco, C. H. Bennet, R Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. Smolin and H. Weinfurter, Phys. Rev. A 52 3457, 1995 *64*

[11] T. Toffoli, Mathematical Systems Theory 14 13 1981.

[12] Lov K. Grover. *A fast quantum mechanical algorithm for database search, Proc. 28th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1996.

[13] D. Deutsch. *Quantum Computational Networks*, Proc. Roy. Soc. Lond. A 425 73-90, 1989

[14] Wallace. *Quantum Computer Simulators –An Overview*, University of Exeter, Department of Computer Science Technical Report 387. www.dcs.ex.ac.uk/~jwallace, 1999.