Theoretical Computer Science 403 (2008) 121–129

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Universality and programmability of quantum computers

Willem Fouché [a], Johannes Heidema [b], Glyn Jones [c], Petrus H. Potgieter [a,*]

[a] *Department of Decision Sciences, University of South Africa, PO Box 392, Unisarand, 0003, Pretoria, South Africa*

[b] *Department of Mathematical Sciences, University of South Africa, PO Box 392, Unisarand, 0003, Pretoria, South Africa*

[c] *Department of Physics, University of South Africa, PO Box 392, Unisarand, 0003, Pretoria, South Africa*

ARTICLE INFO

Article history:
Received 8 October 2007
Received in revised form 30 April 2008
Accepted 12 May 2008
Communicated by M. Hirvensalo

Keywords:
Universal quantum computers
Programmable quantum computers
Universal quantum Turing machine
Universal quantum logic circuits

ABSTRACT

Manin, Feynman, and Deutsch have viewed quantum computing as a kind of universal physical simulation procedure. Much of the writing about quantum logic circuits and quantum Turing machines has shown how these machines can simulate an arbitrary unitary transformation on a finite number of qubits. The problem of universality has been addressed most famously in a paper by Deutsch, and later by Bernstein and Vazirani as well as Kitaev and Solovay. The quantum logic circuit model, developed by Feynman and Deutsch, has been more prominent in the research literature than Deutsch's quantum Turing machines. Quantum Turing machines form a class closely related to deterministic and probabilistic Turing machines and one might hope to find a universal machine in this class. A universal machine is the basis of a notion of programmability. The extent to which universality has in fact been established by the pioneers in the field is examined and this key notion in theoretical computer science is scrutinised in quantum computing by distinguishing various connotations and concomitant results and problems.

1. Introduction

In its attempt to cast light on the concept of programmability in quantum computing, this paper discusses two notions of "universality", namely

- a universal set of generating "components" for a given class of machines, and
- a universal machine in a class.

In his classic introduction [1] to the theory of recursive functions, Hans Hermes devotes Chapter 2 to the "engineering" of Turing machines (defined below). He introduces a finite set of elementary Turing machines and then describes how to combine them to build more complex ones and proves that the elementary machines constitute a universal generating set: any Turing machine whatsoever is equivalent (in terms of input–output behaviour) to a Turing machine constructed by combining elementary machines. Then, in the last chapter of the book, Hermes proves that there exists a universal Turing machine which can simulate the input–output behaviour of any Turing machine if provided with an appropriate program. These are instances of the two different notions of universality. Confusing and conflating the existence of a universal generating set with the existence of a universal machine can engender some conceptual perplexity, from which the literature on quantum computing does not seem to escape completely. This contribution examines these issues in more detail.

* Corresponding author. Tel.: +27 82 772 7990; fax: +27 12 429 4898.
E-mail addresses: php@member.ams.org, potgiph@unisa.ac.za (P.H. Potgieter).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.05.007

2. Classical and probabilistic Turing machines

By the beginning of the twentieth century mathematicians had become quite interested in establishing a formal model of computability. In 1936 Alan Turing described an abstract device, now called a *Turing machine*, which follows a simple, finite set of rules in a predictable fashion to transform finite strings (input) into finite strings (output, where defined). The Turing machine (TM) can be imagined to be a small device running on a two-way infinite tape with discrete cells, each cell containing only the symbol **0** or **1** or a blank. The TM has a finite set of possible internal states and a movable head that can read the contents of the cell of the tape immediately under it. The head may also, at each step, write a symbol to the cell over which it finds itself. There are two special internal states: an *initial state* q_0 and a *halting state* q_H.

A TM has a finite list of instructions, or *transition rules*, describing its operation. There is at most one transition rule for each combination of cell content (under the head) and internal state. If the internal state is q_i and the head is over a cell with content S_j then the machine looks for a rule corresponding to (q_i, S_j). If no rule is found, the machine enters the halting state immediately. If a rule corresponding to (q_i, S_j) is found, it will tell the machine what to write to the cell under the head, whether to move one cell left or right or to stay put and which internal state to enter. There is no transition rule corresponding to the halting state. Sometimes we refer to the entire collection of individual rules for all the different (q_i, S_j) as *the transition rule* of the machine.

A *computation* consists of starting the TM with the head over the first non-blank cell (which we may label position 0 on the tape) from the left of the tape (it is assumed that there is nothing but some finite *input* on the tape) and the machine in internal state q_0. Now the transition rules are simply applied until the machine enters the halting state q_H, at which point the content of the tape will be the *output* of the computation. If, for some input, the machine never halts then the output corresponding to that input is simply undefined. It is clear how every TM defines a (possibly, partial) function $f : \mathbb{N}_0 \to \mathbb{N}_0$ from the set of counting numbers, coded in binary, to itself.

Turing machines are the canonical models of computing devices. No deterministic device, operating by finite (but possibly unbounded) means has been shown to be able to compute functions not computable by a Turing machine. In fact, one may view one's desktop computer as a Turing machine with a *finite* tape.

A probabilistic Turing machine (PTM) is identical to an ordinary Turing machine except for the fact that at each machine configuration $\left(q_i, S_j\right)$ there is a finite set of transition rules (each with an associated probability) that apply and that a random choice determines which rule to apply. We fix some threshold probability greater than even odds (say, 75%) and say that a specific PTM computes $f(x)$ on input x if and only if it halts with $f(x)$ as output with probability greater than 75%.

3. Quantum Turing machines (QTMs)

A natural model for quantum computation is based on the classical Turing machine. The *quantum Turing machine* (QTM) was first[1] described by David Deutsch [3]. The basic idea is quite simple, a QTM being roughly a probabilistic Turing machine (PTM) with complex transition amplitudes (the squared moduli of which add up to one at each application) instead of real probabilities. A small number of quantum algorithms have been discovered but it has recently been shown that for at least one – Deutsch's algorithm – the efficiency of the quantum algorithm can be duplicated classically [4]. Without loss of generality everything can be assumed to be coded in binary, so that each position on the tape of the QTM will correspond to a single qubit (quantum bit). A unit of quantum information, the qubit is a two level quantum mechanical system, whose state is described by a linear superposition of two basis quantum states, often labelled $|0\rangle$ and $|1\rangle$. The actual (quantum) state space of the machine will be a direct sum of n-qubit spaces (where n is an indication of how much tape has been used, each n-qubit space being the n-fold tensor of the single qubit space). The direct sum is, however, not a complete inner-product space and therefore – by the postulates of quantum mechanics – not a valid state space. However, the underlying Hilbert space can be taken to be the completion of the direct sum and a unitary operator U on the direct sum (see [5]) can be extended to a unitary operator $\widehat{U}$ on the Hilbert space. This completed space and operator will correspond to the physical system associated with the QTM, thereby taking care of the *physicality* of the QTM.

3.1. Operation of a QTM

In the following the *classical machine* is a machine with a two-way infinite tape, starting over position 0 on the tape as described above, that we use as a kind of template for the quantum Turing machine. The corresponding quantum Turing machine (QTM) might work as follows (based on the Deutsch description [3], Ozawa [6], Bernstein and Vazirani [5]).

I. The quantum state space of the machine is spanned by a basis (here called the *computational basis*) consisting of states

$$|h\rangle|q_C\rangle|T_C\rangle|x_C\rangle$$

where $|h\rangle$ is the halt qubit, $h \in \{0, 1\}$ and (q_C, T_C, x_C) is a configuration of the corresponding classical machine, where x_C denotes the position of the head, q_C the internal state of the machine and T_C the non-blank content of the tape.

[1] Paul Benioff had related a similar idea somewhat earlier [2] but primarily in connection with presenting a possible physical basis for reversible computing.

II. Special initial and terminal internal states have been identified (corresponding to the initial state and halting state of the classical machine).

III. The single transition rule is now a unitary operator U which, in each step, maps each basic $|h\rangle|q\rangle|T\rangle|x\rangle$ to a superposition of only finitely many $|h'\rangle|q'\rangle|T'\rangle|x'\rangle$, where
   (a) the rule is identical for $|h\rangle|q\rangle|T_1\rangle|x\rangle$ and $|h\rangle|q\rangle|T_2\rangle|y\rangle$ when T_1 in position x and T_2 in position y have the same content, i.e. the rule depends only on the content of the tape under the head and the internal state q and not on the position of the head or on the content of the rest of the tape;
   (b) T' and T differ at most in position x;
   (c) $|x' - x| \leq 1$ (depending on whether the corresponding classical machine moves one position to the left, to the right, or not at all);
   (d) $h' = 1$ if and only if q' is the halting state of the classical machine; and
   (e) $T' = T$, $q' = q$ and $h' = h$ whenever $h = 1$.

   Finitely many subrules

$$|h\rangle|q\rangle|T\rangle|x\rangle \longmapsto \sum_{i=1}^{n} c_i|h_i\rangle|q_i\rangle|T_i\rangle|x_i\rangle \tag{1}$$

   will determine U as there are, by the stipulations above, only finitely many possible – given that the alphabet of the tape (binary in our case) and the number of internal states are both finite. Note that the transitional rule ("program") will have a finite specification only if the transition amplitudes in the superposition of the $|h'\rangle|q'\rangle|T'\rangle|x'\rangle$ are all *computable* complex numbers, which we will of course assume to be the case throughout. The transition rule can also, obviously, be extended (linearly) to finite superpositions of $|h\rangle|q\rangle|T\rangle|x\rangle$.

IV. The machine is started with a finite superposition of inputs

$$|0\rangle|q_0\rangle|T\rangle|x\rangle.$$

   Because of the form that the transition rule is allowed to take (and the fact that there are only finitely many internal machine states) the machine will be in the superposition of only *finitely many* basic states $|h\rangle|q\rangle|T\rangle|x\rangle$ at any step during the entire run[2] of computation.

The description of the machine given here differs from a classical reversible Turing machine in two obvious respects.

(a) Transition rules are allowed to map a state of the machine to the superposition of several states. The crucial distinction with classical probabilistic machines is that the QTM goes to a quantum superposition of states whereas the classical PTM can be seen as either going to a classical probability distribution over states or to a specific state with some classical probability. Quantum computing, of course, uses superposition in an essential way[3] – as in the algorithms of Shor or Grover.
(b) The input is allowed to be a superposition of a finite number of "classical" inputs.

It is not immediately obvious why a finite collection of specifications of the form (1) should necessarily define a unitary U, however, just as it might not be apparent why a finite collection of rules

$$|h\rangle|q\rangle|T\rangle|x\rangle \longmapsto |h'\rangle|q'\rangle|T'\rangle|x'\rangle$$

for a "classical" machine would necessarily specify a reversible machine. Unitarity is, of course, a precondition for the quantum device to be feasible.

We conjecture that the specification of what QTMs are, is at least consistent, that such beasties exist mathematically. Let the underlying classical template be a reversible TM, which, after reaching its halting state, keeps moving its head stepwise in one direction without ever changing anything on its tape. (Note that III.(e) does not stipulate that $x' = x$ whenever $h = 1$; to be reversible, something must change at every step.) The corresponding QTM is now constructed by linearly generating its U according to the corresponding transition rules of the simple form above – with no superpositions. Then this U is determined by a permutation of the computational basis of the QTM, and hence is unitary. Should the rules involve superpositions (as happens in all interesting cases), a proof of the unitarity of the induced U is called for.

3.2. Time evolution of the QTM and halting

If U is the operator that describes one application of the transition rule (i.e. one step in the operation) of the machine, then the evolution of an unobserved machine (where not even the halt bit is measured) for n steps is simply described by $V = U^n$.

[2] A more hazy concept than for classical Turing machines, as a QTM only really stops when one has observed the halt qubit and the content of the tape, so one may think of the transition rule being applied *ad infinitum*, step-by-step, unless the operator (physically, classically and externally) stops the machine.

[3] A very readable and accessible explanation of how and why this works can be found in [7].

If the first measurement occurs after n_1 steps, and the measurement is described by an operator J_1, then the evolution of the machine for the first $n_1 + j$ steps is described by

$$U^j J_1 U^{n_1}$$

which is in general no longer unitary since the operator J_1 is a measurement (always in the computational basis). It is important to note that the machine evolves unitarily only when no measurement takes place at all.

The output of the machine is on the tape as a superposition of basis states and should be read off after having measured the content of the halt bit and finding it in the state 1. The operator may at any time measure the halt bit[4] in order to decide whether to read the tape content (and collapse the state of the machine to one of the basis states). The halt bit is intended to give the operator of the machine an indication of when an output may be read off from the tape (and by observation collapsing the system to an eigenstate) without interfering excessively with the computation. It seems that Deutsch's original idea was that there would be no entanglement at all between the halt bit and the rest of the machine, but this cannot be guaranteed. The *output* of a QTM for some specific input x (which may be a superposition of classical inputs) is a probability distribution P_x over all possible contents of the tape at the time of observing the halt bit to have been activated. Note that the observation of an activated halt bit may in itself be a random event, but it has been argued by Ozawa [6] and others, that P_x does not depend on the random *observation* events.

4. Universality and programmability in the machine model

The notion of a *universal* computing device in a specific class is crucial for the development of a complexity theory and – more basically – establishes the notion of programmability. Naturally, we will start the discussion by reviewing the well-established notions of universality in classical deterministic and probabilistic computing before moving on to examine the concept of a "universal QTM" introduced by Deutsch.

4.1. Classical universality and programmability

Consider a general countable class of machines (or algorithms, procedures, programs), say *Manchester machines* (MMs), that compute partial functions, i.e. functions that are not necessarily defined for all inputs (since the machine might not halt, for example, as in the case of a Turing machine). Since there are only countably many machine descriptions, let us assume that each Manchester[5] machine is fully described by a natural number. It should be possible to recover the full description of the machine's functioning from the natural number in an effective way, so it should not simply be any enumeration of the countable set. Let Φ_n denote the partial function computed by machine n and fix an MM-computable bijective function[6] $h : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$, assuming such a function exists.[7]

More specifically for the following Definition 1, think intuitively of h as an abstract 'compiler', which prepares the 'program' n and the 'input' m in the form $h(n, m)$, suitable as input for machine number N to start simulating the computation of machine number n on input m. Beyond the MM-computability of h (most likely by some special-purpose MM-like machine, different from both machines N and n), there is the practical question of the complexity and tractability of h and the choice of an MM-type machine to preprocess (n, m) into $h(n, m)$ as input to N. These questions, important as they are, are left for now; but similar questions can be asked in Section 4.3, where preprocessing of input to the 'universal' quantum component of a "semi-universal hybrid device" is required.

Definition 1. If there exists a number N such that

$$\Phi_N\,(h(n, m)) = \Phi_n(m)$$

which means that the functions are either equal and both defined or both undefined, for all n and m, then the machine described by N is called a *Universal Manchester Machine* (UMM).

Programmability is firmly linked to the concept of universality and is, of course, a necessary condition for universality. Is it a sufficient condition? A particular Turing machine is usually thought of as dedicated to a particular task, defined by a set of quintuples describing the operations to be carried out in sequence. Every Turing machine has thus a finite description (of internal states, tape entries and operation rules – which are unbounded but finitely many) which could be used as input to another Turing machine.

4 The halt *qubit*, of course, until we measure it.

5 Alan Turing worked on building and programming one of the first electronic computers in the city of Manchester after the Second World War.

6 It will strike the attentive reader that h is the first (and last) function of two variables to appear here but that we have implicitly considered Manchester machines with one counting number input only. This is illogical, but the problem can be fixed in a well-established way. Suffice it to say that one should be able to consider h MM-computable in an obvious and logical way. One only needs *one* such function h and we will therefore not elaborate here.

7 If it does not, the class of machines would really be very poor. It would not make a big difference if we took, for example, $h : (x, y) \mapsto 2^x 3^y$ instead of an onto function but the convention that h be onto is harmless and convenient.

A universal Turing machine, (of which there are infinitely many), can simulate all the Turing machines, and is thus programmable for the entire class of Turing machines. If a machine is programmable for any device in its class, then it is universal. Not all programmable computer devices are universal in any sense. In fact, one could use the term "programmable" in the context of, say, MMs to describe any device taking a certain non-empty set of inputs of some form depending on (n, m), where n is the "program" and m the "data", and where the action of that device on such an input depends not only on m but also on n. Such machines are universal (for some class, say the class of all MMs) only if they can – through suitable choices of n – mimic the operation of any other machine in the class on any relevant m.

4.2. Probabilistic Turing machine universality

Since halting is a probabilistic notion for a QTM, the notion of universality for quantum devices should be akin to that for probabilistic machines. For probabilistic machines, however, Definition 1 does not directly apply and it is necessary to generalise it as follows.

Definition 2. If there exists a number N such that

$$\Phi_N\left(h(n, m)\right) = \Phi_n(m)$$

which means that the functions are either equal and both defined or both undefined **(if deterministic)** and **if not deterministic** then the values have the same distribution, for all n and m, then the machine described by N is called a *Universal Manchester Machine* (UMM).

In the case, for example, of deterministic Turing machines (which are a strict subset of the probabilistic machines) the two notions of universality coincide, of course. The main aim of this section is to discuss this (second) notion of universality for quantum Turing machines (QTMs).

One can easily show, incidentally, that every function f which can be computed in this sense by a PTM, is also computable by some ordinary TM in the usual sense. Nevertheless PTMs have always been of interest because probabilistic algorithms can often be found that are quite fast by comparison to the best known classical procedure. The class of PTMs is often defined by restricting the probabilities to $\frac{1}{2}$ or 1 only. In this case the class can also be obtained by taking the ordinary TMs and adding a special write instruction to write one random bit to the tape. The PTMs are often described, in this model, as "TMs with access to a fair coin toss". It is easy to see how a universal machine might be described in this class: it would simply be a universal TM equipped with the random output instruction. Such a *universal PTM* (UPTM) could obviously simulate any other "coin toss" PTM perfectly, by which is meant that the output of the UPTM would have exactly the same distribution as the output of a PTM for which it is executing a program.

Now, which PTMs should our UPTM be able to simulate exactly? Well, since each PTM should have a finite description, the UPTM need only be able to simulate a countable collection of PTMs. Let us restrict the set of PTMs to those with *computable* transition probabilities. Each such machine is fully described by the finite set of transition rules and programs for computing each of the associated probabilities. This description is finite – thanks to the restriction of the probabilities to computable numbers.

Since there is no reasonable way of giving a finite description of PTMs with non-computable transition probabilities, apart from the usual paradoxical definitions of the type "one more than the largest number which can be described in thirteen words", this concludes the discussion for PTMs. Introducing arbitrary real transition probabilities makes no sense as it would immediately make any subset of the natural numbers decidable by a probabilistic machine.

4.3. A universal QTM?

Deutsch introduced a "universal quantum computer" (uQC, where u has not been capitalised in order to emphasise the difference between this universality concept and the preceding) in [3]. The Deutsch uQC is in effect a QTM as in Section 3, based on a classical UTM with some additional (8 in [3]) operations that allow any unitary transformation on one qubit to be approximated arbitrarily closely. Deutsch showed in the paper that for any given $L, \varepsilon > 0$ and quantum device U operating on L qubits, there exists a program p_L (a classical finite string of bits) for the uQC that (with input $|p_L\rangle$ followed by any finite superposition of L-qubit basic states) approximates the operation of U on the finite superposition of L-qubit basic states with accuracy at least ε (in the inner-product norm). This is not the same kind of universality that we have for probabilistic and for deterministic Turing machines and even the concatenation scheme used by Deutsch has been questioned (for example, by Shi [8]).

Now, if we consider the earlier (second) definition of universality, then there can be no universal machine for the simple reason that in Deutsch's scheme there are uncountably many (transition rules for) QTMs. For broadly the same reasons as outlined above for PTMs, we shall restrict ourselves henceforth to QTMs with computable transition amplitudes, i.e. transition amplitudes for which both the real and imaginary parts are computable numbers. We now fix a scheme for encoding the QTMs and associate any machine M with the smallest[8] natural number that encodes it. Note that we say

[8] Two distinct natural numbers may, of course, encode physically identical machines.

that a QTM outputs y with probability p if the probability of *ever* observing the machine to be in the halt state with the tape in state $|y\rangle$ is p. Does a universal machine for the (restricted) class of QTMs in the sense of Definition 2 exist?

Deutsch provided the rather incomplete solution mentioned above. Bernstein and Vazirani [5] have given another partial solution. They showed that there exists a quantum Turing machine $\mathcal{U}$ (they actually wrote $\mathcal{M}$) such that [5]

> "for any well-formed[9] QTM M, any $\varepsilon > 0$, and any T, $\mathcal{U}$ can simulate M with accuracy ε for T steps with slowdown polynomial in T and $\frac{1}{\varepsilon}$".

The slowdown and the program for $\mathcal{U}$ *both depend here on the length of the input*. The full Bernstein–Vazirani result could be summarised by the statement that

> there exists a QTM $\mathcal{U}$ such that for each QTM M with finite description $\bar{M}$, n, ε and T there is a program $\mathcal{P}(\bar{M}, n, \varepsilon, T)$ and a function $f_{\bar{M}}(T, n, \frac{1}{\varepsilon})$ (both recursive in their inputs) such that running $\mathcal{U}$ on input $|\mathcal{P}(\bar{M}, n, \varepsilon, T)\rangle \otimes |x\rangle$ where $|x| = n$ for $f_{\bar{M}}(T, n, \frac{1}{\varepsilon})$ steps results – within accuracy ε – in the same distribution over observable states as running M on input $|x\rangle$ for T steps.

The simulation is clearly *only approximate*. What Bernstein and Vazirani mean "with accuracy ε" is that if P is the probability distribution over all observable states of $\mathcal{U}$ after $f_{\bar{M}}(T, n, \frac{1}{\varepsilon})$ steps with the given input and Q is the corresponding probability distribution of M after T steps then

$$\frac{1}{2}\sum_{x} |P(x) - Q(x)| \leq \varepsilon$$

where the summation is over all possible observable states x. Again, approximate simulation is quite different from the universality concept for ordinary and for probabilistic Turing machines (with computable probabilities) as in the latter cases the universal machine's simulation was *exact*. Running $\mathcal{U}$ for exactly $f_{\bar{M}}(T, n, \frac{1}{\varepsilon})$ steps on any input $|\mathcal{P}(\bar{M}, n, \varepsilon, T)\rangle \otimes |x\rangle$ will have simulated the running of M on $|x\rangle$ for T steps. We may not let $\mathcal{U}$ run for any more steps as the state of the machine might then drift away from the to-be-simulated state of M after T steps. This behaviour is rather different from that of the UTM or UPTM—where there is no need to restrict the number of steps executed!

What about the input to the machine? In general, the input to a QTM is allowed to be a (finite) superposition of basis states of the tape but the Bernstein–Vazirani theorem quoted here applies to a single state only. This is not a problem: it is straightforward to see that it also applies to a superposition of m basic states (just replacing ε by $\frac{1}{m}\varepsilon$).

Now, the Bernstein–Vazirani machine $\mathcal{U}$ immediately suggests the following semi-universal hybrid device (SUHD). The device takes the description $\bar{M}$ of a QTM M as well as x and ε (which may be taken to be rational) as input. The machine operates as follows.

```
$T$:= 1;
$n$:= $|x|$;
do
   compute $P$ := $\mathcal{P}(\bar{M}, n, \frac{\varepsilon}{T}, T)$;
   compute $S$ := $f_{\bar{M}}(T, n, \frac{T}{\varepsilon})$;
   run $\mathcal{U}$ on $|P\rangle \otimes |x\rangle$ for S steps;
   signal that quantum part of device may be observed;
   wait a little;
   reset quantum part of device;
   $T$:=$T$+1;
while true;
```

Note that by replacing ε by $\frac{\varepsilon}{T}$ we have ensured that by simply letting the SUHD run, we will not only be able to observe the simulated behaviour of M for ever longer times, but also with ever-increasing accuracy. However, the SUHD is still not universal for the class of QTMs in the sense of Definition 1 or Definition 2. This is true not only for the very obvious reason that its simulation is only *approximate*, but for the much more fundamental reason that we do not know whether it is a QTM itself!

The SUHD is a real hybrid device which consists of a classical Turing-type machine and a quantum part. The classical TM-part computes the P and S above, and $|P\rangle$ then becomes part of the input to the quantum part. Beyond the TM-computability of P and S, we know nothing about the complexity of these computations, similar to our agnosticism about the complexity of the 'compiler' function h in Section 4.1. The SUHD is – in a sense – a robot capable of operating a quantum device (which forms part of itself) and there is no reason to think that such a robot cannot be built. The problem lies therein that the robot only gives a signal when we might observe the quantum part of the device. It cannot know whether we have observed the quantum part or not – otherwise the observer would become part of the device…

[9] Meaning that the time evolution operator is unitary.

Now, any quantum device operates reversibly. In the case of the SUHD the step "reset quantum part of device" is the part which can be problematic in this regard. If the quantum part was not observed during the step "wait a little" then the inverse of the evolution operator of $\mathcal{U}$ can be used to effect such a reset. But, what if the observer(s) did make an observation of the quantum part during "wait a little"? Now, the inverse of the evolution operator of $\mathcal{U}$ will *not* "reset quantum part of device". This is really a serious problem. In an ordinary QTM the evolution of the machine continues even when the halt bit has been observed, but for the SUHD even the observation of the halt bit (which may be in a superposed state, although not necessarily entangled with the rest of the machine) renders the operation of the device non-reversible. This is simply because for the ordinary QTM, the evolution operator can continue after the halt bit has been observed without perturbing the probability distribution that has been defined to be the QTM's *output* (according to Ozawa [6]) since the observation projects one, in a certain sense, only into a specific ($h = 0$ or $h = 1$) branch of the computation. For the hybrid device it is not that simple since the resetting step requires an undisturbed quantum part. If the quantum part has been disturbed at $T = k$, the operation described above will not be able to correctly reset the quantum part of the device and will not execute the loop faithfully for $T = k + 1$. Of course, it is always possible for the operator to be instructed to restart the hybrid device after observation, but then we will be dealing with a new kind of bio-hybrid device and not a universal machine at all. In classical computing this would be the equivalent of the user strictly having to reboot the computer each time after looking at the screen, i.e. there would be no *autonomy* of operation. Pure quantum computing devices are prevented by the No Cloning Theorem from copying initial configurations of substems, which precludes the realisation of such a naïve hybrid operation by a quantum device.

Conjecture 1. *The SUHD derived from Bernstein and Vazirani's $\mathcal{U}$ cannot be made to operate reversibly and is therefore not a QTM.*

The immediate consequence of the conjecture is that (as yet) no universal machine has been shown to exist in quantum computing and that the notion of universal programmability has not really been established for quantum computing in the QTM model.

5. Quantum gates

Quantum gates provide another (and more practical) engineering paradigm for quantum computation initiated by Richard Feynman [9] and David Deutsch [10]. The classical analogue is a logic circuit. In principle, in the quantum gate model, a quantum computation works as follows.

(1) The first step typically involves the preprocessing of the input data on a *classical* computer. For example, in the Shor algorithm for the factoring problem we must ensure in a classical way that the input number is not a prime power.
(2) Based on these preprocessed data, we have to prepare the quantum register. This means, in the simplest case, to prepare classical data e.g. a binary string x of length d, say, as the state $|x\rangle$ in 2^d-dimensional Hilbert space. In most cases, however, one would be required to prepare a superposition of states $|x\rangle$.
(3) Next we apply the quantum circuit C, which is a sequence of local quantum operators, to the input state $|\phi\rangle$ and after the calculation we get the output state $U|\phi\rangle$ where U is the unitary operator corresponding to C.
(4) To read out the data we perform a von Neumann measurement in the computational basis.
(5) Finally we may have to post-process the value on a classical computer. In general we obtain a correct result with probability less than one, which means we have to check the validity of the result with a polynomial time algorithm and if wrong, we have to go back to step 2.

Hence, in this model, a quantum computation is a hybrid of classical and probabilistic algorithms coupled with quantum evolutions of prepared quantum states.

Suppose V is a unitary operator acting on $\mathcal{H}^{\otimes f}$. For $d \geq f$, we call a unitary operator on $\mathcal{H}^{\otimes d}$ an *instance* of V if it is any operator acting like V on a fixed f of the possible d qubits and as the identity on the remaining qubits. In order to discuss programmability in this context, we introduce the idea of instruction sets. An *instruction set* G for a multiqubit of a fixed length d is a *finite* set of quantum gates satisfying the following conditions.

- All gates $V \in G$ are in $SU(2^d)$, that is, they are unitary operators on the 2^d-dimensional Hilbert space $\mathcal{H}^{\otimes d}$ where $\mathcal{H}$ is 2-dimensional over $\mathbb{C}$ and each operator has determinant one.
- For each $V \in G$ the inverse operation $V^\dagger$ also belongs to G.
- The group generated by G is topologically dense in $SU(2^d)$. This means that for any given quantum gate $U \in SU(2^d)$ and any degree of accuracy $\epsilon > 0$, there exists a finite product $V = V_1V_2 \ldots V_k$ of instances of gates from G which is an ϵ-approximation to U, that is to say, such that

$$||U - V_1V_2 \ldots V_k|| < \epsilon.$$

Suppose U and V are two unitary operators on the same state space with U the target unitary operator that we wish to implement and $V = V_1V_2 \cdots V_k$ being the unitary operator that is actually implemented from an instruction set as above. Let M be a positive operator valued measure (POVM) element associated with the measurement and let P_U (or P_V) be the probability of obtaining the corresponding measurement outcome if the operation U (or V) is performed with a starting state $|\phi\rangle$. Then it can be shown that (see [11], p. 195)

$$|P_U - P_V| = |\langle\phi|U^{\dagger}MU|\phi\rangle - \langle\phi|V^{\dagger}MV|\phi\rangle| \leq 2||U - V||.$$

This inequality gives quantitative expression to the idea that when the error $||U - V||$ is small, the difference in probabilities between measurement outcomes is also small.

An example of a set of universal gates is that "generated" by instances of T, the Toffoli gate, H, the Hadamard gate, and the phase gates. It is "generated" in the following sense: We consider all unitary operators for d-multiqubits which is a product of instances of H, T, the phase gates together with their inverses. Then this set G is an instruction set for multiqubits of length d.

6. Universal sets of gates: The Solovay–Kitaev theorem

The problem of quantum compilation is the following: Given an instruction set G, how can we approximate an arbitrary quantum gate by means of a finite sequence of instructions from G in a manner which is both effective (i.e., computable in the classical sense), and efficient as far as both the time and space complexity are concerned. The Solovay–Kitaev theorem [12] is a truly remarkable contribution to this problem:

Theorem 1. *Let G be an instruction set for $SU(2^d)$, and let a desired measure of accuracy $\epsilon > 0$ be given. There is a universal constant c such that for any U in $SU(2^d)$, there exists a finite sequence S of gates from G of length $O_d\left(\log^c(1/\epsilon)\right)$ such that the product of the sequence S is within ϵ of U with respect to the operator norm.*

More precisely, an arbitrary unitary operator U on d qubits can be approximated to within a distance ϵ in the operator norm by using $O\left(d^2 4^d \log^c(d^2 4^d/\epsilon)\right)$ instances of gates from G. This can be shown [11, pp. 199–200] to be close to optimal in the following sense: For a given instruction set G and a measure of accuracy $\epsilon > 0$, there are unitary transformations U on d qubits which take $\Omega(2^d \log(1/\epsilon)/\log(d))$ instances of gates from G to implement an approximation V such that $||U - V|| < \epsilon$.

Many authors state that this can be done in an effective and efficient manner. This must be read with some care! We call a unitary operation *recursive* with respect to the chosen measurement basis if all its matrix entries relative to this basis are recursive complex numbers. Recall that a complex number is a *recursive* complex number provided both its real and imaginary parts are recursive real numbers. A real number x is recursive if there is an algorithmic procedure which with input a natural number n will yield a binary rational number of the form $\ell/2^n$ such that $|x - \ell/2^n| < 1/2^n$. Suppose now that all the matrix entries of the gates in G with respect to the orthonormal basis in which the measurement is performed are recursive complex numbers, but that U is not recursive relative to this basis. Suppose we have an effective procedure that will yield for any given natural n descriptions of instances $V_1, \ldots, V_k$ of gates from G such that $||U - V_1 \cdots V_k|| < 1/n$. Then it is clear that all the matrix coefficients of U with respect to the measurement basis are recursive complex numbers – a contradiction. Our impression is that it may be possible to compute U effectively provided U is recursive with respect to the measurement basis. Then, the time complexity of finding the sequence S of gates will depend on the complexity of determining the elements of U and of doing the required algebraic operations. The claim is that this accuracy can be obtained using $O_d\left(\log^{2.71}(1/\epsilon)\right)$ computational steps. As we understand matters at this stage, this is correct if the computation is relative to an *oracle* that has complete information about U with respect to the measurement basis. It remains to be investigated how the recursive complexity of U affects this claim. It is also ambiguous to state that the approximation can be done in an efficient manner, for as stated above, the worst case approximation will always be at least exponential in the length of the multiqubit on which the unitary transformations operate.

7. Conclusion

Research into quantum computation over the past 20 years has been very successful in stimulating the development of quantum cryptography (already in industrial application), the study of quantum information and the discovery of novel algorithms for traditionally hard and interesting problems such as prime factorisation. One would be wise, however, to heed the words of Andrew Steane [13]:

> The title quantum computer will remain a misnomer for any experimental device realised in the next twenty years. It is an abuse of language to call even a pocket calculator a computer, because the word has come to be reserved for general-purpose machines which more or less realise Turing's concept of the universal machine. The same ought to be true for QCs if we do not want to mislead people.

This paper has attempted to explain why certain (strong and interesting) results in quantum computation still fall short of establishing universality (and programmability) for quantum computing. At the very least, researchers in the field should attempt to explain how the results of Deutsch, Bernstein and Vazirani, Solovay, Kitaev and others can be used or expanded to construct a fully programmable universal quantum device. In the worst case, one needs to prove that such a fully universal quantum computer does not exist. We conjecture that, like the QTMs and quantum circuits, other approaches to quantum computation which boast "universal resources" (e.g. measurement-based quantum computation [14]) may confront similar conceptual problems in the quest for a universal programmable machine.

References

[1] H. Hermes, Enumerability, Decidability, Computability. An Introduction to the Theory of Recursive Functions (G.T. Hermann and O. Plassmann, Trans. from German), second ed., Springer-Verlag New York, Inc., New York, 1969 (Die Grundlehren der mathematischen Wissenschaften, Band 127).
[2] P. Benioff, The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, Journal of Statistical Physics 22 (1980) 563–591.
[3] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 400 (1985) 97–117. URL http://links.jstor.org/sici?sici=0080-4630%2819850708%29400%3A1818%3C97%3AQTTCPA%3E2.0.CO%3B2-A.
[4] C.S. Calude, De-quantising the solution of Deutsch's problem quant-ph/0610220, URL http://arxiv.org/abs/quant-ph/0610220.
[5] E. Bernstein, U. Vazirani, Quantum complexity theory, SIAM Journal on Computing 26 (1997) 1411–1473. URL http://link.aip.org/link/?SMJ/26/1411/1.
[6] M. Ozawa, Quantum nondemolition monitoring of universal quantum computers, Physical Review Letters 80 (1998) 631–634.
[7] Lance Fortnow, One complexity theorist's view of quantum computing, Theoretical Computer Science 292 (3) (2003) 597–610.
[8] Y. Shi, Remarks on universal quantum computer, Physics Letters A 293 (5–6) (2002) 277–282.
[9] R.P. Feynman, Quantum-mechanical computers, Foundations of Physics 16 (1986) 507–531.
[10] D. Deutsch, Quantum computational networks, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 425 (1989) 73–90. URL http://links.jstor.org/sici?sici=0080-4630%2819890908%29425%3A1868%3C73%3AQCN%3E2.0.CO%3B2-5.
[11] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, UK, 2000.
[12] A.Y. Kitaev, A.H. Shen, M.N. Vyalyi, Classical and quantum computation, in: Graduate Studies in Mathematics, American Mathematical Society, Providence, Rhode Island, 2002.
[13] A. Steane, Quantum computing, Reports on Progress in Physics 61 (2) (1998) 117–173. URL http://stacks.iop.org/0034-4885/61/117.
[14] M.V. den Nest, A. Miyake, W. Dur, H.J. Briegel, Universal resources for measurement-based quantum computation, Physical Review Letters 97 (2006) 150504-4. URL http://link.aps.org/abstract/PRL/v97/e150504.