THE EMERGENCE OF NEW MODELS OF COMPUTING: FROM DIGITS TO QUANTUM COMPUTING AND BEYOND

John E. Gray and A. D. Parks
Code N92 and Code B10
Naval Surface Warfare Center Dahlgren Division
Dahlgren, VA
22448
jgray@nswc.navy.mil and dparks@nswc.navy.mil

ABSTRACT

We begin with a brief review of the basic notions of classical computability theory and the Turing machine. The concept of decidability and computability, as well as the Church-Turing thesis are also introduced. The recently formalized foundations of quantum computability theory are surveyed next. Included are such topics as the universal quantum computer; the use of quantum parallelism in computation; and the Church-Turing principle. We highlight some avenues of research for new computational paradigms and suggest that underlying the new advances in both the theory and practical aspects (actual devices), is a new conceptual basis for interpreting the interactions that produce complex and partially understood processes.

keywords: computation, quantum computing, Turing machines, Church-Turing thesis, generalized superposition

International Journal of Computing Anticipatory Systems, Volume 2, 1998
Ed. by D. M. Dubois, Publ. by CHAOS, Liège, Belgium. ISSN 1373-5411 ISBN 2-9600179-2-7

1. INTRODUCTION

All higher animals seem to have internalized certain models of computation that enable them to distinguish between the *one* and the *many* (Melzak 1976). Like tool making, counting skills exhibit a graceful degradation from the higher to the lower animals. As civilization evolved, more sophisticated models of arithmetical operations were developed to meet contemporary and, sometimes, future needs. From Archimedes' "The Sand Reckoner" to the Arabic utilization of zero, elementary methods to aid computation were adapted by various civilizations as they recognized the superiority of methods developed by others. Gradually, it was realized that mechanical means could be used to perform arithmetic operations. Both Pascal and Leibniz dreamed of artificial means to perform the drudgery of computation. They even considered artificial means of thought. By the nineteenth century, further innovations in arithmetic seemed unlikely and the search for improved methodology led instead to a search for improved mechanical means to accomplish computation. The British navy funded Babbage's research concerning the construction of a mechanical gear based computer which he termed the "Analytical Engine". Although it was never built it was realized that this machine could act on symbols other than numbers. This led to some of the first notions concerning machine programming such as those suggested by Lady Lovelace.

The late ninetieth century saw logic reduced to simple arithmetic by Boole and others. Early in the twentieth century, Russell and Whitehead wrote a book entitled *Principia Mathematica* which purported to derive mathematics from logic without contradiction. David Hilbert then proposed a challenge to his fellow mathematicians to prove by the methodology of Russell that mathematics was both *consistent* (contradiction-free) and *complete* (every true statement could be derived within the conceptual framework set within Principia). This would reduce mathematics to mechanical theorem proving which could then theoretically be done on a machine. This program was demolished by Gödel in the early thirties when he showed that consistency is not compatible with completeness. Alan Turing, using the insight gained from Gödel, quickly managed to prove a similar result applicable to computing machines. As part of the process of doing this, he defined conceptually what is meant for a machine to be a computer, i.e. he defined those operations required for a machine to do elementary arithmetic. Furthermore, his machine is universal in the sense that anything normally thought of as computing could be done by his machine. Shortly after this work was done, the first mechanical universal computers were built. After World War II, the first electronic computers were built and the applications of this new tool to a wide variety of problems quickly grew. In recent years, theoretical computer science has been concerned with two main questions based on Turing's model of computers: can one develop rules that capture the essence of intelligence (AI); and what, if any, are the natural limitations imposed upon computation by a Turing machine model? In this paper, we ignore the first of these questions and concentrate on the second. The second question has led to the study of the computational complexity of various problems, as well as that of certain natural processes in the external world.

A renewed interest in the foundations of quantum mechanics has recently been exhibited within the physics community dealing with such fundamental issues as quantum measurement theory, wave-particle duality, nonseparability, decoherence, and associated geometric interpretations. Concurrently, the computer science community has directed its efforts towards a more thorough understanding of such basic concepts as computational complexity, parallel/distributed processing, reversible computation, and component miniaturization. A new cross-disciplinary science is emerging based upon the growing recognition of the many common aspects that physics, certain biological systems, and theoretical computation share. Revolutionary advances in micro- and nano-technologies, as well the highly competitive and demanding environment of the information age, is driving a rapid evolution of the associated technologies to new unconventional approaches for collecting, storing, processing, and transferring information. Quantum mechanical superposition is just one example of a host of families of types of superposition that Nature provides to us.

Biology, even more than physics, provides a rich source of potential interactions that may also serve as new models of computational processes. Thus, we believe that the search for new ways to combine data based on new types of superposition (generalized superposition or homomorphic) will likely lead to a deeper understanding of complex systems. Ultimately, we believe that these complex systems are both amenable to computation and serve as a source for potentially new models of computation more general than the "universal computation model" proposed by Turing. Alternatively, they may help provide an understanding of hierarchies of computational models similar to Gödel's discussion of how adding additional axioms to a system results in a "speedup" in theorem proving.

We begin with a brief review of the basic notions of classical computability theory and the Turing machine. The concept of decidability and computability, as well as the Church-Turing thesis are also introduced. The recently formalized foundations of quantum computation theory are surveyed next. Included are such topics as the universal quantum computer; the use of quantum parallelism in computation; and the Church-Turing principle (a physical extension of the Church-Turing thesis). The massive parallelism that can be obtained from quantum mechanical superposition, as well as the associated interference effects are currently being explored as novel approaches to solve difficult computational problems. Finally, we speculate on some avenues of research for new computational paradigms and suggest that a new understanding of the interactions that produce complex physical process will likely result from the quest for these new paradigms.

2. MODELS OF COMPUTAION

2.0 Introduction

The basic notions of classical computability theory (these discussions are based on Hopcroft (1979) and Sudkampf (1988)) are of fundamental importance to our discussion. Since these ideas may be unfamiliar to readers whose focus is not in the computational sciences, we provide an overview of the basics that are found in the standard texts. Classical computability theory is structured around the formal notion of a classical Turing machine (TM) in terms of *effective processes*, i.e. those processes which can be performed in a determinate and precisely specified manner using steps which can only be executed by finite *mechanical* means. Thus, in order that a process be effective, it must possess the following properties:

- *mechanistic*- it consists of a finite sequence of instructions each of which can be carried out without insight, ingenuity or guesswork
- *deterministic*- when presented with an input string, it always produces the same result

Note, that in general, it is not sufficient to have a pseudo-algorithm as a solution to a decision (yes/no) problem; one must be able to transform the decision problem from its natural domain into an equivalent problem that can be decided by a TM. This *construction of a representation of the problem* is not trivial and is often ignored in discussions about computation. However, in physical applications it is the important problem while the TM aspect, i.e. computational model, is considered trivial.

2.1 Turing Machines

A TM can be pictured as a box with a tape running through it. The tape consists of a sequential collection of squares which may extend infinitely in either direction. The box is capable of being in any one of the internal states contained in a finite set Q, where $q_0 \in Q$ is always the initial internal state. It is also able to scan or print on the tape any symbol in a finite set S which contains as an element a distinguished symbol "b". The machine is started by being given a tape, which may have some symbols printed on it (one to a square), and by being set to scan some tape square while in some initial internal state q_0. The action of the machine is determined by a partial transition function Θ: $Q \times S \rightarrow Q \times S \times \{L,R\}$, where $L(R)$ means shift the tape one square to the left (right). Thus, at any instant, the box is in an internal state $q_i \in Q$ and is scanning a tape square that bears a single symbol $s_i \in S$. The machine may continue working indefinitely or may eventually stop when the internal state and symbol scanned form a pair not in the domain of Θ. Note the transition function can be considered to be equivalent to an instruction set of quintuples of the form $q_i s_j q_k s_m \delta$, $\delta \in$ {L,R}, with the meaning: "if the machine is in internal state q_i and scanning symbol s_j, then (1) change to internal state q_k; (2) replace s_j with s_m; and (3) shift one tape square in direction δ." Because of this equivalence, "transition function" and "instruction set" are used interchangeably. A TM operates on

an infinite tape, though only a finite number of tape squares are occupied at any instant by symbols other than "b". It is therefore possible to provide an instantaneous description of the TM using a string consisting of a finite number of elements of S (possibly with repetitions) and a single element of Q which is not the rightmost member of the string. This string represents the symbols on the tape, the internal state, and the tape symbol being scanned. If α_i and α_{i+1} are instantaneous descriptions, a basic move, denoted $\alpha_i \rightarrow \alpha_{i+1}$, means that α_{i+1} is obtained from α_i by application of Θ to the associated internal state-scanned symbol pair in α_i. A computation of a TM is a finite sequence of basic moves $\alpha_i \rightarrow \alpha_{i+1}$, $i = 0,1,2,...,$ n-1, where α_n is terminal, i.e. there is no β where $\alpha_i \rightarrow \beta$.

Let M be a TM with state and symbol sets Q and S respectively. If w is a finite string comprised of symbols (possibly with repetitions) from S, a *word*, then $M(w)$ exists if there is a computation with $\alpha_0 = q_0 w$, $q_0 \in Q$. If Ω is the set of all words on S and $\lambda(M) = \{w \in \Omega: M(w) \text{ exists}\}$, then M is said to enumerate λ(M). This set of words is also referred to as the language accepted by M. Note that for any $w \in \Omega$, there is no way of telling *a priori*, whether $M(w)$ exists. If Ω is the set of words on any finite set of symbols, then a subset E of Ω is *recursively enumerable* (r.e.) when there exists some TM that enumerates (or accepts the language) E. Thus, every TM defines a r.e. set. A r.e. set E is *recursive* if its complement E' in Ω is r.e., so that if M and M' enumerate E and E', respectively, then either $M(w)$ or $M'(w)$ exists for each $w \in \Omega$. Thus, it can be decided in a finite length of time whether or not $w \in E$.

A *decision problem* is a general question which contains a description of all parameters and which has either a "yes" or "no" as an answer. Many general problems can be stated without loss of solution difficulty as decision problems. The question of whether there exists a TM for solving such a problem is equivalent to determining whether an associated language encoding instances of the problem using a finite set of symbols is recursive. If the language is recursive, then the problem is said to be *Turing decidable*. Otherwise, it is *undecidable* and there is no TM which determines the answer.

The TM may also be viewed as a computer of functions. Let Ω be the set of words on a finite symbol set and Ω^n be a n-fold Cartesian product. A partial function f: $\Omega^n \rightarrow \Omega$ is Turing computable if M is a TM such that for all $x_1,x_2,..x_n$, $y \in \Omega$, when M is started with tape configuration $x_1 b x_2 b \ldots b x_n$, it terminates with tape content y when $f(x_1,x_2,\ldots,x_n) = y$ and does not terminate if $f(x_1,x_2,\ldots,x_n)$ is undefined.

Decision problems or functions which are evaluated using effective processes are said to be *effectively decidable or effectively computable*. A TM is an **attempt** to formalize what is meant by an effective process. While there is general agreement that an effective process captures what is meant by computation, it cannot be proven that there exists a TM for every effective process. Thus one has the formalization of this correspondence, the **Church-Turing thesis (CT)**: *A problem (function) is effectively decidable (computable) if, and only if, it is Turing decidable (computable).* Adopting the CT is tantamount to

assuming a formal theory of computation, because it implies the existence of a well defined boundary between that which is decidable (computable) and that which is not. The statement "the problem P is not Turing decidable" and "the function ϕ is not Turing computable" assert within the context of the CT that there are no TM's which solve P or compute ϕ.

The model for probabilistic computation is obtained from the TM by allowing machines access to the simplest type of randomness. A *probabilistic TM* (PTM) is a TM with distinguished internal states called "coin tossing states". For each such state there are two possible next states. PTM computations are deterministic except for "coin tossing" states where an unbiased coin is "tossed" by the machine to decide the next state. An input word is accepted by a PTM only if it produces a final output word with a probability greater than 0.5. The class of partial functions computed by PTM's is the same as that computed by TM's. Nondeterministic TM's arise when the codomains of their transition functions are finite subsets of those for the associated deterministic machines. This implies, in general, that there are a finite number of fixed transition choices for any domain element. Although the addition of nondeterminism to TMs does not change their overall language acceptance capabilities, these type of machines remain useful for the study of computational complexity.

One of Turing's most interesting findings is that there exists *a universal TM (UTM)*, which is a TM that is capable of simulating any other TM. It can be shown that the TM's can be enumerated by an effective process. If $M_1, M_2, \ldots$ are all the TM's, then given an n, M_n can be effectively found. In addition, if an initial tape configuration w is provided, then $M_n(w)$ can be computed if it exists. Thus, the procedure of going from a pair (n,w) to $M_n(w)$ is an effective process and the TM which implements this process is the UTM denoted by U. (Observe that as a result of this, we may make the following equivalent restatement of CT: *a function is effectively computable if, and only if, it can be computed by a UTM.)* The UTM is an important abstraction of the general purpose computer that has proven useful. Given any digital computer with any input, there is always an n and w such that $U(n,w)$ produces the same output. Conversely, if a computational problem cannot be solved by U, then it cannot be solved by any digital computer.

2.2 The Machine/Language Hierarchy

As noted above, TM's can be viewed as language acceptors. There is a hierarchical classification of TM's that is naturally induced by the "grammatical" properties of the families of accepted languages. We do not address grammar theory, but it is useful to define the hierarchy and summarize the properties of each machine class. Using class name abbreviations, the machine hierarchy is given by the following chain:

$$FA \subset PDA \subset LBA \subset T \qquad \textbf{(2.1)}$$

where T is the class of TM's, *LBA* is the class of linear bounded automata, *PDA* is the class of pushdown automata, and *FA* is the class of finite automata. Each class represents a special kind of TM.

The simplest TM's are the FA's which function as word recognition devices. When given a word, the device scans each symbol sequentially from left to right. This action, along with the associated state changes is represented by a transition function $\Theta_{FA} : Q \times S \rightarrow Q$, where Q and S have the usual meanings. The word is accepted by *FA* when its state is an element of a distinguished subset of Q. The family of languages accepted by class *FA* is denoted by L_3.

A *PDA* scans input words from left to right and has a possibly infinite auxiliary tape called a pushdown store. The stored information must be used in the reverse order to that in which it is inserted (last in first out). Upon scanning a symbol s while in some internal state with p the topmost letter in the store, the *PDA* erases p and inserts a (possibly empty) word $p_1p_2 \ldots p_k$ into the store; goes to another internal state; and either continues to scan s or goes to the next tape symbol. The action of the *PDA* is given by the transition function $\Theta_{PDA} : Q \times (S \cup \{\omega\}) \times P \rightarrow Q \times \Pi$, where Q and S have the usual meanings; P is a finite set of pushdown symbols with p_0 the store bottom symbol; Π is the set of all words on P including the empty word; and ω is a distinguished symbol which allows the *PDA* to act under the influence of the current internal state and stack top without reference to the input tape. An input word is accepted by a *PDA* in the same manner as for *FA*'s. The family of languages accepted by the class *PDA* is denoted by L_2.

A *LBA* can move in both directions when scanning an input word and may replace scanned symbols by another symbol. However, while doing this it may only use as much tape as is occupied by the input word. This action is described by the transition function $\Theta_{LBA} : Q \times S \rightarrow Q \times S \times \{L,R\}$, where all sets have the usual meanings. A word is accepted by *LBA* in the same manner as previous machines. The family of languages accepted by the class *LBA* is denoted by L_1.

It can be shown that there is a language hierarchy associated with (2.1) given by:

$$L_3 \subset L_2 \subset L_1 \subset L_0 \qquad \textbf{(2.2)}$$

where L_0 is the family of languages accepted by class *T*. This inclusion chain clearly delineates the power of each machine class in terms of its language acceptance capability. Thus, for example, *T*'s which are not *LBA*'s are more powerful in this sense than *LBA*'s. We note that the utilization of nondeterminism does not change the overall language acceptance power of class *T* and *FA* machines. However, it does enhance that for the *PDA* class. The effect of nondeterminism upon the power of the *LBA* class machines is unknown.

2.3 Logic Gates and Universality

Although mathematicians prefer to think abstractly about computation as ascribed above, it is sometimes more insightful to think of computation in terms of more physically tangible concepts such as logic gates. Thus, in our discussion of the TM, we assumed certain primitive operations that were fundamental to its operations are logical primitives

that could be assumed (move left or right, read, write, store, erase). While it doesn't, in principle, matter how we implement our calculation on a machine, it is helpful to consider a specific model for building a computing machine. Digital computers are useful for this purpose, even though we could use a number of other physical models, such as hard-sphere gasses, nonlinear dynamic systems, billiard balls, an abacus (moving stones around), etc, (Lloyd 1994). Thus, we can build up to very complicated computations using simple electronic circuits which are designed to do simple binary addition for binary inputs *A* and *B*. In fact, Claude Shannon, of information theory fame, was the first to realize that binary arithmetic could be implemented electronically. His master's degree thesis was devoted to this subject and has proven to be very influential. As an example, let us add two binary digits together with a sum **S** and a carry **C** as shown in Table 2.1. Logical relations **AND, XOR, OR** are subsets of binary addition (Table 2.1). The carry is **AND**, the sum is **XOR**, while the **OR,** is a combination of sum and carry. Because of this equivalence, one can implement a method for doing arithmetic in terms of electronic circuits. Such implementations are called Boolean logic gates or simple logic gates.

A	**B**	**S**	**C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLE 2.1
(BINARY ADDITION)

A	B	**A and B**
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 2.2
(AND)

A	**B**	**A xor B**
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 2.3
(XOR)

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

TABLE 2.4
(OR)

One additional logic gate that is commonly used in discussions of gates is the *not-and* or **NAND** gate which is implemented as complement (negation) of the **AND** gate. Logical operations that can be used to construct all other logical operations are termed *universal.* The **NAND** gate is an example of a gate that is universal.

Ekert and Jozsa (1996) have provided the following useful summary of this classical computation: "*For any input of size* ***n*** *we set up the input in* ***n*** *bits, followed by a string of bits in a standard state* ***0,*** *providing extra working space. Each step of the computation consists of selecting two bits—bits* $(i_k j_k)$ *for the k-th step—and applying a specified* ***Boolean gate B*** *to these bits, which are subsequently replaced in the string. Here a Boolean gate* $\boldsymbol{B}_k$ *is a Boolean operation with two input bits and two output bits. The gates* $\boldsymbol{B}_k$ *are chosen from some finite fixed set of gates. It can be shown that various small finite sets of gates suffice to perform any computation in this way. Thus for any input size* ***n*** *we have a specified sequence of Boolean gate operations (the program), and their concatenated sequential application may be viewed as a network of gates* $\boldsymbol{G}_n$*. The full computation* ***C*** *corresponds to a family of networks* $\boldsymbol{C=\{G_0, G_1, \ldots\}}$*, parameterized by the input size.*" (Note there is an additional regularity condition, (Papadimitriou 1994), needed to avoid uncomputable functions in the structure of $\boldsymbol{G}_n$.) Thus, to transfer any computation into a physics based computational model, it suffices to do two things: transform the digits into the equivalent physical string, billiard balls for example, and then find a physical implementation of the Boolean gates or logic tables (proper geometrical combinations of corners of billiard tables for example). Thus, if nature provided us with a new universal gate, then we would have a new potentially richer means of accomplishing a computation. An example of this is the universal quantum mechanical gates and their physical implementations which have recently been discussed in the literature, i.e. Barenco (1995) and Milburn (1989).

3. Quantum Theory and Quantum Computation

3.0 Church-Turing Principle

A physical system is perfectly simulated by a computing machine if there is an instruction set which renders the machine computationally equivalent to the physical system, e.g. equivalently prepared inputs yield outputs that are statistically indistinguishable. A machine is said to operate by *finite means* if, and only if, during any step in its motion only a finite subsystem of it is involved and the motion depends only

on the state of the finite subsystem and is finitely specifiable in the mathematical sense. Deutsch (1985) has reinterpreted Turing computable functions to be those which can be computed by a real physical systems. Thus, he has taken the mechanistic aspect of CT to mean emulated by a real physical system, i.e. one that obeys the laws of physics. While classical computers clearly operate according to the laws of physics, they can certainly simulate systems that "don't obey the laws of physics". Thus, one must be careful with the language being used. What is meant is the logic gates that comprise the physical components must be based on the laws of physics and specifically those of quantum mechanics. (If the gates were biological, such as DNA for example, a non-reductionist would claim the gates were not based on physics, since it has never been proved that biology reduces to physics.) With these assumptions, one might conclude that it is useful when searching for new computational paradigms to view computer science as a branch of physics.

Deutsch combined these ideas about computer science with the concept of "perfect simulation" and "finite means", and replaced the CT with one based on physics (denoted DCT): *every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.* While both the universal Turing machine and the quantum computer, which we discuss shortly, operate by finite means, Deutsch argued that DCT is not satisfied by the universal Turing machine in classical physics. He argued that a universal Turing machine cannot perfectly simulate any classical dynamic system because there are an uncountable number of continuum input states for the latter. We believe that this argument is false, though it does not effect the design of quantum computers and subsequent discussions about them. The reason for this comes from communication and information theory. The sampling theorem shows that any physical system can be simulated digitally provided it has finite bandwidth. Since only mathematical fictions have infinite bandwidth, one can always simulate almost all systems with the correct application of the sampling theorem. Thus, we prefer to view DCT as a useful fiction to guide the search for new models of computation principles, e.g. a useful principle of ontology, rather as a principle of nature, e.g. a principle of epistemology as Deutsch does.

It is useful to note that Deutsch is not the only one to argue for a physical interpretation of CT. Another physicist, Wolfram (1985), has observed
"There is a close correspondence between physical processes and computations. On the one hand, theoretical models describe physical processes by computations that transform initial data according to algorithms representing physical laws. And on the other hand, computers themselves are physical systems, obeying physical laws."
One might argue that he has formulated a more general form of a physical CT:
"*...One expects in fact that universal computers are as powerful in their computational capabilities as any physically realizable system can be, so that they can simulate any physical system. This is the case if in all physical systems there is a finite density of information, which can be transmitted at only at a finite rate in a finite-dimensional space. No physically implementable procedure could then short cut a computationally irreducible process."*

We term this the Wolfram CT (WCT). It is amusing to note that Wolfram anticipates some of the subsequent claims of speedup of quantum computing by noting that the parallelism of the path integral approach to quantum mechanical systems will allow intractable problems to be solved in polynomial time.

3.1 Quantum Weirdness and the Principle of Superposition

The mathematical formalism for quantum mechanics was originally developed by von Neumann. Here we provide a non-rigorous summary of the axioms of this formal system which are accepted as the basis of quantum theory: **AXIOM 1.** Every physical system can be represented by a Hilbert space Σ and the state of the system described by a vector $|\Psi\rangle$ of Σ. **AXIOM 2.** To every physical observable there corresponds a Hermitian operator on Hilbert space. **AXIOM 3.** The only permitted physical results of measurements of an observable A are elements of the spectrum of the operator $\hat{A}$ which corresponds to $\hat{A}$. **AXIOM 4.** If a measurement of an observable $\hat{A}$ is made upon a system in the state $|\Psi\rangle$, then the probability of obtaining the measurement value α of A is $|\langle\alpha|\Psi\rangle|^2$, where $|\alpha\rangle$ is an eigenvector of $\hat{A}$ corresponding to the eigenvalue α and $\langle\alpha|\Psi\rangle$ is the scalar product of $|\alpha\rangle$ and $|\Psi\rangle$. **AXIOM 5.** There exists for every system an Hermitian operator $\hat{H}$ (the energy or Hamiltonian operator) which determines the time development of $|\Psi(t)\rangle$ through the time-dependent Schrödinger equation:

$$\hat{H}|\Psi(t)\rangle = i\hbar\frac{d}{dt}|\Psi(t)\rangle, \tag{3.1}$$

provided the system is not disturbed. **AXIOM 6.** Immediately following a measurement of the value of α of an observable A, the system is in the eigenstate $|\alpha\rangle$, corresponding to the eigenvalue α.

It should be noted that other axiomatizations exist which include postulates that address subsystems, spin, continuous spectra, state degeneracy and commuting observables. A consequence of the first axiom is the superposition principle, i.e. the states of a physical system obey the principle of *quantum linear superposition.* There is no classical equivalent of this property. In addition, when combined with the other axioms, this axiom provides for the on-classical quantum interference effects. The second and third axioms mean that a measurement must lead to a real measurement value. From the fourth axiom, it can be seen that the result of an observation is in general unpredictable before the observation is performed. This is true even when the state is known, unless, of course, it is an eigenstate of the observable. Axiom 5 means that the dynamical evolution of an undisturbed (e.g. by measurement) system is a causal, deterministic, and reversible process. The final axiom is often referred to as the "projection postulate", since it implies that the pre-measurement of state is "projected onto" the post-measurement state. This discontinuous, irreversible transition cannot be explained by Schrödinger's equation and is aptly called the " collapse of the wavefunction" .

The principle of superposition is the bedrock of much of classical and modern physics. It is the heart of the *quantum weirdness* that is the central source of the non-classical nature of quantum mechanics. Classical superposition in mechanics is just the realization that combination of forces is linear, e.g. when a new particle is added to a system of particles the system can be treated as one force acting on the particle rather than treating the effect of each individual. Classical waves obey a similar rule for combination. Arbitrary combinations of solutions can be combined to give another solution to the wave equation. Uniqueness of solutions occurs when boundary conditions are imposed. Wave superposition superficially resembles quantum mechanics. The following explanation is clearest from Dirac (Dirac 58) and further illuminates the previous axiom set: *"...The non-classical nature of the superposition process is brought out clearly if we consider the superposition of two states **A** and **B**, such that there exists an observation which, when made on the system in state **A**, is certain to lead to some particular result, **a** say, and when made on the system in state **B** is certain to lead to some different result, **b** say. What will be the result of the observation when made on the system in the superposed state? The answer is that the result will be sometimes **a** and sometimes **b**, according to a probability law depending on the relative weights of **A** and **B** in the superposition process. It will never be different from both **a** and **b**. The intermediate character of the state formed by superposition thus expresses itself through the probability of a particular result for an observation being intermediate between the corresponding probabilities for the original states, not through the result itself intermediate between the corresponding results for the original states."*

To illustrate the superposition principle in its full weirdness, consider the famous feline murder scheme of Schrödinger. Obtain any quantum mechanical system that can be in one of two states, say $|excited\rangle$ or $|ground\rangle$ and map the state of health of the cat into these two states as follows: place the cat, preferably white, into an opaque enclosure with a flask filled with poison; attach a hammer to the side of the flask with a trigger that is set to activate at the end of an hour, depending on whether the quantum mechanical system is an excited or ground state; using a radioactive atom with a half-life of one hour for definiteness; wait one hour; and then seal the poison from enclosure with the cat. According to the observer, prior to any measurement of the flask at this time, at the end of the hour, the cat is in a state that is the combination of the two states, namely $|health\rangle = c_1|excited\rangle + c_2|ground\rangle$, where the c's are complex numbers and $\|c_1 + c_2\| = 1$. Thus, the cat is in a state that is neither alive nor dead but is in a mixture of these two states. The cat, by the act of observation, is forced into a definite state, namely dead, if the flask is broken, or alive otherwise. (It is not entirely clear what this combination of states really means.) Quantum two state systems, such as spin-up (labeled $|1\rangle$) and spin-down (labeled $|0\rangle$) states for an electron, are fundamental to quantum computers. Many examples of quantum two-state systems are known in physics and any of these can potentially be used to implement a *quantum bit or qubit.* Unlike classical bits which exist in either $|1\rangle$ or $|0\rangle$ state, qubits can exist as superposition of both, i.e. $c_0|0\rangle + c_1|1\rangle$.

3.2 Quantum Turing Machines

There has been an explosive growth in interest in quantum computing since the early work by Feynman and Deutsch (e.g. see the surveys by DiVincenzo (1995) and Ekert (1996)). We will concentrate here on Deutsch, because of his emphasis on physical principles. Deutsch's work is important for the following reasons: (i) it provides the underlying theory necessary to study whether computing machines capable of harnessing quantum mechanical effects can solve problems more efficiently than classical machines, and (ii) it provides, via the DCT, a partial equivalence between physics and computer science. These ideas are beginning to have a significant influence upon the physics and computer science communities.

The architecture of a Deutsch quantum computer (denoted DM) abstractly resembles that of a TM. A DM is a normalized vector in the Hilbert space $\mathfrak{h}$ spanned by eigenvectors $|x;\overline{n};\overline{m}\rangle = |x : n_0, n_1, ...n_{M-1}; m_{-1}, m_0, m_1, ...\rangle$ of the observables $\hat{x}, \hat{n} = \{\hat{n}_i : i \in Z_M\}$, and $\hat{m} = \{\hat{m}_i : i \in Z\}$, where Z is the set of integers, $Z_M = \{0, 1, ..., M-1\}$; and $n_i, m_i \in \{0,1\}$. Here $|\overline{n}\rangle$ encodes the internal state of the DM, $|\overline{m}\rangle$ serves as an infinitely long tape with input, and $|x\rangle$ corresponds to the tape location being currently scanned. The dynamics of a DM are produced by a constant unitary operator $\hat{U}$ which specifies the evolution of any state in $\mathfrak{h}$ during a single computation step of duration Δt. Thus, $|\Psi(k\Delta t)\rangle = \hat{U}^k|\Psi(0)\rangle$, where k is a non-negative integer, $|\Psi(0)\rangle = \sum a_{\overline{m}}|0;\overline{0};\overline{m}\rangle$ with a finite number of the $a_{\overline{m}} \neq 0$, and $\sum|a_{\overline{m}}|^2 = 1$. Here $|\overline{0}\rangle = |0,...,0\rangle|\overline{m}\rangle$ contains the input, and the $a_{\overline{m}}$ vanishes when an infinite number of $m_i = 1 in \overline{m}$. The matrix elements of $\hat{U}$ are constrained to the form:

$$\langle x';\overline{n}';m'|\hat{U}|x;\overline{n};\overline{m}\rangle = \left[\delta_{x',x+1}\hat{U}^+(\overline{n}', m'_x|\,\overline{n}, m_x) + \delta_{x',x-1}\hat{U}^-(\overline{n}', m'_x|\,\overline{n}, m_x)\right]\Delta \qquad \textbf{(3.2)}$$

where $\delta_{x',x\pm1}$ ensures a unit change in tape position and $\Delta = \prod_{y\neq x}\delta_{m_y',m_y}$ constrains memory involvement to location x during a computational step. The operators $\hat{U}^{\pm}$ are arbitrary functions which are consistent with the unitarity of $\hat{U}$ and describe the dynamical motion. There is a DM for each permitted choice of $\hat{U}^{\pm}$ and each exists, i.e. there is a quantum computation for a given input $|\overline{m}\rangle$, if its run time expectation value is finite. The observable $\hat{n}_0$ can serve as a completion flag which can be set internally to unity if the associated DM exists.

The operators $\hat{U}^{\pm}$'s for the DM's that are equivalent to reversible TM's are given as

$$\hat{U}^{\pm}(\overline{n}', \overline{m}'|\,\overline{n}, m) = \tfrac{1}{2}\delta_{\upsilon,\overline{n}'}\delta_{\mu,m'}\,[1 \pm \gamma], \qquad \textbf{(3.3)}$$

where the functions $\nu:(\overline{n},m) \mapsto n''$, $\mu:(\overline{n},m)\mapsto m''$, *and* $\gamma:(\overline{n},m) \mapsto \pm 1$. (These are essentially composite TM transition function projection maps.) Clearly, there is a DM that is equivalent to the UTM.

The universal quantum computer (denoted UDM) not only subsumes the properties of the UTM, but also simulates with arbitrary precision any quantum computer by permitting the utilization of eight distinguished instruction sets which provide the following four unitary transformations and their inverses for the evolution of single computational binary basis states, i.e. $|1\rangle$ and $|0\rangle$ into linear superposition, qubits:

$$\begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix};\begin{bmatrix} \cos\vartheta & i\sin\vartheta \\ i\sin\vartheta & \cos\vartheta \end{bmatrix};\begin{bmatrix} e^{i\vartheta} & 0 \\ 0 & 1 \end{bmatrix};\begin{bmatrix} 1 & 0 \\ 0 & e^{i\vartheta} \end{bmatrix} \qquad \textbf{(3.4)}$$

Here φ is some irrational multiple of π. These transformations generate under the operation of composition a group G' that is dense in the group G of all unitary transformations on the Hilbert space spanned by $\{|0\rangle,|1\rangle\}$, i.e. $G'\subset G$ and every open subset of G contains elements of G'. Thus, desired transformations of individually specified binary states can be produced with arbitrary precision via generator composition using catenations of these distinguished instruction sets. Indeed, there are instruction sets which induce analogous evolutions for finite numbers of such states.

3.4 Computation Using Quantum Parallelism (Shor's Algorithm)

While the class of functions (on Z) computable by the UDM is the same as that computable by the UTM, the UDM employs the quantum mechanical property of state superposition to provide massive parallel processing capabilities. In particular, the UDM can in a single execution compute N valuations of a function f. This is done via the required unitary evolution of an initial state prepared as a superposition of N basis states with domain values u_i encoded in each eigenvector $|\overline{m}\rangle_i$, $i=1,2,...,N$. Upon completion, the resulting state is a superposition of N basis states with the associated image values $f(u_i)$ encoded in each $|\overline{m}\rangle_i$. Unfortunately, the peculiarities of quantum measurement, i.e. "collapse of the wavefunction", allow a "readout" of only one of the values $f(u_i)$. Thus, the process must be executed many times in order to provide a complete readout and the mean time required to measure all of the values $f(u_i)$ computed in this manner is at least as long as that needed to compute them sequentially. However, quantum mechanical interference effects can, in principle, be used to reinforce results of interest and cancel results that are not of interest.

Shor (1994) has employed the properties of massive quantum parallelism and interference phenomena associated with the quantum mechanical superposition of large numbers of states to develop algorithms which will allow quantum computers to efficiently perform prime factorizations of integers. Shor's work has provided the first real indication of the intrinsic computational power of quantum computers. His algorithm will factor integers on a quantum computer in polynomial time. This result has created a great deal of

excitement within the computer science community, since heretofore, no polynomial time factoring algorithm was known. Indeed, the factoring problem was believed to be so difficult that encryption systems are based on it. Thus, if a quantum computer could be built, codes that require months or years to crack using contemporary classical computers could be broken in seconds using a quantum factoring device. More recently, Grover (1996) has harnessed the power of quantum parallelism and interference to provide an algorithm to solve the data base search problem, to find a specific record in an unsorted list, in a manner that is significantly more efficient than classical approaches.

4. Towards a Generalized Theory of Computation

4.0 Gödel's Principle

It is instructive to return to a paper written by Gödel (1936) shortly after Turing's paper on computability appeared. In this paper, he considered the question of what was the length of a proof in a formal system *S*. He noted that those functions that were recursively defined were not necessarily provable in a system of logic S_i , where i denotes the system of axioms one is using, whereas going from i to i+1 indicates going to a system with additional axioms, but could be provable in a system of logic S_{i+1}. He observed: *Thus, passing to the logic of the next higher order has the effect, not only of making provable certain propositions that were not provable before, but also of making it possible to shorten, by an extraordinary amount, infinitely many of the proofs already available… It can, moreover, be shown that a function computable in one of the systems* S_i, *or even in a system of transfinite order, is computable already in* S_1. *Thus the notion of 'computable' is in a certain sense 'absolute', while almost all meta-mathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend upon the system adopted.* We thus have an interesting dichotomy between proof and computability; there are levels of proof depending on what system of logic one is using but there is only one level of computation. **A function is computable or it is not. There is no middle ground.** Thus we have a principle that should be ingrained in our system of knowledge..

A reading of Gödel suggests is that if a problem is computable, it may be worthwhile looking for a speed-up approach to the problem if the computational time bound exhibits significant nonlinear growth as a function of problem size *n*. One could attempt to reverse engineer the time complexity of the problem via the application of the appropriate logic gate(s) that absorb some, preferably all of the problem complexity. An example that illustrates this point is the binary search problem which is exponentially time bounded (is of the order 2^n). Some of this problem's complexity could be offset by using a massive quantum superposition of states so that the superposition effectively prunes the individual search paths to produce a polynomial time bound for the problem.
There are two essential features of computability that need to be considered as fundamental to new understanding of computation models: speed and computability. There are problems, the so called intractable class, which have algorithmic solutions

which have time bounds that are inherently exponential with problem size *n*. These seem ripe for application of a speedup theorem. Another aspect is whether Turing's notion of computing is sufficiently broad to encompass all functions which are computable in the intuitive sense of the word (effectively computable). If we accept CT, then the Turing computability is universal and any new form of computability must reduce to it. In order to demonstrate a more general form of computability, it would suffice to demonstrate a solution to the classical halting problem (to decide a priori whether a TM exists. Since the halting problem is essentially equivalent to the Cantor's proof that the real numbers are uncountable, this demonstration seems unlikely. (It might be interesting to consider the question of proof of the halting problem relative to some conceptual basis of operation such as relative to the rational numbers, irrational numbers, or the real numbers, for example.)

4.1 Computational Tools Drawn from Nature

At this point, we adopt the WCT rather than DCT and proceed with a discussion of how models taken from nature may aid the search for new models of elements that might be used to perform a computation. We choose to view the DCT as a broader inclusive physical principle than the restrictive metaphysical interpretation of Deutsch. The external world provides a number of mechanisms that are not really modeled in our current conceptions of computing. The philosophical status of these has not truly entered into the phenomenological literature, but it is clear that four of the five mechanisms are clearly attributes of the external world. The phenomena we wish to discuss in relation to sources of potential deeper understanding of computation are:

- true randomness: (computational speedup) Randomness as a phenomenon of nature versus probability-which is an artifact of human reasoning in the absence of complete knowledge. It provides a potential mechanism for the speedup of computations.
- superposition: (new logic gates based on physical law) Fundamental forces for particles combine in a manner so that they obey a law of composition or superposition such as occurs when multiple objects are brought together.
- occupancy: (new types of memory architecture) The statistics of how quantum mechanical particles (fermions, bosons) behave indicates that there are potentially multiple models of how one can store bits
- locality: (potential solution to halting problem) Would time travel effect the ability to compute in a manner that allows one to get around the halting problem?
- genetic evolution: (encoding randomness to increase predictive ability) Can correct predictions, such as the tossing of a coin, be mapped into the computation process to aid computation?

Each of these phenomena deserve a fuller treatment than we provide here. We will outline what we mean by these in further detail in the next sections.

4.3 True Randomness

There are some who take probability as an attribute of the natural world and consider it a "measure of randomness" that applies to non-repeatable events or experiments. Thus, a probability is a measure of our state of knowledge about a set of events, rather than a number with some "physical meaning". Physical meaning is possible only when a frequency of occurrence of a countable set of outcomes can be determined so a measure of occurrence of the number of expected events for a specific number of events can be determined. Probability is the calculus of reasoning in the absence of certain knowledge about a subject, hence it is subjective knowledge. It becomes objective only when subjective knowledge has been reduced to certainty. Thus, probability reflects our state of knowledge about events or objects.

That said, randomness is a different matter. Randomness is an attribute of a partially understood physical system that reflects unmodeled initial or boundary conditions. Thus, the answer to the question (Wolfram 1985) "How is apparent randomness produced?" is always conditional. Randomness is always defined with respect to degree of complexity of the procedure used to produce it. Thus, it is meaningless to discuss randomness, without conditioning it with respect to the measure one used to characterize it. **Randomness is in some sense formally equivalent to proof** (this observation encapsulates the contents of Chaitin's body of work (Chaitin (1989))) in mathematics, thus it is in a sense either **trivial** (see the article by da Costa in (Casti 1996)) or a **measure of the information content of the axiomatic system being used**. (Long proofs are an indication of low information content while short proofs are an indicator of high information content, hence triviality.)

Randomness can be used to drive the behavior toward a solution when we don't know *a priori* what the solution should be. It therefore serves the useful function of preventing the biasing of our computations when we are dealing with a situation where we don't have detailed prior knowledge about the systems phase space portrait (complex dynamics characteristics). Following the analogy between proof and randomness, it can be argued that computationally complex randomness can be used to help tackle computation problems that have solutions with "high information content" with respect to that measure. Thus, a properly designed computer experiment with a physically realistic model of the randomness, can be used to significantly reduce the computation load necessary to do a detailed computation or simulation.

4.4 Superposition or Composition as a Basics for Computation

The principle of linear superposition plays a major role in the analysis of wave fields. We are accustomed to its failure when we analyze finite amplitude waves and in the high field case, so it is not sacrosanct. If we know that the rule for combining subsystems into the whole is nonlinear, the principle of generalized superposition allows us to look for a rule that gives the appearance of a rule of composition that can be used to create logic gates which are the basis for building a universal computing machine. Consider a system

with inputs $x_i(n)$, scalars c_i and a system transform T. This system will be linear provided the system obeys the principle of superposition

$T[x_1(n)+x_2(n)]=T[x_1(n)]+T[x_2(n)]$ and scalar multiplication $T[cx(n)]=cT[x(n)]$.

Linear systems are generalized by denoting the rule for combining inputs by $\otimes$, and the rule for combining outputs by $\oplus$. We can then write the generalized superposition principle for the system as (Gray 1996)

$$H[x_1(n)\otimes x_2(n)]=H[x_1(n)]\oplus H[x_2(n)]. \quad \textbf{(4.1)}$$

A similar condition can be defined for scalar multiplication by a constant c:,

$$H[c:x(n)]=c:H[x(n)] \quad \textbf{(4.2)}$$

Systems that have inputs and outputs that satisfy both (1) and (2) are referred to as homomorphic systems since they can be represented as algebraically linear (homomorphic) mappings between the input and output signal spaces. Note that the formalized view of homomorphic transformations can be relevant to a number of other physical systems, such as quantum mechanical systems. There are other likely physical system in nature that can be used to construct logic gates.

4.5 Occupancy

The standard memory architecture of a Turing machine is simple: either a tape position is occupied or it is isn't. This is based on the principle that two objects can't occupy the same place at the same time. While this is an appropriate model for classical Newtonian physics, it doesn't capture what happens in either wave physics, quantum mechanics, or particle physics. Solitons, for example, can pass through each other without interacting. If a soliton wave were used as a memory device, it might be possible for multiple units of information to be stored at the same location. Thus, it would be possible to obtain a speedup by using higher order arithmetic in the elementary logic units of the associated computing machine. The statistics of how quantum mechanical particles (fermions, bosons) behave suggest that there may be multiple approaches to information storage. Fermions obey Fermi-Dirac statistics (no two particles with the same spin can occupy the same spin state, so one spin state can model "1" while the other models "0"), and can be mapped into binary arithmetic systems that produces the standard Turing model of memory. An unlimited number of bosons can occupy the same state without interacting. Thus, if there were a means to accomplish arithmetic without requiring uniqueness of the bits, a vast parallelism of memory may be possible which would increases computation speed. Particle physics offers even stranger possibilities. Fractional statistics are postulated for quarks, which lead to strange occupation characteristics that could potentially offer very interesting models of memory.

4.6 Locality

All computations are assumed to be **local** in the following sense: there is a connectivity between m^{th} step and the $(m + 1)^{th}$ step of a computation which could be viewed as tantamount to assuming an arrow of time. Moreover, the modeling of time is immutable,

resembling a Newtonian model of absolute time. It is known that the current theory of gravity favors a model of time that exhibits spatio-temporal characteristics (general relativity which is geometric in nature). Another one of the curious aspects of Gödel's career is that he studied general relativity because of close contact with Einstein. He arrived at the rather remarkable conclusion that in a rotating universe, time travel was possible provided *closed causal loops* were formed between the past and the future. Essentially, these are four-dimensional touri in the structure of space-time. Thus, a non-paradoxical means of the future communicating with the past is possible. If this type of geometry could be mapped into a model of computing it may offer a way of bypassing the paradox associated with the halting problem. In principle, a more general theory of computing would then be possible. Continuous geometry offers a possible way (Melzak 1976) around the obstacle that the halting problem places on computing. Geometrical considerations seem to be the most likely source of truly new theories of computing rather than just speedup potential.

4.7 Genetic Evolution

Some aspects of computation resemble trying to predict the outcome of a random event rather than a deterministic computation. An example would be trying to match a series of coin tosses or some other simple binary event such as survival of an encounter with a predator. Evolution seems to deal with this computation problem by the usage of the genetic code to encode random predictive ability to increase survival skills. Since prediction can be viewed as a class of computational problems, it is possible to use this mechanism for some types of computation. The genetic code can be shown to be a universal computing machine and offers speedup potential (see Adelman (1994)). Thus it provides a potential speed up mechanism, though there are significant difficulties with encoding and decoding.

5.0 Conclusions

By assuming a physical interpretation of CT, e.g. WCT, rather than the DCT, it is possible to consider a wider range of physical mechanisms that might be employed to enhance computational power. As a result, many new insights can possibly be drawn from nature to help improve or subsume standard approaches to computation and computability.

REFERENCES

Leonard M. Adleman, (1994), "Molecular Computation of Solutions to Combinatorial Problems", *Science*, Nov. 11, Vol. 206, page 1021.

A. Baremco, (1995), "A Universal two-bit gate for quantum computation", *Proc. R. Soc. Lond. A.*, Vol. 440, 679.

John L. Casti and Anders Karlqvist, (1996), *Boundaries and Barriers: On the Limits to Scientific Knowledge*, Addison Wesley.

Gregory J. Chaitin, (1987), *Algorithmic Information Theory,* Cambridge Tracts in Theoretical Computer Science 1.

David Deutsch, (1985*), Proc. R. Soc. London Ser.* A, Vol. 400,

Paul Dirac, (1958), *Principles of Quantum Mechanics*, Oxford University Press.

David P. DiVincenzo, "Quantum Computing", Science, Vol. 270, 13 October, 1995, pp. 255-261.

Artur Ekert and Richard Jozsa, (1996), "Quantum Computation and Shor's factoring algorithm", *Reviews of Modern Physics*, Vol. 68, No. 3, July 1996.

Richard P. Feynman, (1985), "Quantum Mechanical Computers", *Optics News*, February.

Richard P. Feynman, (1996), *Feynman Lectures on Computation*, Addison Wesley Press.

Kurt Gödel, (1936), "On the Length of Proofs", in Kurt Gödel Collected Works, Volume 1, 1929-1936, edited by Solomon Fefferman, 1986.

John E. Gray, (*1996*), "Emergence, A Statistical Mechanics Viewpoint", in *Actes du Symposium ECHO, eds. Andree Ehresmann, G. L. Farre, J. Vanbremeersch, Amiens (France), 21-23 Auot.*

L. K. Grover, (1996), in Proceedings of the 28th Annual ACM Symposium on Theory of Complexity, ACM Press, New York, pp. 212-219.

J. E. Hopcroft and J. D. Ullman, (1979), *Introduction to Automata Theory, Languages and Computation*, Addison Wesley, Reading MA.

Seth Llyod, (1994), "Necessary and Sufficient Conditions for Quantum Computation", *Journal of Modern Optics*, Vol. 41, No. 12, 2503-2520.

Seth Llyod, (1995), "Almost Any Quantum Logic Gate is Universal", *Physical Review Letters*, 10 July, Vol. 75, No. 2, pp 346-349.

Z. A. Melzak, (1976), *Mathematical Ideas, Modeling and Applications: Volume II of Companion to Concrete Mathematics*, John Wiley and Sons, New York.

A. J. Milburn, (1989), "Quantum Mechanical Fredkin Gate", *Phy. Rev. Lett.*, Vol., 62, 2124.

C. H. Papdimitriou, (1994), *Computational Complexity*, Addison-Wesley, Reading MA.

W. Shor, (1994), "Algorithms for Quantum Computation: Discrete Log and Factoring", in of *the Proceedings 35th Annual Symposium on the Foundations of Computer Science*", IEEE Computer Society, Los Alamitos, CA.

Thomas A. Sudkamp, (1988), *Languages and Machines*, Addison Wesley Publishing Company, Reading MA.

Stephen Wolfram (1985), "Undecidability and Intractability in Theoretical Physics", *Physical Review Letters*, Vol. 54, No. 8, pp. 735-738.