A Painless Survey of Quantum Computation

Hersh Toibman

December 22, 2004

Abstract

This paper is a survey of advances in quantum computation from around 1999-2005, with discussion of questions and issues surrounding these topics and quantum computation as a whole.

1 Introduction

It is traditional to begin papers about quantum computation with a few pages describing what it is, a little perspective, and its amazing capabilities and potential. I do not claim to be any sort of authority at all on any of these topics, so I cannot possibly indulge in such pomposity; as someone who is almost entirely self-taught in the subject, I also know that these introductions are generally not very helpful, generally because they are intended as filler material, while the main body of the paper focuses on one specific topic. However, there are several papers and books that are explicitly intended to be introductory, and some are very useful. Such papers include [13], [14] and [15]. Other introductions are: [16], [17], [18], and [19]. In order to understand anything at all about quantum computing, it is necessary to have at least a nodding, if unfriendly, acquaintance with two subjects: complex algebra and quantum mechanics. There are many introductions to both these subjects, in varying degrees of readability and mathematicality. A very readable introduction to quantum mechanics is [5]; others include [6], [7], [8], and [9]. For complex algebra, I have found [1], [2], [3], and [4] very useful.

I have tried to write this paper as simply as possible; I don't see a point in making it a torturous read. For example, I used colloquial language, and have inserted sprinkles of cynicism whenever it struck my fancy. I have also used a minimum of math, and have presented a brief introduction to matrix arithmetic to help the reader understand the matrices used. I have tried to explain as much background material as necessary for understanding what comes next. I therefore hope that this paper makes for pleasant reading.

The aim of this paper is to provide a compact overview of progress, ideas, and issues in quantum computing that have cropped up in the past few years. Until about seven years ago, many surveys and introductions to this field were written . Then, almost inexplicably, the number of surveys dwindled, so one would be hard-pressed to find something written after 2000. One reason for this seems to be an apparent lack of theoretical progress in the field. Researchers are starting to deal with the practical implications of quantum mechanics and computing: teleporting matter [21], creating physical quantum computers[1] and implementing preexisting algorithms. However,

[1] see section 3 of this paper.

when all is said and done, all the experimenters have to say for themselves is "Hurrah! This is what we did, and this is how we did it." This is important because it concretizes theories, but it is not in of itself inherently *new*. This paper seeks to summarize whatever progress *has* been made in quantum computing for the past several years. It is divided into three sections:

1. Section 2 presents what is possibly the shortest introduction to matrix arithmetic ever written.
2. Section 3 describes progress in algorithms, namely, quantum game theory. A brief introduction to classical game theory is given, and then its quantum applications are discussed.
3. Section 4 discusses schemes for the physical implementation of a quantum computer, with a brief description of the concepts upon which they depend. Strictly speaking, this subject does not belong in this paper, because it involves physics, not computation. However, it is still an important and useful topic.
4. Section 5 discusses the debate surrounding the sort of computers and languages that are most appropriate for simulation of quantum effects.
5. Section 6 is the obligatory conclusion.

2 The Matrix Reloaded

This section is a review of just enough matrix algebra as you will need to follow the mathematics in this paper.

Addition of two matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \pm \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a \pm e & b \pm f \\ c \pm g & d \pm h \end{pmatrix}$$

Multiplying an entire matrix by some number:

$$q \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} qa & qb \\ qc & qd \end{pmatrix}$$

Multiplying two matrices: Two matrices $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$ are multiplied as follows:

1. We take the first row of the first matrix and the first column of the second matrix: $\begin{pmatrix} \mathbf{a} & \mathbf{b} \\ c & d \end{pmatrix} \begin{pmatrix} \mathbf{e} & f \\ \mathbf{g} & h \end{pmatrix}$ The first element of the row is multiplied by the first element of the column. Then we multiply the second element of the row to the second element of the column, and so on until we exhaust the row, column, or ourselves. We add all the solutions together to get the first element of our new matrix: $\begin{pmatrix} ae + bg & \\ & \end{pmatrix}$.
2. We then go on to the second column of the second matrix: $\begin{pmatrix} \mathbf{a} & \mathbf{b} \\ c & d \end{pmatrix} \begin{pmatrix} e & \mathbf{f} \\ g & \mathbf{h} \end{pmatrix}$. We multiply and add as per step 1, giving us: $\begin{pmatrix} ae + bg & af + bh \\ & \end{pmatrix}$.

3. Now that we've finished the top row of the first matrix, go to the second one and repeat the multiply-add routine, starting again from column one. Do this until we finish every row and column. We end up with $\begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$.

Most people start off doing this with both index fingers and a prayer.

The tensor product of two matrices: This is symbolized by $\otimes$.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} e & f \\ g & h \end{pmatrix} & b\begin{pmatrix} e & f \\ g & h \end{pmatrix} \\ c\begin{pmatrix} e & f \\ g & h \end{pmatrix} & d\begin{pmatrix} e & f \\ g & h \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{pmatrix}$$

The matrices we will deal with will be easy to solve, because they primarily have only 2 numbers: 1 and 0.

There are a few matrices that are used all the time; in fact, they have their own name: the Pauli matrices. The ones necessary for our purposes in this paper are:

The identity matrix: $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ For any 2×2 matrix M, $IM = M$. This is useless for multiplication, but helpful with tensor products.

The reversal matrix: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. This flips a matrix around.

The Hadamard matrix: $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The Hadamard matrix is extremely important in quantum computation, as it superposes qubits.

In quantum computation, a qubit has only two basic values: $|1\rangle$ and $|0\rangle$. We represent $|1\rangle$ as: $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|0\rangle$ as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

3 Algorithms

Since the creation of quantum algorithms in the mid-1990's —namely, Shor's and Grover's algorithms and their spawn— there has actually been an extremely disturbing dearth of new algorithms. A quantum algorithm for solving a maze has been suggested [33]; however, it is incomplete, and furthermore, it only proposes to solve complete binary trees, while further exploration suggests that a simpler and more general classical algorithm exists[2]. Much of the other algorithmic progress that has been made concentrates on elaborating the previously existing ones. However, there has been important progress been made in quantum game theory, a related topic.

[2]The classical algorithm is almost trivial, and applies even for incomplete binary trees. We reference the nodes using a two-part address: The first part represents the path from the root to the node, the second part represents the level. If the first half, every 0 represents a left, every 1 represents a right. The number then reads as a set of directions to that node. For example, 10100|00001 would be right-left-right-left-left until the fifth level, while 10100|00100 would be right-left-stop, since the 1 in the right half of the address is in the third place. Using this scheme, we can get from the root to any node in any binary tree in linear time, using only $O(\log_2 n)$ space in memory. In contrast, the suggested quantum algorithm requires $O(n + \log_2 n)$.

In truth, quantum game theory appears at first glance to have nothing to do with algorithms. Game theory is a field that deals with how people act when confronted with problems, while computer science is interested in what is the best solution to problems. This difference disqualifies a great amount of quantum game theory, and we will discuss an example of this right away. On the other hand, there are games (such as the Coin Flip) where game theory asks, "Is it possible to do better if we change our actions?" This sort of games lends itself nicely to quantum algorithms.

3.1 Quantum Game Theory

Game theory is the study of the interaction between parties in situations where they have conflicting interests. A simple example of a game is chess, where each player wants to capture the other's king. The example of chess will serve quite well to illustrate many important concepts of game theory (we'll get quantum later).

3.1.1 Classical Game Theory in a Nutshell

A game such as chess, where every win is balanced by a loss, is called a *zero-sum game*[3]. Chess also has a limited number of ways to play, and so it is *finite*[4][30]. Each player has as much information as he, she, or it needs. There will be no sudden changes in the rules, which makes it a game of *perfect information*. Chess players also have their own little tricks and favorite moves, called *strategies*. The goal of game theory, then, is the analysis of these games and strategies.

An important, albeit imaginary, concept in game theory is that of *utility*. For example, if Alice is hungry, she may have a choice of apples, oranges, or bananas. She can assign a number to the amount of desirability that these choices have. For example, if she doesn't like bananas, she can give them a utility of -1. Apples can be a 1, while oranges will be a 2^5. Different strategies generally lead to different outcomes with different utilities. In this way, it can be said that the object of game theory is an increase in utilities.

With these concepts, we can now discuss an important game.

3.2 The Prisoner's Dilemmas

3.2.1 The Classical Prisoner's Dilemma

The ever-popular Prisoner's Dilemma is generally used to illustrate the utilities of strategies. In this game, we have two prisoners, Alice and Bob[6], and we want to convict one of them. We offer each certain benefits if they whistle on the other. They have a choice: they can either speak now or forever hold their peace. The utility of incentives is represented by the chart on the next page.

[3] Eli Ehrenfeld has pointed out that perhaps chess cannot be considered zero-sum, as it is possible to draw. Due to time constraints, I was unable to explore the issue fully, but it seems reasonable to say that in a draw, both players half-win.

[4] Okay, so there are about 10^{120} possible ways to play[29], but it's still *finite*.

[5] This scheme is recognized as ridiculous due to its subjectiveness, but it's tolerated because it's important.

[6] [35] noted that these names have become folkloric, and obviously are used because of their initials. In the Coin Flip game, the names Picard and Q are usually used, though I will not because I am not a Trekkie and I feel the relationship between Alice and Bob requires further exploration. Are they married? Why are they in prison? Will they really rat on each other?

		Bob	
		C	D
Alice	C	(3,3)	(0,5)
	D	(5,0)	(1,1)

The first number in each doublet is Alice's payoff, the second is Bob's. C stands for "confess", where the player would snitch on the accomplice; D stands for "deny." An important feature of this game is that it is non zero-sum; unlike chess, it is possible for both players to win this game. The goal here is different: Both players want to win as much utility as possible.

Let's pretend that we're Bob. We see this chart and realize something interesting. If we want to gain no matter what Alice does, it would behoove us to deny all charges. If Alice confesses, we gain a utility of five; if she denies, we still gain one. Denial is our *dominant strategy*, where we gain no matter what the other party does.

Now let's pretend we're Alice. It is obvious that our dominant strategy is still denial: if Bob confesses we gain five; if he denies we gain one. It would thus make sense for Alice to deny all charges too.

It would be logical to assume that Alice and Bob will both deny involvement. When we look at the chart to see what happens in this situation, we find that there is a mutual utility of 1. This is called the *Nash equilibrium*, where both players use their dominant strategies. If we were to take a closer look at our chart, though, we would see that while the Nash equilibrium of strategies is for the two to deny the charges, they would both stand to gain more if they both *confessed*. In that scenario, they both would gain as much as possible without hurting the other. This strategy is said to be *Pareto optimal*. This is why this game is called a dilemma: Chances are, both parties will lose due to their desire to gain.

3.2.2 The Quantum Prisoner's Dilemma

In order to solve this dilemma, [22] suggest introducing a new move, called Q, which is in essence a superposition of $|C\rangle$ and $|D\rangle$. Alice and Bob now each receive a qubit. They can put this qubit into state $|C\rangle$, $|D\rangle$, or $|Q\rangle$. A complex equation to determine the payoffs for Q is given; we will simply rely on [36], who presents them in the following format:

		Bob		
		C	D	Q
Alice	C	(3,3)	(0,5)	(1,1)
	D	(5,0)	(1,1)	(0,5)
	Q	(1,1)	(5,0)	(3,3)

The Q strategy works as follows: If one uses Q and the other uses C or D, than both parties receive payoffs as if they had both used the other strategy, D or C. We can understand Q as having the ability to influence the other's decision, a feature suspiciously similar to entanglement. Notice that there is now a new Nash equilibrium. Both parties stand to gain if they use strategy Q. Not only this, but (Q,Q) is also Pareto optimal. Thus, the dilemma is solved.

3.2.3 Some Issues with QPD

We have mentioned before that not all quantum-game strategies are quantum algorithms. In the case of the Quantum Prisoner's Dilemma, the problem is very subtle: While we applied an obviously quantum operation to a problem, it has been argued[36] that the actual solution is not quantum. Rather, the *problem* has been restated in a quantum way. The method of finding the answer, though, remains purely classical. It is not a quantum algorithm at all.

Another, perhaps more crucial problem, lies in the restatement of the game. Our interest in quantum algorithms lies in the improved solution of classical problems. While introducing quantum operations into the game is legal[7], changing the game entirely is not. For example, Shor's algorithm is considered valid because it uses quantum effects to factor numbers, a classical problem. To quote [36]

> Games are defined by their rules, and if you change their rules, you change the game.

Here, we have presented an entirely different game, which incidentally has a Pareto optimal Nash equilibrium. The original classical Prisoner's Dilemma, however, remains a dilemma.

3.3 Heads, I win; Tails, You Lose

With the above discussion of the Quantum Prisoners' Dilemma in mind, we now present the much-aforementioned Coin Flip game.

3.3.1 The Classical Coin Flip Game

Alice and Bob, having successfully escaped prison, are about to embark on yet another adventure in game theory. This time, their game will be very simple: they have a coin that starts off at heads, and they can either flip it or keep it the same. After a prespecified number of turns, the coin is checked. If it is heads, Bob wins. Otherwise, Alice wins.

If the game were to last three turns, we can have a nice little chart of possible game plans, representing flip with F and no flip with I (identity).

	II	IF	FI	FF
I	-1	1	1	-1
F	1	-1	-1	1

The number represents Alice's payoffs; Bob's payoff is the opposite of Alice's.

This game is obviously a two-player, zero sum game. Also, it would be silly to simply do the same thing all the time: Alice would eventually figure out what is going on. Therefore, Bob is better off with a combination of flips and identities[8]. This is a *mixed strategy*. The expected payoff would depend on his flip:identity ratio.

[7] As in fact will be done in the Coin Flip game

[8] As a matrix, $M = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$, p being the probability of a flip ($0 < p < 1$).

3.3.2 Flipping a Quantum Coin

In the quantum Coin Flip game, we replace our classical heads-tails coin with a quantum coin. This coin can be in $|heads\rangle$,$|tails\rangle$, or a superposition thereof ($a|heads\rangle + b|tails\rangle$). We also give one player an unfair advantage over the other: Bob is now allowed to Hadamard the coin. Will this ability enable Bob to improve his outcome?

The answer turns out to be yes. Using his Hadamard, Bob is able to make the coin $\frac{1}{\sqrt{2}}(|heads\rangle + |tails\rangle)$. Alice can opt to keep it the same or flip it, which will create the state $\frac{1}{\sqrt{2}}(|tails\rangle + |heads\rangle)$. In other words, she will have accomplished nothing. When Bob applies another Hadamard, the state will change to $\frac{1}{2}(|heads\rangle + |tails\rangle + |heads\rangle - |tails\rangle) = |heads\rangle$.

This can be made more concrete using matrices. $|heads\rangle$ corresponds to $\begin{pmatrix}1\\0\end{pmatrix}$ and $|tails\rangle$ to $\begin{pmatrix}0\\1\end{pmatrix}$. The game starts off with the coin in state $|heads\rangle$. Applying the Hadamard gate results in:

$$\underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\underbrace{\begin{pmatrix}1\\0\end{pmatrix}}_{heads} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$$

Our coin is in a neat superposition of up or down. Now, if Alice so desires, she can flip this quantum coin by applying the flip matrix:

$$\underbrace{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}_{F}\underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}}_{coin} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$$

Despite her best efforts, the coin remains unchanged.

On his second turn, Bob applies another Hadamard

$$\underbrace{\left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\underbrace{\begin{pmatrix}1\\1\end{pmatrix}}_{coin} = \frac{1}{2}\begin{pmatrix}2\\0\end{pmatrix} = \underbrace{\begin{pmatrix}1\\0\end{pmatrix}}_{heads}$$

Thus, Bob will win every time, and can use his winnings to post bail when he gets caught[9].

---

[9]The same applies if we begin the game with $|tails\rangle$.

Imagine that Alice doesn't flip the qubit on her turn. In this case, the game is as follows:

$$\underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\underbrace{\begin{pmatrix}0\\1\end{pmatrix}}_{tails} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \Longrightarrow \underbrace{\frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\underbrace{\begin{pmatrix}1\\-1\end{pmatrix}}_{coin} = \underbrace{\begin{pmatrix}0\\1\end{pmatrix}}_{tails}$$

If Alice were to *flip* the coin:

$$\underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\underbrace{\begin{pmatrix}0\\1\end{pmatrix}}_{tails} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \Longrightarrow \frac{1}{\sqrt{2}}\underbrace{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}_{F}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\end{pmatrix} \Longrightarrow \underbrace{\frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{H}\begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}0\\-1\end{pmatrix} = -|tails\rangle$$

Because the scalar associated with the coin's state represents the square root of the probability of obtaining that state, $-|tails\rangle = (-1)|tails\rangle$ has the same probability, 1, as $|tails\rangle$. Generally, only the evolution of $|heads\rangle$ is discussed, presumably because the math is easier.

It is interesting that before the second Hadamard is applied to the qubit, had it been observed, there would have been a 50% chance that we would have found tails; yet, after the Hadamard operation is done again, the 50% chance disappears and becomes 0%. The net effect of the Hadamard is the same as adding $\frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ —it contributes a $-\frac{1}{2}$ probability to the coin's coming up $|tails\rangle$. This is a basic example of quantum interference, where probabilities cancel or complement each other[10]. This is one of the weirdest aspects of quantum physics.

3.3.3 Some Issues With The Quantum Coin Flip

The Coin Flip game has not been impervious to criticism. The first question that may be asked is:"By giving Bob a quantum advantage, and using a quantum coin, haven't we changed the rules, like in the Prisoners' Dilemma?" The answer, of course, is yes. The Quantum Coin Flip is not the Classical Coin Flip, but unlike the Quantum Prisoners' Dilemma, it never claimed to be. The question asked in the Coin Flip game is: "What would happen if we change the game?" The fact that the game would not be the same is a given. This is different from the Prisoners' Dilemma, where it was:"If we change some rules, is it possible to do better at this classical game?"

Another criticism cuts into the matter of which solutions may be properly called quantum. [37] cites [38], which claims that the Quantum Coin Flip is not really quantum, for two reasons. First, it leaves out an important quantum effect: entanglement. Entanglement is the cause for the greatest asset of quantum computing: quantum parallelism. [37] demonstrates, though, that while entanglement is all nice and well, it is not strictly necessary for quantum computation. It also appears that non-entanglement is not a disqualification from quantum algorithms, as long as there is *some* improvement due to quantum effects.

Another criticism mentioned by [37] is that the Quantum Coin Flip can be simulated on a classical machine. This is very interesting, because as we will discuss later, there is a strong effort to create a paradigm that can successfully simulate quantum effects. According to the critics of the Coin Flip game, any problems solved on a classical machine would by definition not be considered quantum.

4 Creating Quantum Computers

Realizing that quantum computation can potentially overpower its classical counterpart, researchers have begun frantically searching for ways to create a real-life, bona fide quantum computer. Before doing so, they established certain guidelines, which dictate exactly what the computer will do. A great deal of ideas was plagiarized from classical computer theory. In order to be useful, a computer must be *Turing complete*: it must be capable of some form of input/output, it must be able to do *any* operation we ask of it, and it must have iteration (the ability to repeat an operation a specific number of times). For the second requirement, any operation known to man or machine is broken down to repeated applications of a single *universal operation*. In the case of classical computers, this is the NAND, or "not AND". The third requirement, iteration, can be satisfied with recursion.

In addition to I/O, NAND, and iteration, quantum computers come with their own set of rules, since they are so radically different from classical computers. The most ob-

[10]The Hadamard complements $|heads\rangle$ by changing it from 50% to 100%.

vious of these, of course, is the ability to superpose and entangle the qubits. Altogether, there are five rules that all quantum computers must meet, which are:

1. All quantum computers must have "well characterized" qubits. This means that we must be able to entangle, superpose, measure, manipulate, and reverse the qubits. We need to know the relationship each qubit has with its neighbors. Without this information, we would not know whether our output is legitimate or gibberish.

2. Similarly, we must be able to initialize our qubits to a specific state, such as $a|0\rangle + b|1\rangle$. This ensures that we know where our output is coming from and also is significant in quantum error correction.

3. The qubit must have long *decoherence times*. Unfortunately, Nature has a very short attention span and quantum states do not stay the same. Thus, $|0\rangle$ may very quickly turn into $|1\rangle$ of its own volition. The longer the decoherence time, the more time we have to measure the qubit accurately.

4. We need to be able to implement a universal gate. Although NAND works universally on classical computers, quantum computers are slightly more complicated, because it is important that all operations are *reversible*. In other words, we need to be able to determine from any output what the input that produced it was. The ordinary NAND does not have this happy feature, so in order to make it usable for quantum computers, we must tweak it to output *two* qubits, like this: NAND(*x,y*)→*x,z*. If we know what *x* and *z* are, we can figure out what *y* was, and so NAND becomes reversible. This reversible operation is called the C-NOT gate.

5. Finally, as mentioned before, we need to be able to read input and output[20].

Researchers have turned their attention to two likely candidates for qubit representation: atoms and photons.

4.1 Atomic Quantum Computers (NMR)

Before discussing schemes to create quantum computers from atoms, it is necessary to know what an atom *is*. We all learned in third grade that an atom is comprised of protons and neutrons in a nucleus, and electrons orbit the nucleus like minuscule planets. This is overly simplistic and inaccurate; in reality, the electron forms a sort of cloud around the nucleus. There can also be many electrons "orbiting" a nucleus, some of them sharing the same path. Electrons in the outer orbit are called valence electrons, and may be shared with other atoms. This show of generosity understandably causes the atoms to form a bond, which is called *covalent bonding*. Several such bonded atoms form a *molecule*[11] [23].

Both nuclei and electrons have certain "spins" to their magnetic fields. Thus, an atom can be imagined as a spinning sun, with the earth orbiting and rotating at the same time[26][12]. The nucleus's *spin number*, I, determines how many axes its magnetic field can spin (called *precession* by people who want to sound smart)[13]. An atom that has $I = +\frac{1}{2}$ will have 2 possible spins in a magnetic field: either it will align itself with the field ($I = +\frac{1}{2}$) or it will align itself *against* the field ($I = -\frac{1}{2}$). If we were to

[11] There are other sorts of bonds too, but we are only interested in covalence here.

[12] At the same time, of course, it *can't be* imagined that way. There's quantum for you.

[13] The relation between I and the number of possible spins is given as $2I + 1$.

put multiple nuclei together in another magnetic field —for example, in a molecule —each nucleus would be affected not only by the global field, but also by each others' tiny fields [27]. As you probably can imagine, these characteristics offer plenty of reason to pursue NMR in the hope of finding a quantum computer, and many aspects of manipulating and organizing these computers are discussed in [28].

4.1.1 Semiconductors

A relatively early proposal for the construction of a quantum computer involves silicon semiconductors [24]. The idea behind a semiconductor has to do with a substance's molecular structure: as previously mentioned, an atom is able to bond with a specific number of other atoms, depending on the number of outer electrons it has. Silicon, known to physicists everywhere by its symbol Si, has four outer electrons and is thus able to bond with another four atoms. If we were to bond a silicon atom to another four silicon atoms, and each of those four were bound to another four, etc., we would have Figure 1.

In this setup, the silicon cannot conduct electricity, because all the electrons are bound in place, and in order to be a conductor, there must be some electrons that are free to move around.

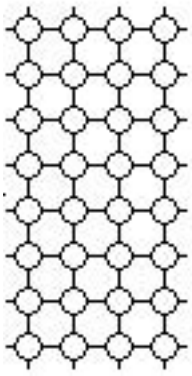


Figure 1:

This problem can be avoided by spiking the Si with another element which does not have four valence electrons. This procedure is known as *doping*[25]. If the element has five valence electrons, as is the case with arsenic or phosphorus, it will bond to four Si atoms and still have one electron left, which can move around the substance and conduct electricity. This is called an *n-type semiconductor*, because the extra electrons contribute a **n**egative charge to the lattice. In this case, the added element is called a *donor*. The same thing can be accomplished by adding atoms with *three* valence electrons[14], in which case there is a hole in the lattice where the atom bonds to only three Si atoms. This hole can be shuffled throughout the lattice, and because the lack of electron contributes a positive charge, it is called a *p-type semiconductor*. Here, the added element is called an *acceptor*.

In the case of n-semiconductors, it is possible for donor atoms to interact with each other via the extra electron. In this case, a qubit would be represented by the donor's spin (either spin-up, which is $I = +\frac{1}{2}$ and represented as $|\uparrow\rangle$; or spin-down, which is $I = -\frac{1}{2}$ and represented as $|\downarrow\rangle$). Two gates are then required: an A gate, which controls the speed of the spins, and a J gate, which controls the electron-mediated interaction between the qubits. The C-NOT would be implemented with two electron-nucleus systems. Input would comprise the manipulation of the A and J gates, along with a globally applied magnetic field, which flips nuclear spins. Output would have to be obtained in two steps: first, nuclear spins would be converted into states with different polarizations[15]. The electron spin would be determined by its wavefunction. Another option is manipulating the electrons to travel to a donor, and exchange energy in such a way that the donor's state can be determined.

A slightly different flavor of this scheme is that of quantum dots. Quantum dots consist of spare electrons trapped within small amounts of semiconductor material, and can be thought of as artificial atoms. In this scheme, a qubit is two dots of different

[14] e.g. boron or gallium

[15] Polarization is the direction that the electron cloud is the most dense. This is where the solar system analogy fails.

sizes. An extra electron is added to do calculations. If this electron is in the larger dot, this is considered the $|1\rangle$ position; if it is in the smaller, it is a $|0\rangle$ position.

Quantum computers have actually been built using another very similar scheme. In [32], the authors describe how they implemented a C-NOT:First, two atoms, a hydrogen for control and a carbon for flipping, were rigged together with a magnetic field. Using a well-aimed radio frequency pulse (called RF),with very exact length and strength, they turned the spin of the "flip" atom a bit downwards. Which direction it spun depended on the influence of the control atom: if the control atom was up, the flip atom's spin would be toward one direction, had it been down, the flip atom's spin would have been in exactly the other direction. Then another RF was applied to the flip atom, this time to rotate it 90 degrees. Depending on where the control qubit's spin was in the beginning, the flip qubit's spin would be either the opposite of what it was or the same. This is a C-NOT.

Another, more recent quantum computer using NMR techniques was created by [31]. In this instance, Shor's Order Finding Algorithm was implemented by a molecule with 5 fluorine spins, the largest quantum computer built so far.

4.2 Photonic Quantum Computing

In addition to atoms, physicists have also recognized photons as being well suited for quantum computation. To understand photons, it is necessary to backtrack a bit and discuss harmonic oscillators.

4.2.1 Harmonic Oscillators

A harmonic oscillator is a "black box," which only allows energy to escape through its opening. Bouncing inside this box is a particle charged with electromagnetic radiation. As it moves about, it releases energy in discrete amounts [16], namely, multiples of $\hbar\omega$, $\hbar$ being $\frac{h}{2\pi}$,[17] and ω being the frequency of the trapped radiation. Each of these quanta of released energy is called a *photon*, and has no charge.

4.2.2 Computers Out of Photons

One scheme for a computer using photons as qubits involves two black boxes sharing one photon. If the photon is in one box, the qubit is said to be in state $|10\rangle$; if it is in the other, it is in state $|01\rangle$. It is possible to imagine a photon moving along in space, in which case its box is imagined to be moving with it. This then satisfies requirement 1 on our to-do list —we have a qubit. Manipulating these qubits requires a few pieces of equipment, which are much easier to obtain than is the case with atomic computers. We need:

Mirrors to direct the paths of the photons.

Phase shifters to refract[18] the photon by a given amount.

Beamsplitters , which are partially silvered mirrors, and reflect some light and send some light off into another direction.

[16] This was discovered by Max Planck in the early 1900's, and was the discovery that started the entire field of quantum mechanics.

[17] h, known as *Planck's constant*, is equal to $6.6260755 \times 10^{-34} J \cdot s$

[18] *Refraction* is when light bends. For example, if you were to put a spoon halfway into a glass of water, it would appear to be bent at the surface of the water.

A *Kerr* device to refract the light in proportion to its intensity. Glass and sugar water are examples of Kerr devices, although the effect is very weak. This has the fabulous ability to cause interaction between two photons, for our entanglement pleasure.

Photons can be created using lasers and can be measured with photodetectors. Thus far, we have satisfied requirements 1,2,3, and 5 from our list. All that remains is for us to find a C-NOT.

As we said above, a photon qubit is represented by $|10\rangle$ or $|01\rangle$. In addition, there are two other states: the photon is in both at once ($|11\rangle$) or the photon is in neither ($|00\rangle$). In order to make a C-NOT, we need two qubits refracted (shifted) by $e^{i\pi}$. We will call this shift K. As a matrix,

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

The Hadamard matrix, $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$, and the identity matrix, $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. The tensor of the Hadamard and identity matrices is:

$$I \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

The matrix product

$$(I \otimes H)(K) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$(I \otimes H)(K)(I \otimes H) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \text{C-NOT}$$

Thus, C-NOT can be implemented using a combination of mirrors and exact phase shifts.

5 Quantum Programming

Thus far, we have discussed two crucial aspects of quantum computation: what we want to accomplish (algorithms) and with what we want to accomplish it (implementations). However, we are still a long way from reaching our goals; as we mentioned before, the largest quantum computer to date is a glorious five qubits large, which is not enough to store the word "Hello." It seems to be the broad consensus that useful quantum computers are not anywhere close to being created. However, quantum computation is a computer science, and as such it is important that we are able to run quantum programs. Without a means to test new ideas, quantum computing remains a theoretical

field, living inside people's heads, occupying e-space in the Los Alamos Archives, but only an occasional tourist in the real world. For this reason, computer scientists are looking for methods to simulate quantum effects on modern, classical computers. It may not be the optimal solution, but it will have to do until a better option presents itself. Of course, as we mentioned before, [38] believes that any computation done on a classical machine is by definition classical. However, this opinion is obviously disputed by many, including [37], [35], [34] and [39]. We will discuss two fundamental, closely-related questions that have yet to be resolved.

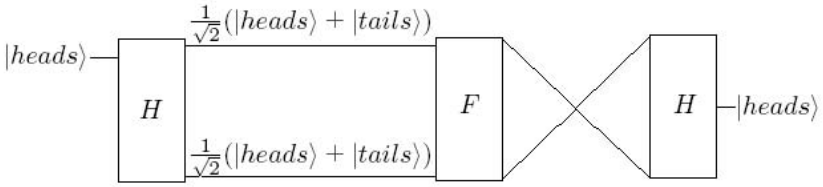
5.1 Analogue Computers vs Digital

The first issue that must be discussed is the sort of computer that will be used for quantum simulation. This question is not new; classical computation suffered the same growing pains when it was first being explored.

Analogue computers are custom-made to solve specific problems. The example given in [15] is that of Gaudí, who solved a complex architectural problem by tying ropes together and hanging them from the ceiling. The computer in this case is the ropes, and their advantage can be readily seen: the problem is solved quickly, easily, and efficiently. As [39] points out, it would have been a terror to program a digital computer to solve this problem. This is the advantage of analogue computers: they are easier to program, and they solve the problem with greater accuracy, whereas digital computers rely on techniques such as floating-point approximation. It would seem at first glance that analogue machines are more suitable for quantum computing. This is because in many cases, they are capable of solving NP-class problems, which is where quantum computers are expected to shine.

However, if analogue computers are so wonderful, why are there very few, if any, still in use? The reason is that analogue computers have one very fatal disadvantage. They can only solve the one problem they were designed for. Mr. Gaudí's ropes will only help him design the one building; next time, he will have to discard them, get new ones, and re-tie them. For analogue computers that are more difficult to build, the time, expense and inconvenience generally make it more feasible to use digital computers, which offer a one-size-fits-all architecture for solving problems.

This problem is readily observed in quantum computation. In almost every paper about quantum computing, a description of the gates used to implement an algorithm is given. For example, a circuit implementing the Quantum Coin Flip Game may appear as:



This circuit cannot then be used to say, sort a laundry list. In order to do that, we'd need to build an entirely different machine. In most papers, where the authors (who generally are physicists) use illustrations to explain a point, they don't care how feasible the implementation of their gates are in real life. However, computer scientists are interested in this issue, and they want a more general architecture, and often find these illustrations unhelpful and annoying. Digital computers have thus become the technology of choice for quantum simulation.

5.2 Quantum Programming Languages

Once we have decided to use digital computers, we are faced with the problem we have avoided thus far: How are we going to make a computer, which is the epitome of classicism[19], behave like a quantum system? How can we simulate quintessentially quantum qualities such as superposition, entanglement, and interference? A few programming languages have been suggested, and we will consider two important examples.

One option for programming quantum simulations is, of course, the imperative paradigm. A language cleverly named Quantum Computing Language —or QCL— has been designed using a C++ interpreter. It has all the familiar components that all programmers have come to know and love: its basic data type is the quantum register `qreg`, which is simply an array. It is used in exactly the same way as in C. For example, `qreg qubit[1];` is a single solitary qubit. Functions are available to perform π-rotations, Hadamard gates, *et cetera*. Modeling the unitary operators that are the bread and butter of quantum physics, these functions are also unitary. The `if`→`then` control sequence is a unitary matrix that has the conditional clause built in. Thus, QCL simply uses a classical method to perform quantum operations [34].

However, some see a deep flaw[39] with this idea. In quantum computing, it is essential that we have a minimum of interaction with the qubits; otherwise, our actions will constitute a measurement and the wavefunction will collapse. In QCL, there is no such collapse of the wavefunction unless we want one. Another, more deeper problem, is the fact that an imperative language is by definition a list of instructions. This implies a deterministic approach to quantum effects, which are inherently probabilistic. A more appealing candidate, then, would seem to be a functional language, which does not so much instruct the computer as describe the results it wants to see.

To this end, a Haskell implementation of quantum systems has been developed[35]. It provides a high level of abstraction, so it is usable for many different quantum systems, rather than obligating the programmer to construct his systems from scratch. As a functional language, it relies on the fact that the programmer is uninterested in how the actual calculations are done, so long as a valid answer is obtained. Thus, the computer is left to its own designs, satisfying the need for there to be no observation during its implementation.

6 Conclusion

From the overall survey provided in this paper, it seems clear that there has not really been much progress over the past few years. Algorithms have not been created; rather, they have been stolen and modified from game theory or other sources. Computers have been created, but while they are tremendous advances in physics, their computational capacity is negligible. The issues discussed in section five are unresolved; there are people who disagree with the conclusions presented.

On the other hand, this lack of progress may not be a bad thing. Quantum computing is maturing very much like a baby: at first, everybody is awed by its novelty, and every new trick —be it the word “Mama” or a factorization algorithm— is greeted with “ooooh”s and “ahhhh”s. Everyone wants to see it, hold it, and send it to museums to improve its IQ[20]. Quantum computing has outgrown that stage; it is now time to enroll our child in nursery school and find out how bright his future really is. While it may

[19] After all, the very idea of a program implies a deterministic approach.

[20] In the case of quantum mechanics, the equivalent of taking it to the museum is [40].

not seem to be growing at present, the groundwork for its future growth is being laid out.

7 Acknowledgements

I would like to thank Professor Murray Gross for his encouragement, help, and threats; Professor Paula Whitlock, for her encouragement, help, and lack of threats. Cameron Zwarich answered the math questions that I asked at the strangest hours, and introduced me to WinEdt (available at `http://www.winedt.com`), which, while it overGUIes LaTeX, is still very helpful and a pleasure to use.

References

[1] Cohen, P. *Elements of Linear Algebra*, Chapman & Hall, 1994

[2] Matthews, K. *Elementary Linear Algebra —Lecture Notes* <http://www.numbertheory.org/book/>

[3] Mathworld, <http://mathworld.wolfram.com>

[4] Wikipedia, <http://www.wikipedia.org>

[5] Styer, D. *The Strange World of Quantum Mechanics*, Cambridge University Press, 2000

[6] Ponomarev, L., translated by Repiev, A. *The Quantum Dice*, Institute of Physics Publishing, 1993

[7] Polkinghorne, J. *The Quantum World*, Longman, 1984

[8] Shankar, R. *Principles of Quantum Mechanics*, Plenum Press, 1994

[9] Gillespie, D. *A Quantum Mechanics Primer*, International Textbook Co, 1970

[10] Spiller, T. *Basic Elements of Quantum Information Technology*, Introduction to Quantum Computation and Information, Chapter 1, World Scientific Press, 1998

[11] Paquin, C. *Computing in the Quantum World* <http://www.iro.umontreal.ca/~paquin/Qu/quantumComp.ps>

[12] Arrighi, P. *Quantum Computation Explained to my Mother*, quant-ph/0305045

[13] Nielson, M., Chuang, I. *Quantum Computation and Quantum Information*,

[14] Reiffel E., Polak, W. *An Introduction to Quantum Computing for Non-Physicists*, quant-ph/9809016

[15] Aharonov, D. *Quantum Computation*, quant-ph/9812037

[16] Deutsch, D., Ekert, A. *Machines, Logic, and Quantum Physics*, math.HO/9911150

[17] Josza, R. *Illustrating the Concept of Quantum Information*, IBM J. RES. & DEV. VOL. 48 NO. 1 JANUARY 2004

[18] Ashok, C. *Introduction to Quantum Computation*, quant-ph/0312111

[19] Kamara, S. *Quantum Information Theory* <http://www.wu-wien.ac.at/usr/h99c/h9951826/quantnotintro.pdf>

[20] DiVincenzo, D. *The Physical Implementation of Quantum Computation*, quant-ph/0002077

[21] Weiss, P. *Teleporting Matter's Traits*, SCIENCE NEWS, 165:387

[22] Eisert, J. *Quantum Games and Quantum Strategies*, PHYSICAL REVIEW LETTERS 83:3077

[23] Hill, J., Kolb, D. *Chemistry for Changing Times*, Prentice Hall (1998) p.63

[24] Kane, B. *A Silicon-Based Nuclear Spin Quantum Computer*, NATURE, VOL 393, 14 MAY 1998, pp.133-137

[25] Cutnell, K., Johnson, K. *Physics*, John Wiley & Sons (1998) pp. 722-723

[26] Tipler, P., Llewllyn, R. *Modern Physics*, W.H Freeman and Company (1999) p. 312

[27] Paudler, W. *Nuclear Magnetic Resonance: General Concepts and Applications*, John Wiley & Sons (1987) pp. 2-6

[28] Vandersypen, L., Chuang, I. *NMR Tecniques for Quantum Control and Computation*, quant-ph/0404064

[29] Luger, G. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Pearson Education Limited, 2002

[30] Davis, N. *Game Theory: A Nontechnical Introduction*, Basic Books, Inc., 1970

[31] Steffen, M., Vandersypen, L., Chuang, I. *Toward Quantum Computation: A Five-Qubit Quantum Processor*, IEEE MICRO, 21(2):24-34, March 2001

[32] Gershenfeld, N., Chuang, I. *Quantum Computing with Molecules*, SCIENTIFIC AMERICAN June 1998 pp.66-71

[33] Bashuk, M. *Solving a Maze with a Quantum Computer*, quant-ph/0304146

[34] Ömer, B. *Classical Concepts in Quantum Computing*, quant-ph/0211100

[35] Karmczmarczuk, J. *Structure and Interpretation of Quantum Mechanics —A Functional Framework*, ACM 1-58113-758-3/03/0008

[36] Van Enk, S., Pike, R. *Classical Rules in Quantum Games*, PHYSICAL REVIEW A 66:024306

[37] Meyer, D. *Quantum Games and Quantum Algorithms*, quant-ph/0004092

[38] Van Enk, S. *Quantum and Classical Game Strategies*, PHYSICAL REVIEW LETTERS 84:789

[39] Gross, M., White paper currently in preperation

[40] Prvanovic, S. *Mona Lisa —Ineffable Smile of Quantum Mechanics*, physics/0302089