Lecture 1: Course overview; bits & pbits

Andrew J. Landahl, alandahl@unm.edu
(**DRAFT**: Thursday 30 August, 2007)

Welcome to UNM Physics 452/581, *Introduction to Quantum Information*. In class, I reviewed the administrative structure of this course, the details of which can be found on the course website at http://info.phys.unm.edu/~alandahl/phys452f07.html

1 Overview

This course will cover four main topics in quantum information, each for about a month apiece:

- Algorithms
- Error correction
- Communication
- Cryptography

One look at this list and I think you'll agree that computer science lingo is setting our agenda! But that's actually a good thing. The pioneers of quantum mechanics never had it so good. They groped and struggled with elusive concepts like "complementarity" and "state collapse" without the benefit of well-developed information theory. Now, nearly 100 years later, we can look back and recast quantum mechanics in a way that makes more sense in terms of these concepts. This, in essence, is the *information physics revolution*: recasting much (some hope all) of physics in information-theoretic language to give new insights.

Information physics is an active research area spanning many physics subdisciplines. Some of the most exciting questions in science lie at the physics-information interface. For example, many of you are probably familiar with the postulates of special relativity, which state that the speed of light and the laws of physics are the same in all inertial reference frames. Does that mean that nothing can travel faster than the speed of light as measured by an inertial observer? No! It means that *information* cannot travel faster than the speed of light as measured by such observers. Sinusoidal wave trains are routinely measured to move faster than light, but they carry no information in an already-established wave. On the other hand, modulation at the wave's source is information-bearing, and can only propagate along the wave at a speed no faster than the speed of light. For those of you who are familiar with wave theory jargon, this is the difference between the phase velocity and the group velocity of a wave.

A more exotic example at the information physics frontier is the Black Hole Information Loss Paradox. In this paradox, one imagines throwing an encyclopedia into a black hole. Stephen Hawking taught us that black holes aren't entirely black and that they radiate energy for quantum-mechanical reasons. In principle, one day a black hole will evaporate entirely, leaving nothing behind. The paradox is that quantum mechanics predicts that information is never lost in the universe, but if that's true, where did the information in the encyclopedia go? The answer to this question is still unresolved. One possible answer is that

the information is somehow subtly encoded in quantum correlations in the outgoing radiation, which is otherwise thermal. Another possible answer is that to the outside observer, the encyclopedia takes an infinite amount of time to fall in, and indeed the black hole itself would take an infinite amount of time to form as seen from the outside, evaporating by pre-Hawking radiation before it even formed an event horizon in the first place. The resolution to this paradox is viewed by some to be of critical importance to forming a consistent theory of quantum gravity. Indeed, some have gone so far as to say that discovering the resolution to this problem is as important to quantum gravity as discovering the resolution to "black-body radiation problem" was to quantum electrodynamics.

The examples I've given seem to be very much about relativity and information. But what about *quantum* information? And for that matter, what exactly *is* quantum information anyway? I'm sure a number of you who are taking this class have undergone exchanges something like the following:

"I'm taking this new class on quantum information this semester."

"Really? Sounds cool. So what is quantum information, anyway?"

"Umm...I have no idea. But I hope to find out!"

A concise definition of quantum information that is acceptable to everyone in the field is hard, and perhaps impossible, to come by. Part of the problem is that the best way to understand what quantum information *is* to understand what quantum information *does*. So defining quantum information is kind of like trying to define the word "knife" to someone who has never seen one before. Nevertheless, [pun alert] here's my stab at it:

> **Quantum information** *noun*: Information that is acquired, processed, or transmitted by a system whose description requires quantum mechanics.

It will turn out that quantum systems are perfectly good at manipulating what many would call classical information as well, so that is at least one way that this definition is lacking. The point I'm trying to make is, don't dwell too much on getting an exact definition right now. Let's get a sense of what one can do with quantum information first and perhaps later you yourself will be able to construct a better definition.

* * *

Why study quantum information? There are any number of reasons, but here are a few commonly-mentioned ones:

Physics. Quantum information helps us define the boundary of what is possible and impossible for the representation and manipulation of information in our universe. I've already given some examples along these lines. Personally, this is what motivates me the most.

Engineering. Faster = smaller = quantum. In other words, to make faster computers, components need to get closer together because information can only travel as fast as the speed of light. They therefore must get smaller. (And you thought it was just convenient

that as computers got faster they got smaller). Once computer components get small enough, quantum mechanics becomes necessary to accurately describe their functioning. Rather than viewing concepts like "Heisenberg uncertainty" as a barrier to pushing classical information onto quantum systems, perhaps there is a way to harness truly quantum information to do things even better. (On *Star Trek: The Next Generation*, there are even fictional devices called "Heisenberg compensators" that are supposed to handle this engineering barrier.) Perhaps some day there will be entire "quantum engineering" departments that work on these problems, but for now the subject of quantum engineering is mostly up to the physicists and materials scientists.

Computer science/mathematics. Quantum information is a vast undiscovered country of new algorithms and theorems. Even if it takes generations to build quantum information processing devices, the algorithms and theorems discovered now will last forever. Moreover, it is fair to say that theoretical quantum information applications have driven the experimental development of quantum information processing devices more than any other incentive.

* * *

A mantra often mentioned in the context of quantum information is

Information is physical.

What is meant by this is that information, in whatever form it takes, must be represented in a physical object somewhere. The physical laws governing all objects must therefore set the limits and possibilities for information processing. As these laws are quantum mechanical, these limits and possibilities must be stated in quantum-mechanical terms. At the same time, it is wise not to take this mantra too far. One of the greatest conceptual leaps made by humanity is the separation of information from its representation. We can do mathematics with pencil-and-paper, in our minds, or on computers, which are often more convenient than relying on sheep, fingers, or stones for our calculations. So another mantra is

Information is fungible.

In modern language, we recognize this as the hardware/software divide. It is possible to develop software completely abstractly without knowing the exact details of what hardware is implementing it. So in this course we will be dealing with quantum information concepts without considering what quantum hardware it will run on, except insofar as that hardware will be governed by quantum mechanics. Of course, the best software designers take into account hardware details and the best hardware designers take into account software details. For example, graphics processors famously are physically laid out in a way that mirrors the typical spatial transformations they will be expected to compute. So while we will largely ignore implementation issues in this course, we won't neglect them entirely.

* * *

I haven't given you much of a preview of what you can do with quantum information or the historical development of the field. That's the way a course like this typically opens,

but I thought I'd be a little different and give a fresh perspective. However, I still think it's important to get that "big picture" view of where this field came from and where it is going, so I'm going to refer you to some places to read about this. The first place I'd like you to read is your textbook, *Quantum Information and Quantum Computation* by Nielsen and Chuang, Sec. 1.1: *Global perspectives*. After that, for a more entertaining spin on some of the same subjects, I recommend reading Dave Bacon's Physics 599 Lecture notes *Introduction and basics of quantum theory*, Sec. II: *Whence quantum computing?* at http://www.cs.washington.edu/education/courses/cse599d/06wi/lecturenotes1.pdf. Finally, although some portions are a bit dated now, especially regarding experimental implementations, John Preskill's lecture notes are excellent and his Chapter 1 has my favorite overview of the field because it is so thorough; it can be found online at http://www.theory.caltech.edu/people/preskill/ph229/notes/chap1.ps.

For at least a brief teaser, let me end this overview with a list of what I consider to be a major result in each of the four areas of the course we'll be covering.

- A quantum algorithm that can factor an n-digit number in only $\mathcal{O}(n^3)$ steps, whereas the fastest-known algorithm (publicly, at least) takes a time $\mathcal{O}(e^{n^{1/3}(\log n)^{2/3}})$. This algorithm can crack the widely used RSA public-key cryptosystem.
- A proof that arbitrarily reliable quantum computation is possible, even if the components used to build the quantum computer are faulty and fail with an error rate no greater than one part in one thousand (higher or lower depending on assumptions about the technology). This is sometimes called the *threshold theorem* of fault-tolerant quantum computation.
- A quantum channel's simultaneously achievable capacity for classical communication, quantum communication, and entanglement generation or consumption, all in the absence of back-communication, has been completely solved in terms of entropic quantities. A similar three-way tradeoff generalizes the classical process of data compression to the quantum realm.
- A small shared key between two parties can be exponentially expanded using a quantum communication protocol whose security is based on the laws of physics and not the hypothetical complexity of some mathematical problem. This process, called *quantum key distribution* establishes a large secret shared key between the parties that can subsequently be used as part of a classical cryptography protocol known as the *one-time pad* whose security is based on information theory.

2 Bits & Gates

While there are many models of classical computing, perhaps the most practical is the circuit model. With it, one can decompose even the most complicated computation into simple parcels of information called bits that are acted on by simple operations called gates. (Later in the course we will discuss a generalization where information is represented in *dits*,

which are d-state systems.) Bits are elements of a two-element set we will denote by $\mathbb{B}$, and gates are functions between ordered lists, or *strings* of bits:

$$\begin{aligned}\text{Bits}: &\mathbb{B} := \{0,1\} \\ \text{Gates}: &f : \mathbb{B}^n \to \mathbb{B}^m.\end{aligned}$$

Parallel composition of bits and gates is represented by ordered lists of these; for example bits a and b combine in parallel to form the bit string (a,b), whereas the gates $f : \mathbb{B}^n \to \mathbb{B}^m$ and $g : \mathbb{B}^k \to \mathbb{B}^l$ combine in parallel to form the parallel gate $(f,g) : \mathbb{B}^{n+k} \to \mathbb{B}^{m+l}$. There is no sense in which bits combine in series, but gates combine in series through function composition. For example, the serial composition of the aforementioned gate f followed by g is the function $g \circ f$. A *circuit* is just a collection of gates composed in serial or in parallel. In essence, then, a circuit is nothing more than a very complicated gate.

One way to represent a gate is by its *truth table*, a table which shows which inputs map to which outputs. Truth tables for the well-known gates NOT, AND, OR, and XOR are given below, with canonical circuit elements for these gates drawn underneath. (`N.B. I haven't actually drawn the circuit elements yet.`) The symbols $\wedge$, $\vee$, and $\oplus$ are standard ones from Boolean logic used to denote AND, OR, and XOR respectively, whereas an overbar like $\bar{x}$ is the way we will denote the NOT of an input. The gates NOT and AND act exactly as we use them in the English language if we assign the symbol 1 the semantic meaning "true" and the symbol 0 the semantic meaning "false." However, it is the exclusive-or gate XOR, not the gate OR, that acts most like the way we use "or" in the English language. For example, when someone asks you, "Coffee or tea?," you know they don't mean "or both."

NOT

x	$\overline{x}$
0	1
1	0

AND

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

A gate that one usually doesn't give much thought to but which is very important in quantum mechanics (because of its absence) is the FANOUT gate. It simply makes a copy of the input bit on a new wire, as is shown in the table below.

FANOUT

x	x	x
0	0	0
1	1	1

When we get to quantum circuits, a gate we will talk about a lot is the controlled-NOT or CNOT gate. It is really just a classical gate, and is given by the truth table below. In words, its action is to apply a NOT gate to the target bit if the control bit is a 1 and do nothing to it otherwise. Either way, the control bit keeps its value at the output. Another way of thinking about this gate is as an XOR gate where the control bit comes along for the ride. That is because the target bit takes on the value of the XOR of its two input bits.

CNOT

x	y	x	$x \oplus y$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

A feature that the NOT and CNOT gates share that the other gates we discussed do not is that they are *reversible*. By that I mean that if one knows the truth table of the gate and its output values, then one can always reconstruct the inputs to the gate. For the case of NOT and CNOT, the gates are their own inverses, so they are manifestly reversible. However, generally a reversible gate is some *permutation* of its input bits.

Part of the interest in reversible gates comes from considering how to dissipate heat in computers. *Landauer's principle* states that each bit of lost information leads to the release of $k_B T \ln 2$ of heat, where k_B is Boltzmann's constant and T is the temperature of the system. I'm not going to dwell on or prove this statement, but it is fascinating and related to a puzzle known as *Maxwell's demon* that lasted for some time before Landauer's principle was put forward. I encourage you to read more about it if you are intrigued. The point I would like to make is that every AND and OR gate in a computer must dissipate at least $k_B T \ln 2$ of heat because one bit of information is lost in each. Technologically our chips today generate about 500 times more heat than that per gate operation, but perhaps some day this will become a limit. So an exciting outcome of reversible computing research is that in principle computation is possible without costing any power. Given how hot my laptop battery gets, I'd be happy to see this happen sooner rather than later!

Just how computationally powerful are reversible gates? Before answering that, I should answer the question of just how computationally powerful irreversible gates are. The following theorem demonstrates that the answer to this last question is "very."

Theorem 1. *The gate set* $\{\text{NAND} := \text{NOT} \circ \text{AND}, \text{FANOUT}\}$ *is* universal *for classical computation, i.e., any* $f : \mathbb{B}^n \to \mathbb{B}^m$ *can be decomposed into some combination of these gates acting in serial or in parallel on the input bits.*

I won't prove the above theorem because it would take us too far off our main line of development. One thing I would like to reinforce about the theorem, which you would see if you did go and prove it, is that the FANOUT gate is critical to include if one wants to achieve true universality. So next time you are at a cocktail party and someone says, "Well,

as everyone knows, the NAND gate is universal," you can reply, "With the FANOUT gate, of course."

Given the universality of irreversible gates, let's get back to the original question: Just how computationally powerful are reversible gates? It turns out that it is impossible to get a universal reversible gate set for classical computation that contains only one- and two-bit gates. (Yet another theorem I won't prove here!) So we need at least a three-bit gate somewhere. A gate that does the trick is the Toffoli gate (named for Tom Toffoli), denoted TOF, which is a controlled-controlled-NOT and whose truth table is the following:

TOF

x	y	z	x	y	$z \oplus (x \wedge y)$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Somewhat surprisingly, the Toffoli gate is universal for classical computation all on its own, given the ability to prepare input bits as 0 or 1 at will:

Theorem 2. *The gate* TOF *is universal for classical computation.*

Proof. An AND gate can be had by setting $z = 0$. A NOT gate can be had by setting $y = 0$ and $z = 1$. A FANOUT gate can be had by setting $y = 1$ and $z = 0$. □

That's all I want to say about bits and gates. It's time to ramp them up to the next level.

3 Pbits & Pgates

Bits and gates are great when one knows the inputs to a circuit with certainty. But what if one has *nonmaximal information* about the bit values? Then one must use probabilities to describe what's going on. In this case, gates and bits generalize to what we will call *pbits* and *pgates*. These are not standard names, but after using them for a while I think you'll agree that they are reasonable to use.

Pbits are described by a two-component vector of probabilities for each of the two possible

bit values, and pgates are matrices which left-multiply these pbits to yield new pbits:

$$\text{Pbits} : \vec{\mathbf{p}} := \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} \qquad \begin{array}{c} p_0 + p_1 = 1 \\ p_i \geq 0 \end{array}$$

$$\text{Pgates} : P := \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \qquad \vec{\mathbf{p}}' = P\,\vec{\mathbf{p}}$$

The elements P_{ij} of a pgate may be thought of as transition probabilities for moving from state j to state i. These are not totally unconstrained, because they need to map a valid input pbit to a valid output pbit. The conditions required of P to make this happen are summarized in the following theorem, which I state without proof. (The proof isn't hard, though; I encourage you to work out the proof yourself.)

Theorem 3. *A pgate P maps a pbit to a pbit iff P is a* stochastic *matrix, i.e., iff the following constraints are satisfied:*

$$\begin{aligned} P_{ij} &\geq 0 \qquad \forall\, i, j \\ \sum_i P_{ij} &= 1 \qquad \textit{(columns sum to 1)} \end{aligned}$$

A special case of stochastic matrices are permutation matrices, namely square matrices having entries that are 0 and 1 with the property that there is exactly one 1 in each row and column. Hence all reversible gates, which are permutations and which can be represented by permutation matrices in this formalism, are themselves pgates. Some examples of pbits and pgates are below:

$$\text{random bit} : \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad 0 \text{ bit} : \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad 1 \text{ bit} : \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\text{NOT} : \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{RANDOM SWITCH} : \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

* * *

The way to represent the series combination of pgates is by matrix multiplication. For example, pgate P_1 followed by pgate P_2 would be represented by by the product $P_2 P_1$. This is pretty straightforward; the only tricky part is remembering that matrices multiply right-to-left in successive actions.

Combining pbits and pgates in parallel is a bit more elaborate, but both are done the same way via a concept known as the *tensor product*. The tensor product of two pbits $\vec{\mathbf{p}}$ and $\vec{\mathbf{q}}$ is denoted $\vec{\mathbf{p}} \otimes \vec{\mathbf{q}}$ and is the vector

$$\vec{\mathbf{p}} \otimes \vec{\mathbf{q}} = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} \otimes \begin{bmatrix} q_0 \\ q_1 \end{bmatrix} := \begin{bmatrix} p_0 q_0 \\ p_0 q_1 \\ p_1 q_0 \\ p_1 q_1 \end{bmatrix}.$$

The tensor product of two pgates P and Q is similarly denoted $P \otimes Q$ and is the matrix

$$P \otimes Q := \begin{bmatrix} P_{00}Q & P_{01}Q \\ P_{10}Q & P_{11}Q \end{bmatrix} = \begin{bmatrix} P_{00}Q_{00} & P_{00}Q_{01} & P_{01}Q_{00} & P_{01}Q_{01} \\ P_{00}Q_{10} & P_{00}Q_{11} & P_{01}Q_{10} & P_{01}Q_{11} \\ P_{10}Q_{00} & P_{10}Q_{01} & P_{11}Q_{00} & P_{11}Q_{01} \\ P_{10}Q_{10} & P_{10}Q_{11} & P_{11}Q_{10} & P_{11}Q_{11} \end{bmatrix}.$$

This definition of the tensor product really only defines something called the Kronecker product because I have only specified its action on a component-wise representation of vectors and matrices. (I have only specified its action in a single *basis*.) There is a basis-free definition of the tensor product, but it's more abstract than we need; we will abuse language in this course and use the terms 'tensor product' and 'Kronecker product' interchangeably. For those interested in the subtleties of the basis-free definition of the tensor product, I recommend reading the textbook *Classical and Quantum Computation* by Kitaev, Vyali, and Shen listed on the course website as "recommended reading."

An obvious question to ask at this point is, "Why in the world are we using the tensor product to represent parallel composition of pbits and pgates when the much simpler process of grouping objects into ordered lists worked to represent parallel composition of bits and gates?" The answer has to do with probability theory. Two of the central axioms of probability theory (actually one is usually derived from other axioms) are the axioms of mutually exclusive and independent events. These axioms state that the probability of two mutually exclusive events add, whereas the probability of two independent events multiply:

$$\begin{aligned} p(E_1 \cup E_2) &= p(E_1) + p(E_2) && \text{[Mutually exclusive events]} \\ p(E_1 \cap E_2) &= p(E_1)p(E_2) && \text{[Independent events]} \end{aligned}$$

The tensor product representation is precisely what is needed in combination with matrix multiplication for series combination to get the correct output pbit components subject to these axioms. I'll leave it to you to work out a couple of examples to convince yourself that this is the case.

Finally, it's worth mentioning that not all pgates and pbits can be expressed as the tensor product of smaller pgates and pbits. However, all pbits can at least be written as a probabilistic sum of pbit states which themselves have tensor product decompositions. (Teaser: an analogous statement will no longer be true for qubits, and when no such decomposition exists we will say the qubits are in an *entangled* state.) For pgates, not even a decomposition into a sum of tensor product terms is always possible. An example of a pgate that has no such decomposition is the CNOT gate, whose pgate matrix representative is

$$\text{CNOT} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

When acting on the input state that is a tensor product of a random bit and a bit in the

state 0, namely a state whose pbit vector is

$$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix},$$

the CNOT gate produces an output state that is *classically correlated*:

$$\begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix}.$$

This output state is not decomposable into a single tensor product term, but can be described as a probability distribution over such states: the pbits have a 50% chance of being in the state 00 and a 50% chance of being in the state 11.

* * *

Next lecture we will discuss a new kind of nonlinear dynamics possible now that we have moved from bits and gates to pbits and pgates. That dynamics will be derived from the axiom of probability theory known as Bayes' rule.

`Add figures of circuit elements to notes.` ToDo