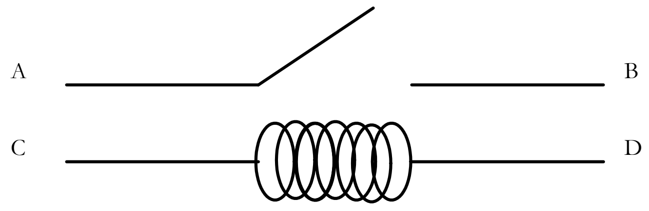
Electrons to Algorithms: The Nature of Computation

Tuesday, September 23, 2014 12:45 PM

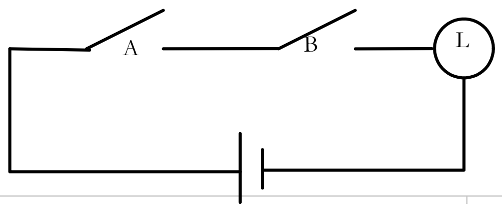
Computers: An Introduction
- Three important factors in a computational system:
    - State that has a representation
    - Ways to manipulate (change) the states
    - Algorithms that give context/meaning to the results of the manipulated states
- Computation is a physical idea: computation is a physical system/reality
- Computers don't require electronics to function!! They can be made out of various logical gates and systems i.e:
    - Tinker Toys
    - Dominoes
    - Hydraulic Valves
    - Marbles
    - Light
    - Slime Moulds…
    - Crabs…? (Look up this paper!!-Unconventional Computing Class)
    - Etc…
- Computers can do anything that involve information synthesis (poetry, art, math, etc; anything that can happen in the computer between your ears!)
- All you need is something that can represent two states: true(on, active, 1, etc…) and false(off, inactive, 0, etc…) --> Thus we get bits!
    - All you need is something capable of changing states, which is why we only need two states
    - Thus computers do everything is 2's (binary computation!!)
- But what about analog systems?? (Systems with continuous states; not just on or off)
    - Debatable: very few ideas can be truly analog…(at a fundamental level, electrons are either there or not)
    - Etc, etc, etc…
    - Therefore, bits are everything!!
- The big revolution in computing was the invention of electronic computing
    - The concept remained the same, but electricity is much faster, much more accurate, and has less room for error

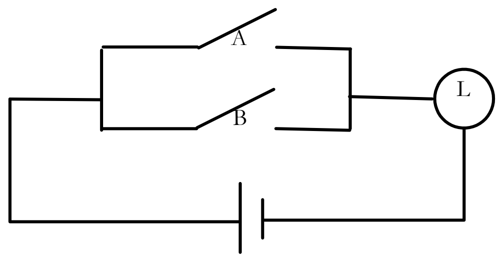
Electrons
- Electrons are weird…
- The only key factor about electrons needed: when electrons go through a wire, they generate a magnetic field
    - Electromagnetism at its finest
    - Therefore, you need to invent an electrical switch: current (or lack thereof) in one wire must cause current to flow or stop flowing in another current
        - This forms the basis of electrical bits: switch off = 0; switch on = 1

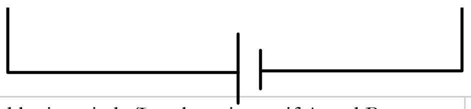


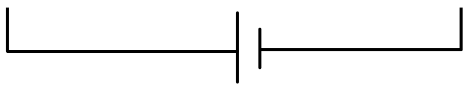
Current controlled switch (Relay): When current runs from C to D, it induces a current that magnetically attracts the switch causing current to flow from A to B
(Side note: the term "bug" in a computer theoretically came from times when actual bugs could enter the electronics and prevent the switches from functioning)

Logic
- Once you have binary switches, all you need to finalize the computational system is logic
- There are three main states of logic:
    - And
    - Or
    - Not
    - However, with a NAnd (Inverted And), you can create all other logic









And logic switch (L only activates if A and B are closed)			Or logic switch (L activates as long as A and B are both not open)

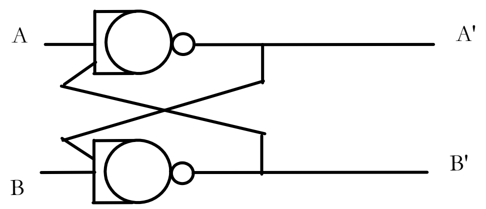
Modern Age Computers
- The main revolution in modern computers is making everything smaller (silicon sandwiches)
  - Integrated circuits are just layers of wires with silicon to separate them
  - With light-sensitive technology, microcircuits could be compacted even further with the precision accuracy of light
    - This process could be consolidated further with inversely magnified lenses

Creating Logic

Proof that Nots and Ors create And

A	B	A^B	AUB	NA	NB	NAUNB	N(NAUNB)
0	0	0	0	1	1	1	0
0	1	0	1	1	0	1	0
1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	1

Bits!!
- Bits are the fundamentals of the universe
  - But how do we store them? Remember, we need to be able to observe static states, and remember what the arrangements of the static states were in order to manipulate the program
    - Magnetic-Core array (Bits stored as the magnetic orientation of magnetic cores)
    - Flip-Flop (rewiring logical gates back into themselves to keep them in a specific states)



Flip-Flop Diagram

1 byte = 8 bits (256 arrangements which includes Aa-Zz, punctuation, 0-9, etc…)
- ASCII code was the "standardized" English code
- Unicode is now more universal

In modern computers, there exist "cells" of bits with their own addresses that the computer calls when the relevant information is needed
- This means all programs do is decode information, find out where to go, and find and decode the next set of information, etc…
  - Computers are one giant game of following the leader on a treasure hunt until you reach 00000000 (the end)
  - Think Assembly Computing

Notations and Control

Tuesday, October 7, 2014 12:40 PM

- Godel, Escher, Bach by Douglas Hofstadter (Connections between art, math, CS, logic, etc.)
- Digital Mantras by Holtzman

- How do we communicate ideas about…?
    - Language
    - Music
    - Visual Art
    - Computation
    - Etc…
    - There needs to be a way to communicate these ideas to a program in order for said program to not just use them, but to intelligently interpret and manipulate them. So how do we represent "knowledge" in a computer? Better yet, what is knowledge??? (Categories by Aristotle)

Two Key Concepts:

- Formal Systems (systems with interconnected structural relationships where the structure needs to be known in order to interact with the system)
    - "Formal" simply refers to "having to do with the form"
- Structural Relationships

Language as a Formal Structure:

- English has many levels of structure: phonetic, morphological, lexical, syntactic, semantic, pragmatic, etc.
- Parsing: Breaking down a "terminal" result into the logic of the computer (determining if a sentence is grammatically correct)-This is what compilers do! (Logic Trees!!)
- Programming Languages are just like any other language (think the syntax of Java/lisp)
    - Grammar is represented in the interpreter/compiler which is where the parsing takes place
        - YACC: Gives you a parser if you give it a syntax (essentially a way to write your own programming language!)
        - Linden Meyer System (~visual grammar)
    - Any representation of a rule will work in a programming language as long as there is distinction and maintains the structural relationships of the system
        - Thus it is important to attempt to represent each factor in a way that is most beneficial to your program

Why do we need Structure?

- The absence of structure is pure randomness
    - This is uninteresting
- Structure and interest are directly proportional

Music as a Formal Structure!!!

- The basic constituents of music are pitch and rhythm.
- Pythagoras's rules of music:
    - Whole number ratios have special significance
    - Whole number ratios define the most consonant musical relationships
- (Extended Music Theory Lecture…)
    - A Generative Theory of Modal Music Fred Lerdahl Ray Jackendoff

Computability

Tuesday, November 4, 2014 12:37 PM

<u>Can a Computer do it Survey:</u>

- Compute Bank Balances (w/ access to Bank Accounts): Yes (Lee says Yes)
- Print "There is a lot of stuff" as many times as there are grains of sand (given knowledge of number of grains of sand): Yes (Lee says Yes)
- Print "There is a lot of stuff" as many times as there are electrons (given knowledge of number of electrons): Yes (People say No) (Lee says Yes as long as there is no storage)
- Produce *new* music in the style of Mozart: Yes (Lee says Yes)
- Beat any human at Chess: Yes (Lee says Yes)
- Print the shortest path to visit every star system in the universe (given knowledge of every star system's position): No (People say Yes) (Lee says No: $O(c^n)$ Problem)
- Verify that a computer program has no bugs: No (Lee says No: "What is a bug?")
- Verify that a computer program has no infinite loops: No (1/2 1/2) (Lee says No)
- Sort a list of names of every human ever by alphabet (given the names): Yes (Lee says Yes: $O(n^2)$ Problem)
- Handshaking algorithm: No (1/2 1/2) (Lee says No)
- Fall in love: No (1/2 1/2) (Lee says Yes: We are computers!)

<u>Uncomputability:</u>

Program 1:

```
-Ask for an input n
-Run program 2 with n as an input
-If program 2 answers "It will stop"
        -while (true)
                -Answer "I will not stop"
-Else
        -Answer "I am stopping"
-END
```

Program 2:

```
-Ask for an input n
-Run program #n, and check to see if program #n will loop forever given n as an input
-If program #n loops forever
        -Answer "It will not stop
-Else
        -Answer "It will stop"
-END
```

Now, give Program 1 the input "1":

- If Program 2 replies that Program 1 will stop, Program 1 will loop forever
- If Program 2 replies that Program 1 will loop forever, Program 1 will stop
- Program 2 is always wrong…Thus Halting fails

<u>Some Infinities are Bigger than Others:</u>

The number of possible computer programs is countably infinite.

Quantum Computing

Thursday, November 6, 2014 12:31 PM

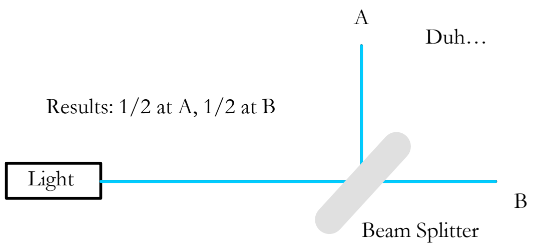
In essence, the ability of Quantum Computers (QC) is greater than those of Turing Machines (TM)

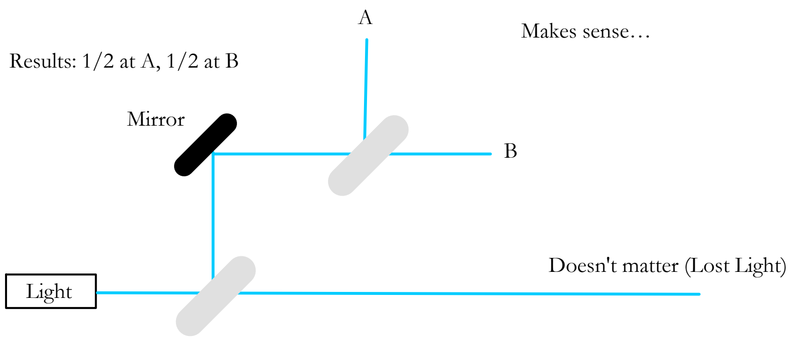
What is Quantum Computing?
- The best theory in science is quantum mechanics
    - Quantum mechanics is a universal scientific method that can describe all matter in the universe
    - All computers require quantum mechanics to work (transistors and microprocessors are only possible because of quantum mechanics)
    - George Gamow: Mr Thompkins Stories (Quantum Mechanics stories)
- QCs require highly dynamic, atomic-scale computation to function
    - A single bit in a computer is considerably larger than quantum states, which means computers are less powerful than they could be
    - QC technology takes advantage of quantum effects than TM technology cannot replicate
        - Ion Traps
        - Nuclear Spins (NMR/MRI): use of positrons
        - Optical Systems: use of photons
        - Qubits
        - Etc…

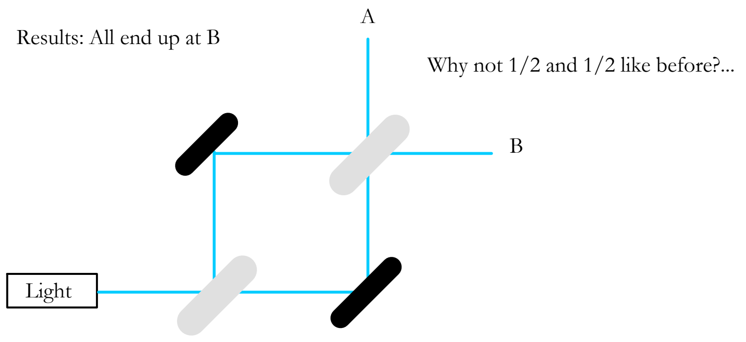
Why do we care?
- Moore's Law: The information that can be stored (and as a result, computational power) has been doubling, and we are approaching the level where quantum mechanics (or large parallel processing) is required to further Moore's Law
- Quantum Mechanics is much more powerful than classical mechanics in terms of computing

Why does Quantum Mechanics Work?
- In Quantum Mechanics, every possibility counts, even if they do not occur
    - This is because QM deals with superpositions, which are eventually concentrated into the events we perceive--but just because they don't happen doesn't mean they don't matter!
    - This means that quantum computing can run several computations at the same time using various superpositions and an exponential amount of various possibilities that can affect the process.
- Quantum Mechanics makes very little sense…but it's how the world works

The Beam Splitter-Photon Bomb Analogy:





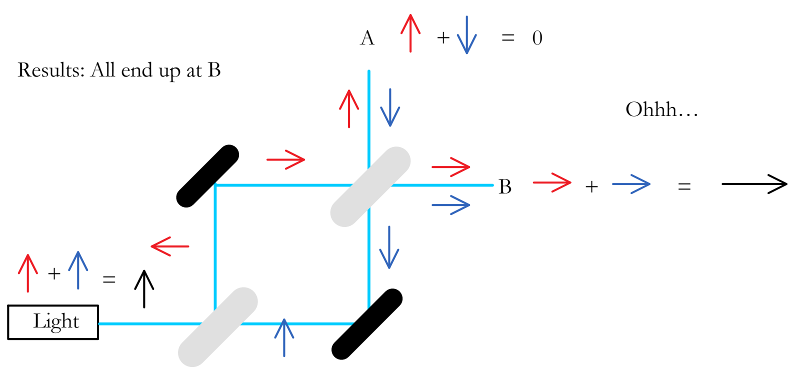


Why? Let's use arrows! (vectors)

Light Vector rules:

When light reflects off of a splitter, it rotates 90 degrees ccw

When light reflects off of a mirror, it rotates 180 degrees



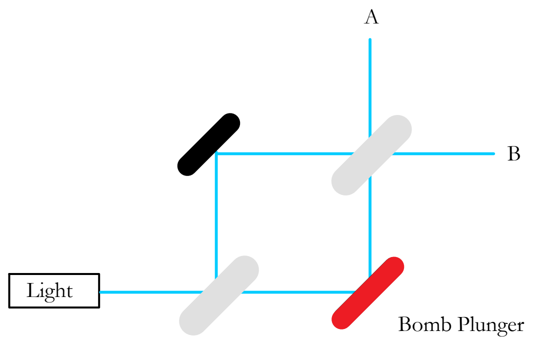
Photon-Triggered Bombs:

A bomb with a plunger on the front that detonates if a photon hits it

If the bomb is a dud, the plunger will stick, and it will act like a normal mirror

If the bomb is not a dud, the plunger will activate, and the photon will scatter

How do we distinguish the duds from the non-duds? Quantum Computing!



Results:

If the bomb detonates, it was good…

If a photon appears at B, there is no conclusion…

If a photon appears at A however, the bomb cannot act like a mirror, because otherwise it would have to appear at B, thus the bomb is good! (I know, QM is really weird…)

This was verified by experiment and is known as Elitzur-Vaidman Bomb Testing

Counterfactual Computation

Getting the result of a program without running the computer

By placing the computer in an interferometer, a photon can be sent through the machine and determine the result without affecting, activating the machine itself

Quantum Speed-Ups

One of the biggest powers of quantum mechanics is its added speed over classical computation

Grover's quantum database search algorithm: $O(\sqrt{n})$, beats the classical order $O(n)$

Shor's quantum prime factoring algorithm: $O(2^{n^{1/3}}\ldots)$ beats the classical order O(n3)

Quantum Consciousness

Penrose's Argument:

Brains can do X for X uncomputable (Brains can do uncomputable things)

Classical computer cannot do X

Therefore, brains aren't computers

This premise fails because brains cannot do all X

This would also imply that brains are even more powerful than quantum computers (because QC cannot solve the Halting problem)

Consciousness could potentially be linked to quantum effects in the brain, but there is very little evidence to support this…

Qubits

Similar to bits (can be "on" or "off")

BUT can also be in a superposition ("both" or "neither": note these are not extra states, they are a combination of states)

Use 2 complex numbers to determine the amplitude of each possibility

Can also become entangled, and something affected by one can influence the other

This means you need 2n complex amplitudes to represent n qubits, since there are some amplitudes that involve the interference of two amplitudes

Grover's Algorithm

Genetic Algorithms can produce new Quantum Algorithms

Complexity

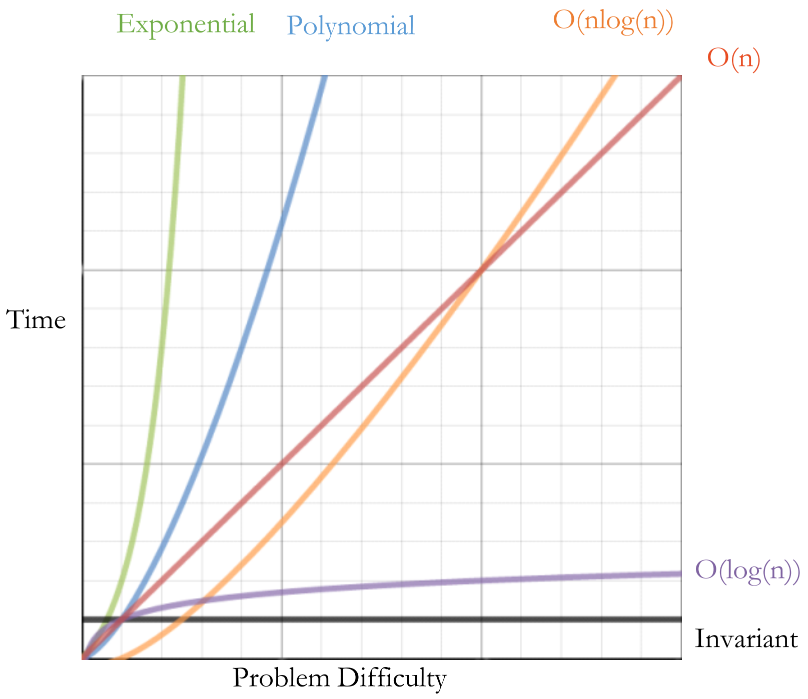
Tuesday, November 11, 2014 12:34 PM

The most useful measure of computer complexity is the measure of the growth rate of the number of steps proportional to the number of elements

- Most complexity measurements only measure speed
- Each assignment counts as a "step"
  - It is assumed that every step is equal
- Only the scaling factor of steps is accounted for
  - Consider the Traveling Salesman problem

Orders (from best to worst):

- O(n^2)
- O(nlog(n))
- O(n)
- Problem complexity is the best possible way a problem can be solved (for sorting, O(nlog(n)))



What is NP Complete (besides just O(2^n) or worse?)

- P is the class of problems that can be completed in polynomial time on a deterministic Turing machine[O(n^k)]
  - Turing Machines: Looks at a symbol, in a given state, and performs a combination of actions
    - Change the symbol it is observing
    - Change its current state
    - Move the list of symbols in either direction
    - Stop
- NP is the class of problems that is solvable on a *non*-deterministic Turing machine (meaning a symbol and state can result in multiple actions) in polynomial time
  - This means a solution can be verified in polynomial time, but not figured out in polynomial time
  - Big Question: P=NP???
- NP Complete means is it as difficult as any problem in NP
- NP Complete Problems:
  - Traveling Salesman (TS)

- Satisfiability Problem (Sat)
- Hamiltonian Circuit (HC)
- Etc. (There are many NP Complete problems)
    - Proving NP Completeness involves problem reduction (since Sat can be reduced to an instance of TS, TS must be as hard as Sat. Since Sat is NP, TS is NP Complete)

Number Systems

Tuesday, November 18, 2014 12:34 PM

Numbers are pretty frickin' important in coding

- Fundamental number systems use "counting numbers", the most fundamental of numbers
  - These are associated with a "mark" to represent each number e.g. tally systems
  - Over time, shorthands are developed for larger quantities
    - e.g. Roman Numerals (I, II, III, IV, V, …, X, …, L, etc…)
  - These number systems are limited however…
    - Zero is omitted
    - There is no way for simple algorithmic manipuation (+, -, x, /)
    - Negatives don't exist
    - Large numbers are difficult
    - Other "abstract" numbers do not exist (ratios/fractions, decimals, real numbers, imaginary numbers, infinity, etc…)
- The next level is base *n* systems e.g. base 10 (Decimal) or base 2 (Binary)
  - In essence, the base represents *how many times you run out of symbols* (Counting method)
    - 10 means "the zeroth element after running out of symbols once"
    - 56 means "the sixth element after running out of symbols five times"
    - 278 means "the eighth element after running out of symbols 7 times after running out of the number to represent how many times I ran out of symbols twice"
    - …This is why numerals are nice…
  - The other representation is based on exponential building (Exponent Method)
    - 59832 means: $5x10^5+9x10^4+8x10^3+3x10^2+2x10^1$
  - So then how does base 2 work?
    - Counting
      - 0 means nothing
      - 1 means "the first element"
      - 10 means "the zeroth element after running out of symbols once"
      - 11 means "the first element after running out of symbols once"
      - Etc…
    - Exponent
      - $101101 = 1x2^6+0x2^5+1x2^4+1x2^3+0x2^2+1x2^1$
  - In both cases, a "zero" element is now present, as well as instances of algorithmic manipulation
    - What about negatives??
      - A symbol often represents a "negative" indicator
      - In binary, a 1 in the leftmost position used to represent "negative", but this messed with the value of 0 as well as corrupting
      - Now negatives are represented by a "Twos-Compliment" Method
        - This makes the math work, but also results in "integer overflow" errors
  - Both Decimal and Binary have the limit of large numbers being difficult to represent. Enter Hexadecimal!
    - Base 16 is nice because it is a breakdown of Base 2
      - 0110100111010010 is hard to remember…so we break it into chunks and represent each number with a new symbol
        - 0110 = 6, 1001 = 9, 1101 = 13, 0010 = 2
        - To keep each block as one digit/symbol, A = 10, B = 11, etc. up to F = 15
        - So 0110100111010010 = 69D2 (which is a lot easier to remember!)
    - …And Hex is still compliant with either "number story"
      - 69D2 means "the second element after running out of symbols "D" times, after running out of numbers to represent the amount of times running out of symbols 9 times, after running out of numbers to represent the number of times to represent the number of times running out of symbols 6 times"

            - OR: $6x16^4+9x16^3+13x16^2+2x16^1$
- And then there are non-numeric data representations e.g. ASCII
    - In ASCII, a pattern of 8 bits represents a character rather than a numeral
        - In this way, 256 different characters can be represented (which is a lot seeing as all the uppercase and lowercase letters, as well as the numerals only take up 62 slots)

Artificial Intelligence

Thursday, November 20, 2014 12:35 PM

- History of AI
  - Originated in Grecian times (450$_{\text{B.C.}}$) using *Mesopotamian and Egyptian* logic (thanks Kwasi)
    - Influenced by the thoughts that all reasoning can be reduced to calculations (so that all arguments can be settled definitely)
      - "I want to know what is characteristic of piety which makes all actions pious […] that I may have it to turn to, and use as a standard whereby to judge all actions and those of other men." (Socrates, when Euthyphro wanted to turn in his own father for murder under pious intentions)
  - Leibniz then believed he had found a universal and exact system in algebra, which could precisely define all concepts of the universe.
    - "If someone would doubt my results, I would say to him: 'let us calculate, Sir,' and thus by taking pen and ink, we should settle the question." (Leibniz)
  - Modern-Day AI operates on one main basis: What the brain does may be thought of at some level as a kind of computation (Turing's idea)
      - *The Imitation Game* (movie) (Benny acts in this!)
    - Therefore: Brain ~ Turing Machine
      - This is why there is a school of CogSci here at Hampshire!! (Hooray!)
    - Said another way: AI is the study of mental faculties thought the use of computational models
- What is AI?

INPUTS		INTERNAL PROCESSES		OUTPUTS
		Deduction		
Vision		Search		Robotics
&	→	Planning	→	&
Language		Explanation		Speech
		Learning		

Simplified explanation of AI by Charniak & McDermott

  - Artificial Intelligence is sometimes defined as the study of how to build and/or program computers to enable them to do the sorts of things that minds can do. (Margaret Boden, Philosophy of AI)
  - AI research is the process of thinking about thinking via programming (Lee Spector)
- What is thinking?
  - Pattern-Based Rules
    - Eliza (Program to imitate a Rorcheran psychotherapist)
    - Look up emacs for built in Eliza
    - Eliza library for Processing (codeanticode)
  - "Logic is the essence of intelligence"
    - Can a computer solve inductive proofs?
      - Prologue (Logic Programming: Give a computer truths and then ask it questions)
  - "Understanding natural language is the best test of human-like intelligence"
  - "Learning is the only real hallmark of intelligence. Get learning right and everything else will follow."
  - "The brain's use of sub-symbolic, massively parallel computation is key. If we want truly intelligent computers, we must build them on the model of the brain, using neural networks."
  - "Evolution produced the only instances of intelligence of which we are currently aware. To produce intelligent computer systems we must harness the same principles of random variation and natural selection, using genetic algorithms."
  - "The process of figuring out what to do now"

        - *Fast, Cheap, and out of Control*
- Philosophy of AI
    - Turing Test
    - Chinese Room
    - Incremental Brain Transplant Argument

Evolutionary Computing

Tuesday, November 25, 2014 12:40 PM

- How does "revolutionary engineering" become "AI"?
  - In other words, how does computing "evolve" with time, and how do we further our understanding of our minds?
  - Evolutionary Computing decides that the "understanding" aspect isn't important, and by simply emulating the rise of intelligence (evolutionary thinking through trial and error), intelligence will arise regardless, even if there is no understanding of "intelligence" itself
    - Reproduction is a perfect example of this: creating intelligence without understanding intelligence itself
- What is the basis of Evolutionary Computing?
  - Random Variation
    - Random variations in generations emulate mutations that can or cannot be beneficial
  - Natural Selection
    - By selecting the beneficial mutations, and discarding the non-beneficial ones, "intelligence" will supposedly result
      - Spoiler Alert: It works!!

$$\begin{tikzcd}
 & \text{Random Generation} \arrow[d] & \\
 & \text{Assessment} \arrow[r] \arrow[dl] & \text{Solution} \\
\text{Selection} \arrow[r] & \text{Variation} \arrow[u] &
\end{tikzcd}$$

A Diagram of the Evolutionary Computing Process: The Genetic Algorithm

- Genetic Programming
  - Genetic Algorithm that produces written executable code
  - Programs are assessed by running and testing results
  - Automatic Programming via evolution
    - Finite Algebra Problems
      - Genetic Algorithms can produce results that the programmer is no capable of solving themselves
    - *Automatic Quantum Computer Programming: A Genetic Programming Approach* (Lee's Book)
      - Genetic algorithms can even generate quantum computer codes
- Evolutionary Computing is a "flavor" of AI
  - …And itself has many different "flavors" of its own.
- Digital Organisms
  - Studies the general principles of living systems
    - Population interactions
    - Population development
    - Etc.
      - Core War, Tierra, Avida, Echo, Polyworld, Framsticks, Breve
  - Unfortunate Necessities
    - Outrageous Simplifications
    - Feature Combination at radically different time & space scales
  - Symbolic Regression
    - Given "x", produce "y"
    - Primordial Ooze: +, -, *, %, x, 0.1
    - Fitness is inversely proportional to error
  - Push: Evolutionary Programming Language

Side Note: Lee is a badass because he challenged G.W. Bush's ideas about the lack of credibility of evolution (You go Lee!!)

AI and Creativity

Tuesday, December 2, 2014 12:29 PM

- Tony McCaffrey: Brainswarming
    - Can we remove the humans from this invention (and just use computers?)