**Code and Poetry: Hardware and Software 10-27-14**

* History of Western thought: dialectic between materialism (hardware) and idealism (software)
  + Plato (idealism): the world in front of you is a shadow of the real world
    - There is an ur-table which is the real, all the imperfect instantiations of tables we see are lesser tables
  + Epicurus (materialism): there’s no other world/world of forms, everything is materially bound, so study the things themselves rather than abstract forms
* History of computing: Llull – Leibniz – Boole – (Russell) – Shannon – Turing
  + Llull: philosopher, mystic, religious figure. Thought about universal language, universal truths about God. Come up with everything we know is true about God, come up with attributes of God – diagram and combine every proposition with every description in every possible way (combinatorial thinking: every combination you come up with is true). Suddenly you can see that some of the propositions about God weren’t discovered before (a new combination of descriptions). Universal language that is always true.
    - Used in procedurals, pulp fiction: combining plot elements at random.
    - This dream of the universal language, combinatorial thinking comes from Llull – making a language so it’s impossible to say anything false (only producing beautiful/true things with that vocabulary)
  + Leibniz: directly influenced by Llull, one of the founders of modern logic (Llull’s universal language is a precursor of logic). Wrote a paper on binary digits – can think of logic in term of binary digits.
  + Boole: and/or, but – restricts further what you can do with the language down to a limited set of operands.
  + Russell: takes idea of universal logic/language into the 20th/21st century
    - Russell discovered Wittgenstein, Wittgenstein taught Turing. Direct lineage into Turing
  + Shannon: takes Boolean language, Leibnizian binary digits. Sees that you can take Boolean logic and implement it as an electrical circuit
    - A ->- not A
    - Anything that comes in comes out as the opposite (A goes in, comes out not A, makes a noise coming in, is silent coming out)
    - AND/OR gates are software: abstract logical ideas – but can have a physical counterpart in the circuitry (hardware)
    - Anything you want to say/represent can go through logical gates
  + Turing/von Neumann: universal language (Llull) + universal logical (Leibniz) + universal machine
  + Computation born under the mystical/transcendent/religious and the militant
    - Politics/policies of mysticism are always violent
      * Have to kill/fight to get place in other world – easy to deny materiality/kill if you focus on transcendence
* Universal Turing Machine (“On Computable Numbers…”)
  + Concerned with computable numbers
  + The Halting problem: what is computable and what isn’t?
    - Before I start working on something, is it possible to compute it?
    - Turing aims to make a machine – if it can solve it, it’s solvable
  + 230-1: what is computable is what is effectively computable by the machine
  + Tape is a medium capable of holding a number or not
  + Notion that the machine only reads one thing at a time (has a reading head which is the machine, reads one number on the tape at the time – scan square, scan symbol)
  + M configuration: alters the makeup of the machine (change in the internal state of the machine – the machine is a set of configurations, self-reconfigurable)
  + Somewhere between M configuration and scan symbol determine the behavior of the machine
  + Machine can rewrite the tape (add something in blank square, or delete)
  + Writing, reading, storage: the basis of all storage media (here it’s just abstract, but this is how hardware works)
  + 232: at each state, the motion of the machine… is completely determined by the configuration
    - Configuration determined by the tape
  + Minimal Turing machine only needs 22 states, 2 operators
  + If this machine can solve the problem, it’s solvable; if not, it’s not
  + Because the machine can rewrite its state, it can imitate any other Turing machine if given enough tape (for the states)/time
    - Technically, according to this a computer from the 1950s can do whatever a modern computer can (but it would take a long time)
    - Only needs 22 states to emulate any other discrete state
    - State: combination of physical state plus whatever the machine just read
  + The program comes from the tape – the tape modifies the state to bring it into another state
  + Foundationally, all machines are Turing machines, but with stuff added/made more sophisticated (like explained in Moye)