**Code and Poetry Lab 10-29-14: Turing Machine**

* Negation machine
  + M1: A -> !A
* Circular vs. noncircular machines: if you want it to stop you need to program it to do so
* Affirmation machine
  + M2: A -> A (leaves it alone)
* Universal Turing machine:
  + Mu: can take as an input a whole machine (takes the circuit and drops it in)
  + Can take on the property of any other Turing machine
  + What is hardware and software here?
* What’s effectively computable is computable for Turing
* “We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions…m-configurations…’directly aware’…behavior of the machine ” (231).
  + Machine that is aware: influential in artificial intelligence
  + Whatever is the current symbol is part of its m-state
* 232: “If at each stage the motion of a machine…is *completely* determined by the configuration, we shall call the machine an ‘automatic machine’”.
* Self-rewriting machine: machine that can reconfigure its internal states
* 250: “the behavior of the computer at any moment is determined by the symbols which he is observing, and his ‘state of mind’ at that moment.”
* How does the author define the state of the machine?
  + State of mind ­+ internal state
* What is software here?
  + Not easy to get at this. The internal state is a physical (hardware) thing here. The symbol seems to be abstractable, so softwareish.
* What’s the difference between data and program?
  + Whatever data is physically in front of you is a part of the physical configuration/state
* Just a little thing in your BIOS that comes to life: boot process takes a little and manages to start up from there
* Emulator: take your Nintendo, emulate the processes on your laptop
* Turing machine can read and write symbols.
  + Discreteness of the states is important: not a continuum
  + Symbol-manipulation machine (doesn’t manipulate fields or anything else), but if you can reduce a field and the operation of a tractor to a set of symbols then you can emulate it.
* **Peer-empower**
* Weasel words: find three papers from hard drive, then convert into plaintext, then look up weasel words (like “very”, “clearly”, “absolutely”), make small list of weasel words, search for those weasel words in your writing, replace using sed (usually with nothing), find the lines you replaced and put them in a separate file
  + Left with corrected document without weasel words, document with old sentences and the new sentences
  + Destructive manipulation of data (data munging): if you can’t go back from the list of words into the novel, you’ve munged (you can’t return to the original text).
    - When you use sed or anything similar, make sure you’re dumping into new file (not changing the original file)
    - Create new file at each (increasingly destructive) step