Recursion Theorem

Informal idea: a Turing Machine can obtain its own description and compute with it

Formally:
- Given a TM $T$ that computes a function $t: (\langle M \rangle, w)$, we can construct a TM $R$ such that $r(w) = t(\langle R \rangle, w)$

Usage:
- If $T$ computes $t(\langle M \rangle, w) = \langle M \rangle$, then $r(w) = \langle R \rangle$. In other words, $R$ ignores its input and prints out a copy of itself (a Quine).
- Computer viruses need to propagate by copying their program.
- Any TM you write can include:
  - $M = "...$
    - Obtain, via the recursion theorem, own description $\langle M \rangle$. ...
  - "$"
Using the Recursion Theorem

$A_{TM}$ is undecidable

- Assume there exists machine $A$ that decides $A_{TM}$.
- $M = \text{“On input } w:\text{"}
  1. Obtain, via the recursion theorem, own description $<M>$.
  2. Simulate $A$ on input $<M, w>$.
  3. Output the opposite of what $A$ says.”
- $M$ on input $w$ halts accepts iff it doesn’t accept.

Fixed-Point Theorem

Given any computable function: $t(a) = b$

There is some Turing machine $F$ with:
- $t(<F>) = <G>$ where $L(G) = L(F)$

For example, there are two Turing machines, $M$ and $N$ with $<M> = 2^*<N>$ and $L(M) = L(N)$

Proof:
- $F = \text{“On input } w:\text{"}
  1. Obtain, via the recursion theorem, own description $<F>$.
  2. Compute $t(<F>) = <G>$ (Description of TM $G$)
  3. Simulate $G$ on $w$."
Creating a Quine

Want to build a TM SELF that outputs its own description <SELF>
- First, define (computable function) q(w) = <P_w>: Pw is a machine that prints out w and halts.
- A: prints out <B> (A = P_<B>)
  - We’ll figure out what <B> is once we write B!
- B:
  1. Computes q(<B>) = <P_{<B>}>= <A> (How does it know <B>?)
  2. Combines <A> and <B> into new machine
  3. Writes the description of the machine to the tape and halts.

Creating a Quine in Ruby

Useful commands:
- puts(string) prints out string without quotes
- string1 % string2 returns string1, but replacing %s with string2
- “abc%sdef” % “012” ==> “abc%012”
- string.inspect returns string with quotes around it (and escaping any necessary embedded characters).

Creating a Quine in C

http://www.nyx.net/~gthompso/quine.htm