Lecture 37: Recursion Theorem

CSCI 81
Spring, 2013

Kim Bruce

Virus Program

- virus() =
  1. For each address in address book do:
     1.1. Write a copy of myself.
     1.2. Mail it to the address.
  2. Do something malicious like change one bit in every file on the machine.
  3. Halt.
- Can we implement step 1.1?
  - Print two copies of the following with the second in quotes: "Print two copies of the following with the second in quotes:"

Fixed Points

- Consider f(k) = k if k < 1
  = f(k-1) + f(k-2) otherwise
- Function f defined in terms of itself.
- Think of as equation to be solved.
  - f = fun(k). if k<1 then k else f(k-1)+f(k-2)
- Write right side as function of g:
  - F(g) = fun(k). if k<1 then k else g(k-1)+g(k-2)
- Looking for f s.t. f = F(f) *f is fixed pt for F*

Recursion Theorem

- Rough versions:
  - First Thm: If F is computable then F has a computable fixed point.
  - Second Thm: We can compute the program for a fixed point of F from a program for F.
- True for any formalism giving all computable fcns.
  - In lambda calculus, $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ gives fixed points. I.e. for all f, if $x_o = Yf$ then $f(x_o) = x_o$

Toward the Recursion Thm

- Lemma: There is a computable function $q : \Sigma^* \to \Sigma^*$, where if w is any string, q(w) is the description of a Turing machine P_w that prints out w and then halts.
  - Proof: Compute q as follows: Given w, let P_w be the Turing machine that erases its input, prints out w and then halts. Return a description of P_w.
    - Try writing P_{abba}
  - Computable by Church-Turing thesis.

Self-reproducing Program

- Self is procedure A followed by B -- prints out <AB>
- A prints out description of B, & B prints out description of A when starting with description of self (<B>) on tape -- done differently.
- A: Use $P_{<B>}$ from previous slide: it prints <B> & halts. Thus <A> = q(<B>) = description of program that prints <B>
  - Problem: Haven't defined B yet!!

Self-reproducing Program

- B will compute A from A's output
  - Recall <A> = q(<B>), so if B can get its own code, then can compute A!!
  - But A left <B> on tape when completed.
  - So after A finishes, B grabs <B> from output tape and runs q(<B>) to get <A>, and now write <A> in front of <B> on output tape.
    - Thus output tape contains <AB>
    - Define B to run on any <C>: runs q(<C>) to get <D> and write <D> in front of <C> on tape.
  - B is well-defined action, so now tie knot and have A print <B> on its tape.

In English ...

- From before:
  - Print two copies of the following with the second in quotes: "Print two copies of the following with the second in quotes:"
  - Part B is : Print two copies ... in quotes
  - Part A is: "Print two copies ... in quotes:"

Recursion Thm for TMs

- Recursion Theorem: Suppose T is a Turing machine that computes a (partial) function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.
  There is a Turing machine R that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every w,
  $r(w) = t(\langle R \rangle, w)$.
- Note: "=" means either both defined and same or both undefined
- Proof similar to the above.
- Intuition, machine can use its own encoding to do further computations.

Using Recursion Theorem

- Alternate proof that A_{TM} undecidable:
  - assume H decides A_{TM}.
  - Construct B:
    - On input w, obtain description <B> of B using recursion thm
    - Run H on <B,w>
    - Do opposite of what H says: Accept if H reject and reject if H accepts.
  - When run B on w, it does opposite of what H says it will do. Thus H can't be correct.

Using Recursion Theorem

- Min_{TM} not semidecidable
  - For contradiction, assume E enumerates MIN_{TM}.
  - C: On input w, write own description <C>
    Run enumerator until get some <D> longer than <C>
    Simulate D on w.
  - Note C and D do same thing, but <C> shorter.
    However <D>, which is clearly not minimal, is in the enumeration!

Godel Incompleteness

Paradoxes?

- This statement is not true.
- This statement is not provable.
- Second is at heart of Gödel Incompleteness.

Gödel Incompleteness

- Recall that T is consistent iff no false statement has a valid proof. ($T \vdash \phi \Rightarrow T \vDash \phi$).
- T is complete iff every true statement has a valid proof. ($T \vDash \phi \Rightarrow T \vdash \phi$).
- Gödel Completeness:
  In predicate logic, if $T \vDash \phi$, then $T \vdash \phi$.

Incompleteness Theorems

- Different notion of completeness -- w.r.t. model
- Gödel Incompleteness 1: For every "interesting" system there are true statements that cannot be proved.
- Gödel Incompleteness 2: For every "interesting" system, the consistency of that system cannot be proved within itself.