Representations

Models of Computation

- Thue systems
  - Post systems
- Lambda calculi
- Partial recursion
- Turing machines
- Markov normal algorithms
- Minsky counter machines
- Type 0 languages
- Kolmogorov-Uspenskii machines
- Neuring machines
- Wang machines
- Random access machines
- Quantum computers
- Billiard ball computers
- Fortran, Algol, Lisp, C, Pascal, Logo, Ada, Java, ...

Today

- Multihead, multitape, multidimension Turing machines
- Counter machines
  - 1, 2, 3, many
- Recursive functions
  - Primitive recursion
  - Minimization

Everyone says...

"The remarkable result about these varied models is that all of them define exactly the same class of computable functions: whatever one model can compute, all the others can too!"

— Bernard Moret

Numbers as Strings

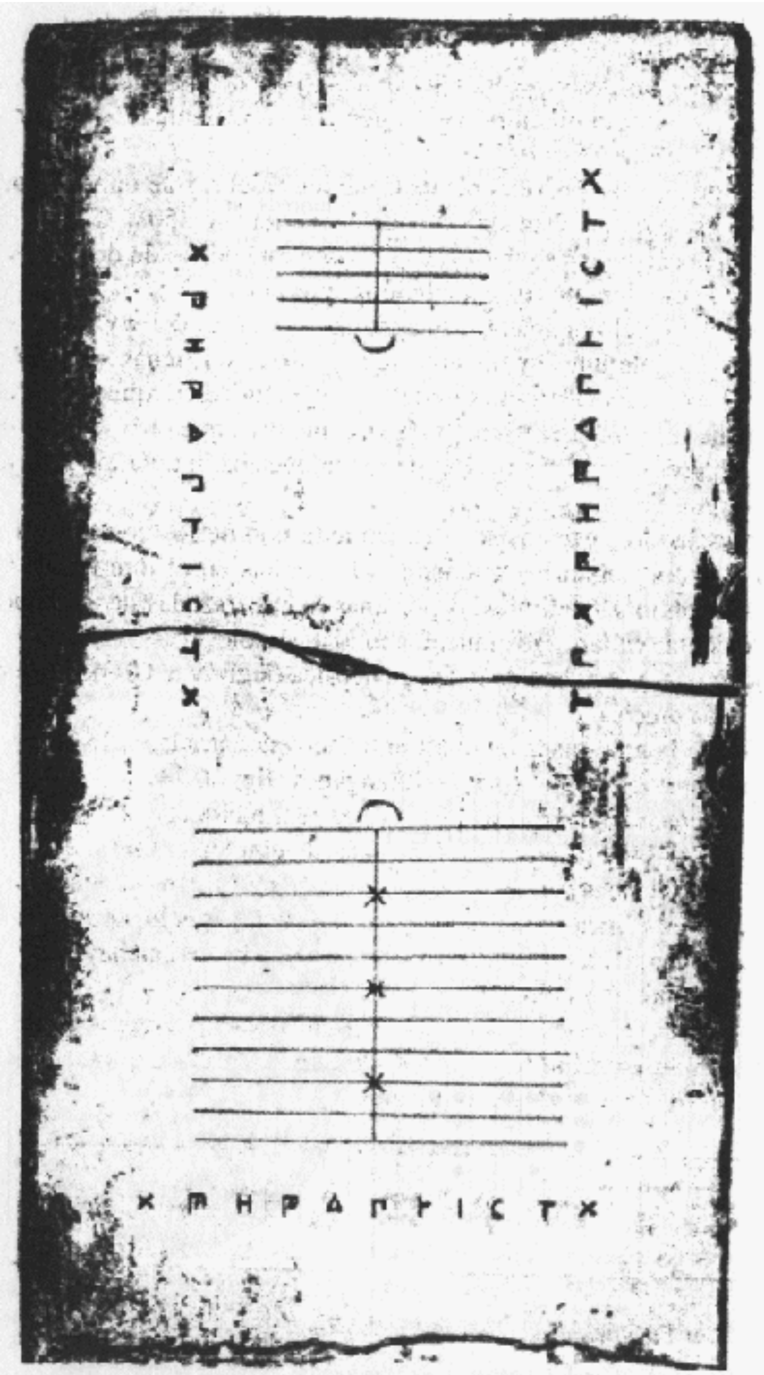
- Decimal
- Binary
- Unary (tally)
- 4 bits per decimal digit

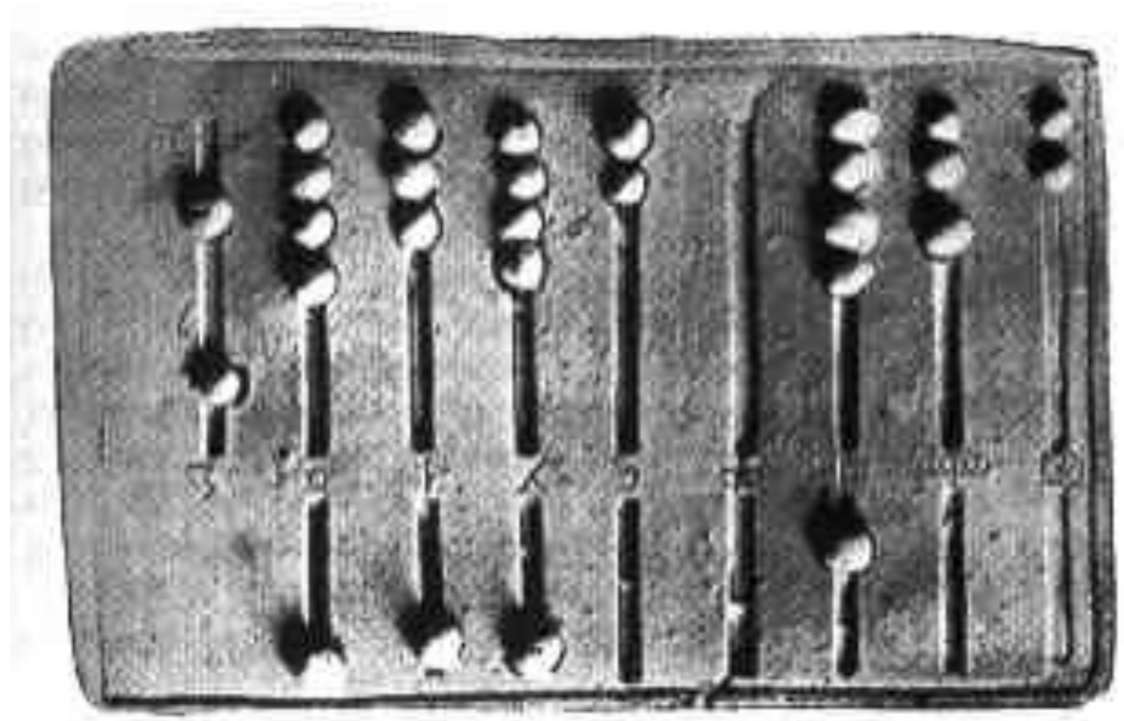
Strings as Numbers

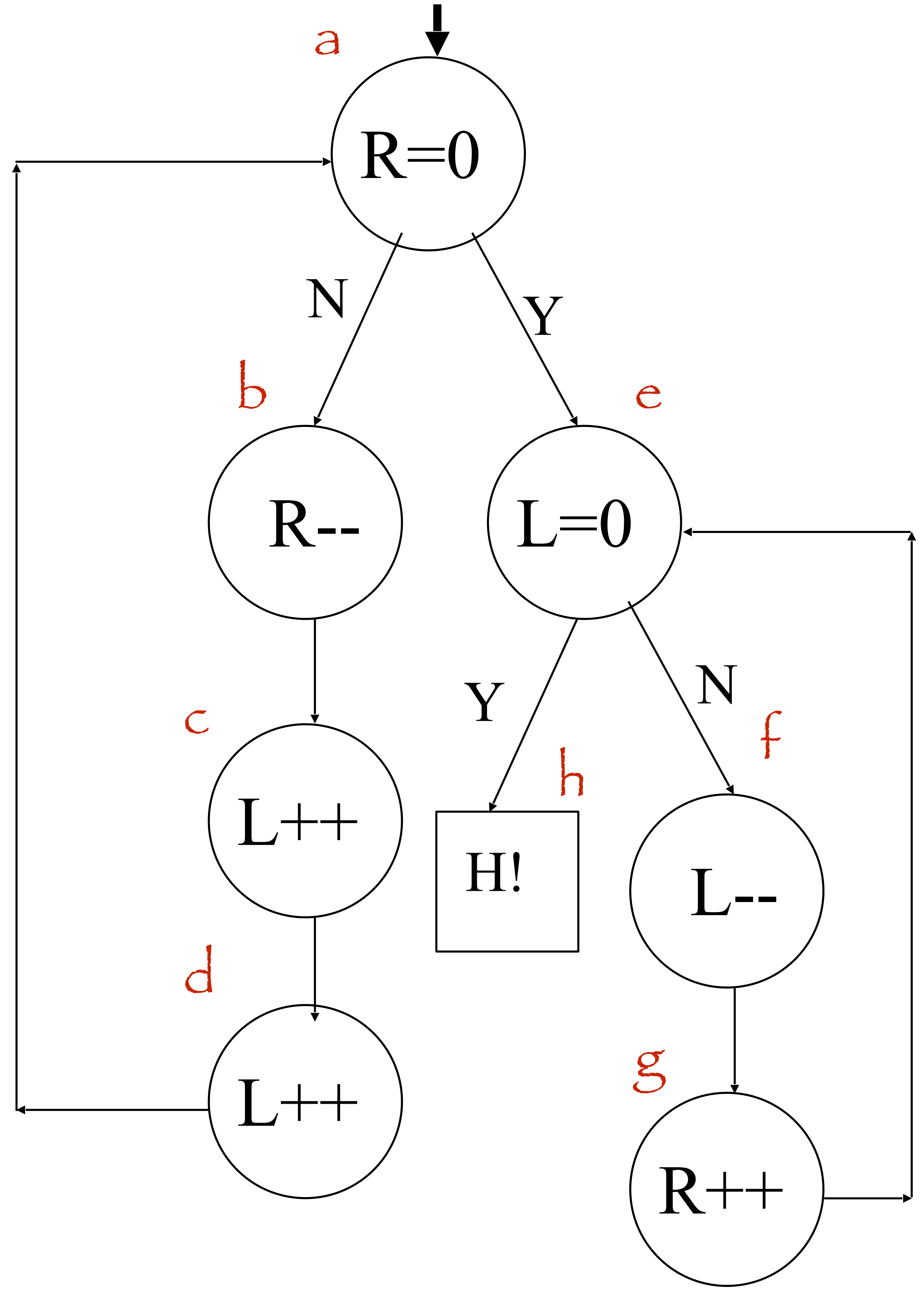
- Base $|\Sigma|$
- Gödel number
- List of numbers

Counter Machine

- Worry beads
- Rosaries
- Brolni [Russia]
- Komboloi (κομπολογια) [Greece]
- Mala [India]
- Jyuzu [Japan]
- מחרוזות תפילה
- מסבחה






a
R=0
N
Y
b
R--
e
L=0
c
L++
Y
N
h
f
H!
L--
d
L++
g
R++

Counter Machines

- Restrict (multitape) TM
  - Alphabet: 0 1 _
  - Tape always: 01*_*
  - Reading/writing at end
    - Change first _ (blank) to 1
    - Change last 1 to _
    - Test for 0

Counter Machine

- 1 counter: very weak (exercise)
- 2 counters: medium
  - Can't compute square or exponential
  - Can compute everything
    - if n is represented by 2^n 1's
- 3 or more: Everything

Primitive Recursion

- 0
- successor +1
- projections
- composition
- f(x,n) := if n=0 then g(x) else h(f(x,n-1),x,n-1)

Ackermann's Function

- $A(0,n) = n+1$
- $A(m+1,0) = A(m,1)$
- $A(m+1,n+1) = A(m,A(m+1,n))$

Lemma: $A(m,n) > m+n$

- Induction on (m,n)
  - $A(0,n) = n+1 > n$
  - $A(m+1,0) = A(m,1) > m+1$
  - $A(m+1,n+1) = A(m,A(m+1,n)) > m+A(m+1,n) \geq m+n+2$

Lemma: $x>y \Rightarrow A(m,x) > A(m,y)$

- Induction on (m,x)
- Assume $x>y$
  - $A(0,x) = x+1 > y+1 = A(0,y)$
  - Is $A(m+1,x+1) > A(m+1,y+1)$?
  - By induction, $A(m+1,x) > A(m+1,y)$
  - $A(m+1,x+1) = A(m,A(m+1,x)) > A(m,A(m+1,y)) = A(m+1,y+1)$

Lemma: $x>y \Rightarrow A(x,n) > A(y,n)$

- Induction on (x,n)
- Assume $x>y$
  - $A(x+1,0) = A(x,1) > A(y,1) = A(y+1,0)$
  - $A(x,n) > x+n \geq n+1 = A(0,n)$
  - $A(x+1,n+1) = A(x,A(x+1,n)) > A(y,A(x+1,n)) > A(y,A(y+1,n)) = A(y+1,n+1)$

Lemma: $A(m+n+2,x) > A(m,A(n,x))$

- Induction $(m+n,x)$
  - $A(n+2,x) > A(n+1,x) \geq A(n,x)+1 = A(0,A(n,x))$
  - $A(m+n+2,0) = A(m+n+1,1) > A(m,A(n-1,1)) = A(m,A(n,0))$
  - $A(m+n+2,x+1) = A(m+n+1,A(n+m+2,x)) > A(m,A(n,A(m,x))) > A(m,A(n,x+m)) \geq A(m,A(n,x+1))$

A isn't Primitive Recursive

- Denote $x = x_1,\ldots,x_k$ and $x_m = \max x_j$
- Say $A_i > g$ [majorize] if $A(i,x_m) > g(x)$ for all x
- Easy: $A_0 > 0$; $A_1 > +1$; $A_0 > proj_i$
- Suppose $f(x) = h(g_1x,\ldots,g_kx)$, $A_s > g_1,\ldots,g_k,h$
    - $A_{2s+2} > f$: $A(2s+2,x) > A(s,A(s,x)) > A(s,\max\{g_jx\}) > h(g_1x,\ldots,g_kx)$

A isn't Primitive Recursive

- Suppose $A_s > g,h$ and $f(x,n)=\text{if } n=0 \text{ then } g(x) \text{ else } h(f(x,n-1),x,n-1)$
- $A(r,n+x_m) > f(x,n)$, $r = 2s+1$, by induction on n:
  - $f(x,0) = g(x) < A(s,x_m) < A(r,0+x_m)$
  - $f(x,n+1) = h(f(x,n),x,n) < A(s,\max\{f(x,n),n,x_m\}) < A(s,A(r,n+x_m)) < A(2s,A(r,n+x_m)) = A(r,n+1+x_m)$
- $f(x,n) < A(r,n+x_m) < A(r,2N+3)= A(r,A(2,N))< A(r+4,N)$ where $N = \max\{n,x_m\}$

General Recursion

- Also minimization
  - f(x) := min n s.t. h(x,n)=0
    - where h is (primitive) recursive
  - Can loop

Questions

- In what sense are

  Turing machines =

  Lambda calculus =

  Recursive functions ?

- In what sense are

  Analogue computers >

  Turing machines >

  Primitive recursion ?

Alonzo Church

The fact... that two such widely different (and in the opinion of the author) equally natural definitions of effective calculability turn out to be equivalent adds to the strength of the reasons adduced below for believing that they constitute as general a characterization of this notion as is consistent with the usual intuitive understanding of it.

Princeton Course

- "The definition of a Turing machine is very robust."
  - Multiple heads
  - Multiple tapes
  - Multiple states
  - Multiple directions
  - Multiple dimensions
  - Multiple worlds

Equivalence of Models

- $TM_2 \propto TM$ [1 tape; 2 channels]
- $CM_2 \propto TM_2$ [111...1BBB...]
- $CM_n \propto CM_2$ [$2^i 3^j 5^k 7^l \ldots$]
- $RAM \propto CM_n$ [$2^x(2y+1)$]
- Scheme $\propto$ RAM [Abelson & Sussman]
- TM $\propto$ Scheme [Interpreter]

Goal

Determine when one model is as powerful as another.

- Intensional (ignore complexity)
- Models operating over different domains

Subrecursive Models

- Primitive recursion
  - Multiple recursion
- Typed lambda calculi
- Finite state automata
  - Nondeterministic FSA
- Pushdown automata
  - Deterministic PDA
- Nerve nets
- Linear-bounded automata
- Single-counter machines
- BLooP
- Univac, Burroughs, IBM 1130, PDP-11, Mac, Pentium IV, Connection Machine, ...

Super-recursive Models

- Oracle machines
- Trial and error predicates
  - Limit recursive functions
- Real recursive functions
- Inductive Turing machines
- Abstract state machines
- Analog recurrent neural networks

Containment

$B \approx_c A$ if $B = A$

for every $f \in A$ there is a $g \in B$ s.t. $g = f$, and vice-versa

$B >_c A$ if $B \supset A$

for every $f \in A$ there is a $g \in B$ s.t. $g = f$, but not vice-versa

Examples: Containment

- Primitive recursion is weaker than general recursion.
- Recursion is stronger than iteration.
- To show that [Inductive Turing machines] are more powerful [than ordinary TMs], we need to find a problem solvable by an ITM and insolvable by a TM …. —Mark Burgin

Hartley Rogers Jr.

"Given a class of nonnumerical inputs and outputs, choose some fixed one-one mapping from this class into the integers. Such a standard mapping is called a coding."

Rogers

- The use of codings raises an immediate question of invariance.

- Once a coding is chosen, will the formal concept partial recursive function on code numbers correspond to the informal notion algorithmic mapping on the uncoded expressions? ... Church's thesis provides an affirmative answer.

Simulation

$B \geq_s A$ if

there's an injective mapping

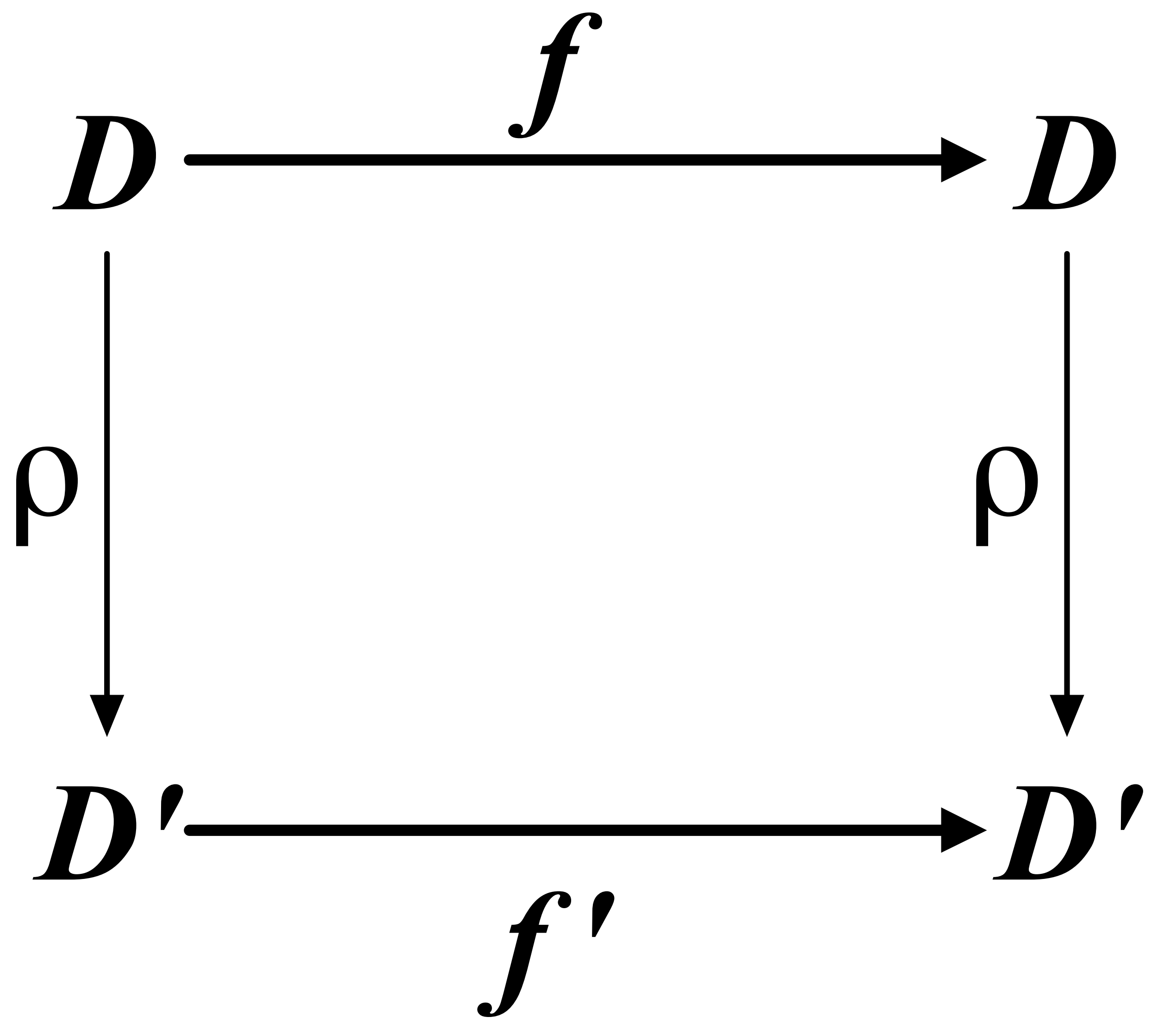
ρ: dom $A \rightarrow$ dom B

s.t. for every $f \in A$

there's a $g \in B$

for which $g(\rho(x)) = \rho(f(x))$ for all x

Simulations



f' simulates f via injection ρ

Examples: Simulation

- TM $\approx_S$ Rec $\approx_S$ λ
  - ρ: **N** $\to$ {1,B}* [Tally numbers]
  - ρ: **N** $\to$ Λ [Church numerals]
  - ρ: Λ $\to$ **N** [Gödel numbering]

Problems…

1. Information can be hidden in a mapping
    - even uncomputable information

2. Different mappings can have opposite effects

3. The three methods are incompatible

Strictly Stronger

- $\geq_S$ is a quasi-ordering

  (reflexive & transitive)

- A $>_S$ B if A $\geq_\rho$ B, via injection ρ,

  but B $\not\geq_\tau$ A via any $\boldsymbol{\tau}$

The Coding

- Let s be successor

  and s' simulate it: $s'\rho = \rho s$

- ρ is recursive is s' is:
  - $\rho(0) = \text{constant}$
  - $\rho(n+1) = \rho(s(n)) = s'(\rho(n))$

- If s is primitive recursive, so is ρ

Rec $>_S$ Prim

- Suppose Prim $\geq_\rho$ Rec
  - $\rho \in$ Prim
- Consider $h(n) = \rho(\min_i \{ \rho(i) > ack(n,n)\})$
  - $\rho^{-1} h \in$ Rec
  - $\rho\, \rho^{-1} h = k \rho$ for some $k \in$ Prim
  - $h = k \rho \in$ Prim
- But it isn't

Rec is Maximal

- Suppose s' is recursive
- Then ρ^{-1} is partial-recursive (it is not always defined)
- For any f $\in$M
  - Since f = $\rho^{-1} g \rho$, f $\in$ Part-Rec
  - rng $g\rho$ = rng ρf $\subseteq$ rng ρ
- Hence f $\in$ Rec