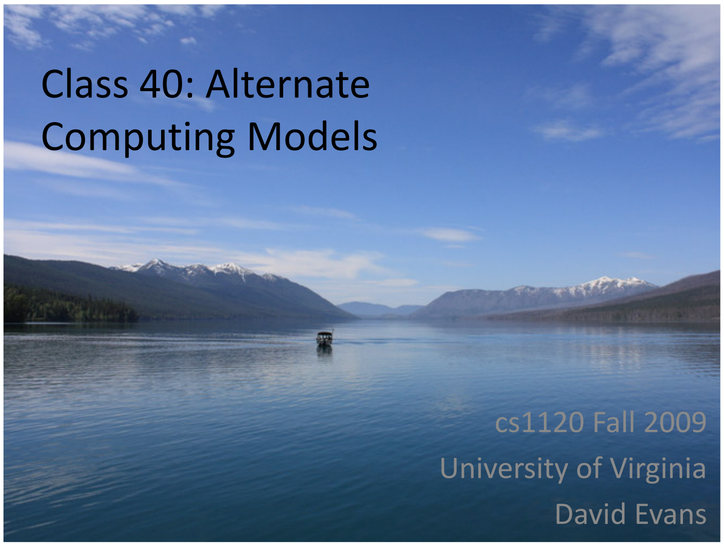


Class 40: Alternate Computing Models

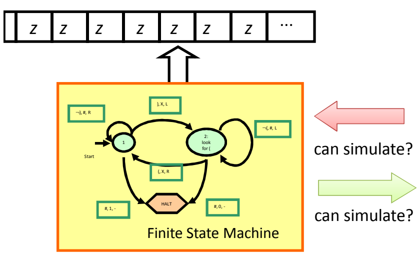
cs1120 Fall 2009
University of Virginia
David Evans

Reminders

- To qualify for a presentation your team must send me an email containing the URL of your site (with some working basic functionality) **before 4:59pm Sunday**
  - Presentation time will be divided between qualifying teams
  - Teams will present in reverse order of qualification time
  - Non-presenting teams only turn in reports instead (before midnight Monday)
- **Final** will be posted Monday, and due Friday (Dec 11)
  - If this scheduling causes you undue hardship, it may be possible to get an extension to Monday (Dec 14)

I will have extended extra office hours (either in my office or Small Hall) on Sunday afternoon, 1:30-5pm (groups that upload projects by Saturday will have priority)

Equivalent Model Computers?



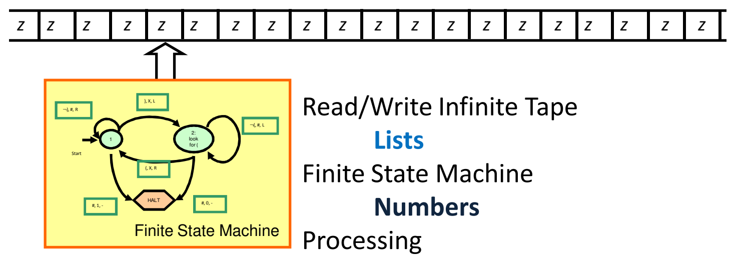
term = *variable*
| *term term*
| (*term*)
| **λ** *variable* **.** *term*

$(\lambda x.\ M)N \Rightarrow_{\beta} M$
with xs replaced by N

Turing Machine

Lambda Calculus

Simulating a Turing Machine



Read/Write Infinite Tape
Lists
Finite State Machine
Numbers
Processing
Way to make decisions (if)
Way to keep going

In search of *the truth*?

- What does **true** mean?
- **True** is something that when used as the first operand of **if**, makes the value of the **if** the value of its second operand:

  (**if T** *M N*) → *M*

Don't search for **T**, search for **if**

$$\mathbf{T} \equiv \lambda x\ (\lambda y.\ x)$$

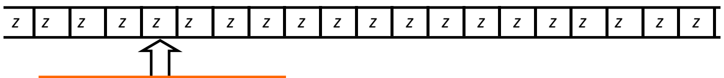
$$\mathbf{F} \equiv \lambda x\ (\lambda\ y.\ y))$$

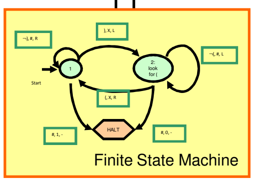
$$\mathbf{if} \equiv \lambda p\ (\lambda c\ (\lambda a\ .\ (pc)a)$$

Just like in **LazyScheme**:

(define true (lambda (a b) a))
(define false (lambda (a b) b))
(define if (lambda (p c a) (p c a))

Simulating a Turing Machine





Read/Write Infinite Tape
    Lists
Finite State Machine
    Numbers
Processing
    Way to make decisions (if)
    Way to keep going

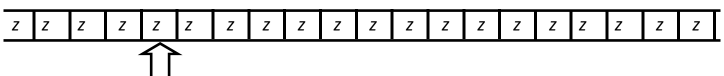
Making Lists

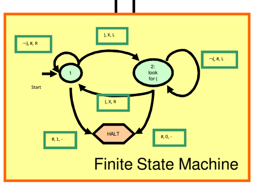
(**define** (make-pair x y)
  (**lambda** (selector) (**if** selector x y)))
(**define** (car-of-pair p) (p **true**))
(**define** (cdr-of-pair p) (p **false**))

$\mathbf{cons} \equiv \lambda x.\lambda y.(\lambda z.(z\ x\ y))$
$\mathbf{car} \equiv \lambda x.(x\ \mathbf{T})$
$\mathbf{cdr} \equiv \lambda x.(x\ \mathbf{F})$
$\mathbf{null} \equiv \lambda x.\mathbf{T}$
$\mathbf{null?} \equiv \lambda x.\ (x\ (\lambda y\ .\lambda z\ .\ \mathbf{F}))$

$\mathbf{T} \equiv \lambda x\ (\lambda y.\ x)$

Simulating a Turing Machine





Read/Write Infinite Tape
    Lists
Finite State Machine
    Numbers
Processing
    Way to make decisions (if)
    Way to keep going

What is 11?

eleven **elf** **undici**
11 十一
once **XI**
أحد عشر **onze** **одиннадцать**
イレブン

Meaning of Numbers

- "11-ness" is something who's **successor** is "12-ness"
- "11-ness" is something who's **predecessor** is "10-ness"
- "Zero" is special. It has a **successor** "one-ness", but no **predecessor**.

Meaning of Numbers

(pred (succ N)) $\rightarrow N$
(succ (pred N)) $\rightarrow N$
(succ (pred (succ N))) $\rightarrow$ (succ N)

(zero? **zero**) $\rightarrow$ **T**
(zero? (succ **zero**)) $\rightarrow$ **F**

Is this enough?

Can we define **add** with **pred**, **succ**, **zero?** and **zero**?

add $\equiv \lambda x\,.\ \lambda y.$
(**if** (**zero?** x) y (**add** (**pred** x) (**succ** y))

Can we define lambda terms that behave like **zero**, **zero?**, **pred** and **succ**?

Hint: The *length* of the list corresponds to the number value.

Making Numbers

$\mathbf{0} \equiv \mathbf{null}$

$\mathbf{zero?} \equiv \mathbf{null?}$

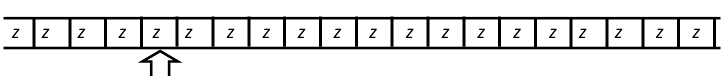
$\mathbf{pred} \equiv \mathbf{cdr}$

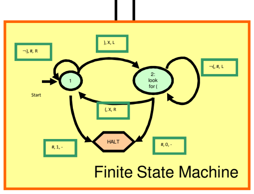
$\mathbf{succ} \equiv \lambda\ x\ .\ (\mathbf{cons}\ \mathbf{F}\ x)$

$\mathbf{pred} \equiv \lambda x.\ (\mathbf{cdr}\ x)$

42 = $\lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y$
$\lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z$
$xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.$
$y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y$
$\lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z$
$xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.$
$y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y$
$\lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z$
$xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.$
$y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y$
$\lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z$
$xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.$
$y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y\ \lambda xy.(\lambda z.z\ xy)\ \lambda xy.\ y$
$\lambda x.\mathrm{T}$

Lambda Calculus is a Universal Computer

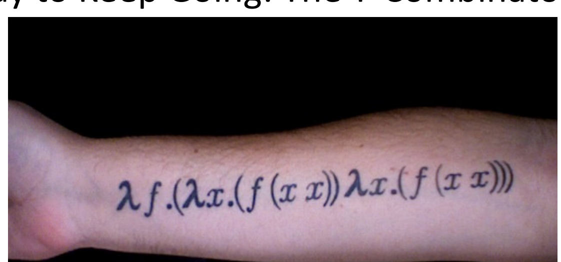




- Read/Write Infinite Tape
  - ✓ Mutable Lists
- Finite State Machine
  - ✓ Numbers to keep track of state
- Processing
  - ✓ Way of making decisions (if)
  - ☯ Way to keep going

We have this, but we cheated using ≡ to make recursive definitions!
Having names and definitions is much more complex than Lambda calculus.

Way to Keep Going: The Y-Combinator



$\mathbf{Y} = \lambda f\,(\lambda x\ .\ (f\,(x\ x))\ (\lambda x\ .\ f\,(x\ x))$
This finds the fixed point of any function!
$(\mathbf{Y}\ G) = (G\ (\mathbf{Y}\ G))$

Universal Computer

- Lambda Calculus can simulate a Turing Machine
  - Everything a Turing Machine can compute, Lambda Calculus can compute also
- Turing Machine can simulate Lambda Calculus (we didn't prove this)
  - Everything Lambda Calculus can compute, a Turing Machine can compute also
- Church-Turing Thesis: this is true for any other mechanical computer also

Computability in Theory and Practice

(Intellectual Computability Discussion on TV Video)

http://video.google.com/videoplay?docid=1623254076490030585#

http://www.funny-videos.co.uk/videoAliGScienceVideo39.html

Ali G Problem

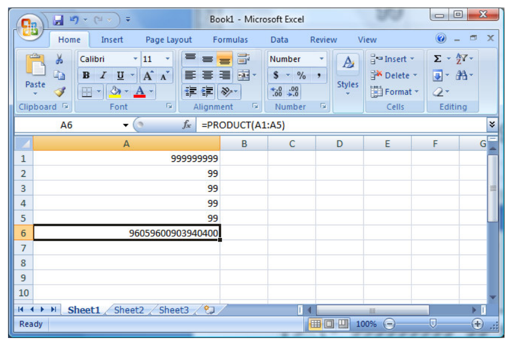
Input: a list of 2 numbers with up to d digits each

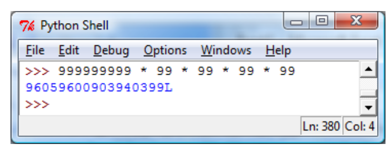
Output: the product of the 2 numbers

Is it computable?

Yes – a straightforward algorithm solves it. Using elementary multiplication techniques we know it is in $O(d^2)$

Can *real* computers solve it?





Ali G was Right!

- Theory assumes **ideal** computers:
  - Unlimited, perfect memory
  - Unlimited (but finite) time
- Real computers have:
  - Limited memory, time, power outages, flaky programming languages, etc.
  - There are many computable problems we cannot solve with real computer: the actual inputs ***do*** matter (in practice, but not in theory!)

Things Real Computers Can Do That Turing Machines Cannot

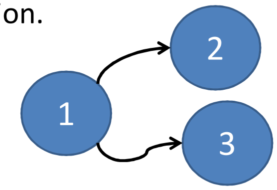
Generate Heat

Provide an adequate habitat for fish

Stop a Door

Nondeterministic Turing Machine

- At each step, instead of making one choice and following it, the machine can simultaneously try two choices.
- If any path of choices leads to a halting state, that machine's state is the result of the computation.



Ways to Think about Nondeterminism

Omnipotent: It can try all possible solutions at once to find the one that is right.

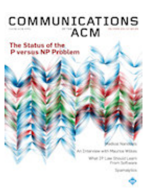
Omniscient: Whenever it has to make a choice, it always guess right.

Can a regular TM model a nondeterministic TM?

Yes, just simulate all the possible machines.

Can a nondeterministic TM solve problems in polynomial time ($O(N^k)$ for some constant k) that cannot be solved in polynomial time by a regular TM?

Answer: Unknown! This is the most famous and important open question in Computer Science: P = NP?



Ways to answer this:

1. Write a polynomial time pegboard puzzle solver (or prove it can't be done)
2. Write a polynomial time optimal photomosaic maker (or prove it can't be done)
3. ...

Course Summary: Three Main Themes

Recursive Definitions

Recursive procedures, recursive data structures, languages

Universality

Procedures are just another kind of data

A universal computing machine can simulate all other computing machines

Abstraction: giving things names and hiding details

Digital abstraction, procedural abstraction, data abstraction, objects

Things that are likely to be on the Final

Defining Procedures
- How to define procedures to solve problems, recursive procedures
- Functional and imperative style programming

Analyzing Procedures
- Asymptotic run-time analysis, memory use

Interpreters
- Understanding how interpreter defines meaning and running time of a language
- Being able to change a language by modifying an interpreter

Computing Models
- Proving a problem is computable or noncomputable
- Is a computing model equivalent to a TM?

NYTimes article today that mentions my 2005 crypto final!

Charge

- Sunday (4:59pm): to qualify for a presentation, you must have some basic functionality working
- Monday: Project Presentations
  - or...Project Reports (for non-presenting teams)
  - Presentation time will be divided among the qualifying teams (if all teams qualify, less than 2 minutes!): time to explain your project and demo its most interesting functionality
- Final Exam: will be posted Monday

I will have extended extra office hours (either in my office or Small Hall) on Sunday afternoon, 1:30-5pm (groups that upload projects by Saturday will have priority)