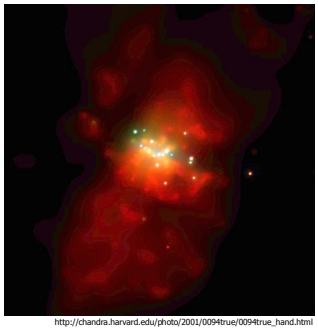
Class 28: The Meaning of Truth



http://chandra.harvard.edu/photo/2001/0094true/0094true_hand.html

CS150: Computer Science
University of Virginia
Computer Science

David Evans
http://www.cs.virginia.edu/evans

λ-calculus

Alonzo Church, 1940

(LISP was developed from λ-calculus, not the other way round.)

$$\begin{aligned} term = \; & variable \\ | \; & term\ term \\ | \; & (term) \\ | \; & \lambda\ variable\ .\ term \end{aligned}$$

Reduction (Uninteresting Rules)

$\lambda y.\ M \rightarrow \lambda v.\ (M\ [y \hookleftarrow v])$

where v does not occur in M.

$M \rightarrow M$

$M \rightarrow N \Rightarrow PM \rightarrow PN$

$M \rightarrow N \Rightarrow MP \rightarrow NP$

$M \rightarrow N \Rightarrow \lambda x.\ M \rightarrow \lambda x.\ N$

$M \rightarrow N$ and $N \rightarrow P \Rightarrow M \rightarrow P$

β-Reduction (the source of all computation)

$$(\lambda x.\ M)N \rightarrow M\ [\ x \hookleftarrow N\ ]$$

Evaluating Lambda Expressions

- *redex*: Term of the form $(\lambda x.\ M)N$

  Something that can be β-reduced
- An expression is in *normal form* if it contains no redexes (*redices*).
- To evaluate a lambda expression, keep doing reductions until you get to normal form.

Recall Apply in Scheme

"To **apply** a procedure to a list of arguments, **eval**uate the procedure in a new environment that binds the formal parameters of the procedure to the arguments it is applied to."

- We've replaced environments with substitution.
- We've replaced **eval** with reduction.

Some Simple Functions

$\mathbf{I} \equiv \lambda x.x$

$\mathbf{C} \equiv \lambda xy.yx$

Abbreviation for $\lambda x.(\lambda y.\ yx)$

$\mathbf{CII} = (\lambda x.(\lambda y.\ yx))\ (\lambda x.x)\ (\lambda x.x)$

$\rightarrow_\beta (\lambda y.\ y\ (\lambda x.x))\ (\lambda x.x)$

$\rightarrow_\beta \lambda x.x\ (\lambda x.x)$

$\rightarrow_\beta \lambda x.x$

$= \mathbf{I}$

Example

$$\lambda f.\ ((\lambda\ x.f\ (xx))\ (\lambda\ x.\ f\ (xx)))$$

Try this one at home...

Alyssa P. Hacker's Answer

$(\lambda f.\ ((\lambda\ x.f\ (xx))\ (\lambda\ x.\ f\ (xx))))\ (\lambda z.z)$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda\ x.\ (\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda z.z)\ (\lambda\ x.(\lambda z.z)(xx))\ (\lambda\ x.(\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda\ x.(\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda z.z)\ (\lambda\ x.(\lambda z.z)(xx))\ (\lambda\ x.(\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda\ x.(\lambda z.z)(xx))$

$\rightarrow_\beta$ ...

Ben Bitdiddle's Answer

$(\lambda f.\ ((\lambda\ x.f\ (xx))\ (\lambda\ x.\ f\ (xx))))\ (\lambda z.z)$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda\ x.\ (\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda x.xx)\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta (\lambda x.xx)\ (\lambda x.xx)$

$\rightarrow_\beta (\lambda x.xx)\ (\lambda x.xx)$

$\rightarrow_\beta$ ...

Be Very Afraid!

- Some λ-calculus terms can be β-reduced forever!
- The order in which you choose to do the reductions might change the result!

Take on Faith (until CS655)

- All ways of choosing reductions that reduce a lambda expression to normal form will produce the same normal form (but some might never produce a normal form).
- If we always *apply the outermost lambda first*, we will find the normal form if there is one.
  - This is *normal order reduction* – corresponds to normal order (lazy) evaluation

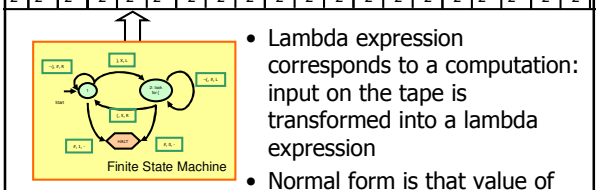
Universal Language

- Is Lambda Calculus a *universal language*?
  - Can we compute any computable algorithm using Lambda Calculus?
- To prove it isn't:
  - Find some Turing Machine that cannot be simulated with Lambda Calculus
- To prove it is:
  - Show you can simulate *every* Turing Machine using Lambda Calculus

Simulating Every Turing Machine

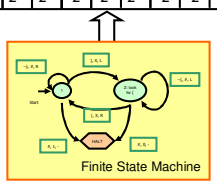
- A Universal Turing Machine can simulate every Turing Machine
- So, to show Lambda Calculus can simulate every Turing Machine, all we need to do is show it can simulate a Universal Turing Machine

Simulating Computation



- Lambda expression corresponds to a computation: input on the tape is transformed into a lambda expression
- Normal form is that value of that computation: output is the normal form
- How do we simulate the FSM?

Simulating Computation



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

Making "Primitives" from Only Glue (λ)

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 $M\,N \rightarrow M$

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

$$\mathbf{T} \equiv \lambda x\ (\lambda y.\ x)$$
$$\equiv \lambda xy.\ x$$
$$\mathbf{F} \equiv \lambda x\ (\lambda\ y.\ y))$$
$$\mathbf{if} \equiv \lambda pca\ .\ pca$$

Finding the Truth

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

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

$\mathbf{if} \equiv \lambda p\ .\ (\lambda c\ .\ (\lambda a\ .\ pca)))$ Is the **if** necessary?

$\mathbf{if}\ \mathbf{T}\ \boldsymbol{M}\ \boldsymbol{N}$

$((\lambda pca\ .\ pca)\ (\lambda xy.\ x))\ \boldsymbol{M}\ \boldsymbol{N}$

$\rightarrow_\beta (\lambda ca\ .\ (\lambda x.(\lambda y.\ x))\ ca))\ \boldsymbol{M}\ \boldsymbol{N}$

$\rightarrow_\beta \rightarrow_\beta (\lambda x.(\lambda y.\ x))\ \boldsymbol{M}\ \boldsymbol{N}$

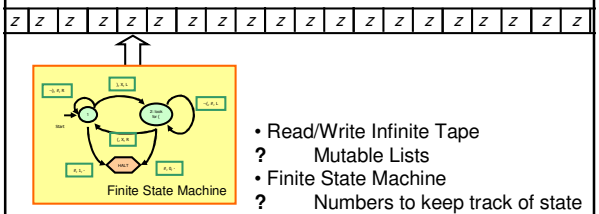
$\rightarrow_\beta (\lambda y.\ \boldsymbol{M}))\ \boldsymbol{N} \rightarrow_\beta \boldsymbol{M}$

and and **or**?

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

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

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

Charge

- PS6 Due Monday
- Start thinking about web application ideas for PS8
  - If you find a team and idea quickly (by Nov 2), you can negotiate what you need to do for PS7
- Week after: we will finish this proof
  - How to make the other things we need to simulate a Turing machine using only Lambda Calculus