A Universal Language

One-Slide Summary

- The **lambda calculus** is a universal, fundamental model of computation. You can view it as "the essence of Scheme". It contains terms and rules describing variables, function abstraction, and function application.
- There are two key reduction rules in the lambda calculus. Alpha reduction allows you to rename variables uniformly. **Beta reduction** is the essence of computation: in beta reduction, a function evaluation is equivalent to replacing all instances of the formal parameter in the function body with the actual argument.
- It is possible to **encode** programming concepts, such as true, false, if, numbers, plus, etc., in the lambda calculus.

λ-calculus

Alonzo Church, 1940

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

term = *variable*

| *term term*

| **λ** *variable* **.** *term*

What is Calculus?

- In High School:

$d/dx\ x^n = nx^{n-1}$ [Power Rule]

$d/dx\ (f + g) = d/dx\ f + d/dx\ g$ [Sum Rule]

Calculus is a branch of mathematics that deals with limits and the differentiation and integration of functions of one or more variables...

Real Definition

- A **calculus** is just a bunch of rules for manipulating symbols.
  - Latin word calx meaning pebble ...
- People can give meaning to those symbols, but that's not part of the calculus.
- Differential calculus is a bunch of rules for manipulating symbols. There is an interpretation of those symbols corresponds with physics, slopes, etc.

Lambda Calculus

- Rules for manipulating strings of symbols in the language:

  term = *variable*

  | *term term*

  | **λ** *variable* **.** *term*
- Humans can give meaning to those symbols in a way that corresponds to computations.

Why?

- Once we have precise and formal rules for manipulating symbols, we can reason with those symbols and rules.
- Since we can interpret the symbols as representing computations, we can use this system to reason about programs.

Evaluation Rules

α-reduction (renaming)

$$\lambda y.\, M \Rightarrow_{\alpha} \lambda v.\, (M \text{ [each } y \text{ replaced by } v])$$

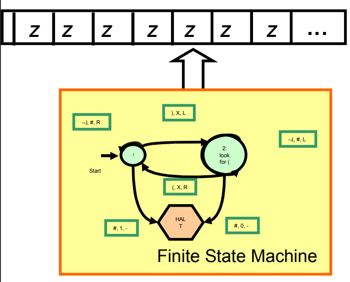
where v does not occur in M.

β-reduction (substitution)

$$(\lambda x.\, M)N \Rightarrow_{\beta} M \text{ [each } x \text{ replaced by } N \text{]}$$

We'll see examples in a bit!

Equivalent Computers?



Turing Machine

≡

$term =\ variable$
$\quad |\ term\ term$
$\quad |\ (term)$
$\quad |\ \lambda\ variable\ .\ term$

$$\lambda y.\, M \Rightarrow_{\alpha} \lambda v.\, (M\,[y \to v])$$

where v does not occur in M.

$$(\lambda x.\, M)N \Rightarrow_{\beta} M\,[\,x \to N\,]$$

Lambda Calculus

Liberal Arts Trivia: Music

- This music genre originated in Jamaica in the 1950s and was the precursor to reggae. It combines elements of Caribbean mento and calypso with American jazz and rhythm and blues. It is characterized by a walking abss line accented with rhythms on the offbeat. In the 1980s it experience a third wave revival and is often associated with punk and brass instruments.

Liberal Arts Trivia: Geography

- This baltic country borders Romania, Serbia, Macedonia, Greece, Turkey and the Black Sea. It was at one point ruled by the Ottomans, but is now a member of the EU and NATO. Sofia, the capital and largest city, is one of the oldest cities in Europe and can be traced back some 7000 years. The traditional cuisine of this country features rich salads at every meal, as well as native pastries such as the *banitsa*.

Lambda Examples

- Identity Function
  - (define identity (lambda (x) x))
  - **identity** = λ x. x
- Square Function
  - (define square (lambda (x) (* x x))
  - **square** = λ x. (* x x)
- Add Function
  - (define (add x y) (+ x y))
  - (define add (lambda (x) (lambda (y) (+ x y))))
  - **add** = λ x. λ y. (+ x y)

β-Reduction (the source of all computation)

$$(\lambda x.\ M)N \Rightarrow_\beta M[\,x \rightarrow N\,]$$

Replace all x's in M with N's

Note the syntax is different from Scheme:
$(\lambda x.M)N$ === ((lambda (x) M) N)

β-Reduction Examples

Recall: $(\lambda x.\ M)N \Rightarrow_\beta M[\,x \rightarrow N\,]$

- Square Function
  - square = λ x. (* x x)
  - (λ x. (* x x)) 5
  - (λ x. (* x x)) 5 $\Rightarrow_\beta$ (* x x)[x→5]
  - (λ x. (* x x)) 5 $\Rightarrow_\beta$ (* x x)[x→5] $\Rightarrow_\beta$ (* 5 5)
- Add Function
  - add = λ x. λ y. (+ x y)
  - (λ x. λ y. (+ x y)) 3 $\Rightarrow_\beta$???
  - ((λ x. λ y. (+ x y)) 2) 6 $\Rightarrow_\beta$???

Get out some paper!

β-Reduction Examples

Recall: $(\lambda x.\ M)N \Rightarrow_\beta M[\,x \rightarrow N\,]$

- Square Function
  - square = λ x. (* x x)
  - (λ x. (* x x)) 5
  - (λ x. (* x x)) 5 $\Rightarrow_\beta$ (* x x)[x→5]
  - (λ x. (* x x)) 5 $\Rightarrow_\beta$ (* x x)[x→5] $\Rightarrow_\beta$ (* 5 5)
- Add Function
  - add = λ x. λ y. (+ x y)
  - (λ x. λ y. (+ x y)) 3 $\Rightarrow_\beta$ **λ y. (+ 3 y)**
  - ((λ x. λ y. (+ x y)) 2) 6 $\Rightarrow_\beta$ (λ y. (+ 2 y)) 6 $\Rightarrow_\beta$ **(+ 2 6)**

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*.

Some Simple Functions

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

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

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

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

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

Example

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

Do it on paper!

Possible Answer

$$(\lambda f.\ ((\lambda x.f(xx))\ (\lambda x.\ f(xx))))\ (\lambda z.z)$$
$$\to_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda x.\ (\lambda z.z)(xx))$$
$$\to_\beta (\lambda z.z)\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$$
$$\to_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$$
$$\to_\beta (\lambda z.z)\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$$
$$\to_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$$
$$\to_\beta \ldots$$

Alternate Answer

$$(\lambda f.\ ((\lambda x.f(xx))\ (\lambda x.\ f(xx))))\ (\lambda z.z)$$
$$\to_\beta (\lambda x.(\lambda z.z)(xx))\ (\lambda x.\ (\lambda z.z)(xx))$$
$$\to_\beta (\lambda x.xx)\ (\lambda x.(\lambda z.z)(xx))$$
$$\to_\beta (\lambda x.xx)\ (\lambda x.xx)$$
$$\to_\beta (\lambda x.xx)\ (\lambda x.xx)$$
$$\to_\beta \ldots$$

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!

Liberal Arts Trivia: Classics

- The Temple of Artemis at Ephesus, the Statue of Zeus at Olympus, and the Mausoleum of Maussollos are three of the **Seven Wonders of the Ancient World**. Name the other four.

Liberal Arts Trivia: British Lit

- This 1883 coming-of-age tale of "pirates and buried gold" by Robert Louis Stevenson had a vast influence on the popular perception of pirates. Its legacies include treasure maps with an "X", the Black Spot, tropical islands, and one-legged seamen with parrots on their shoulders.
  - Name the book.
  - Name the morally gray, parrot-holding mutineer.

Take on Faith (until Grad PL)

- 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 is **not**:
  - 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

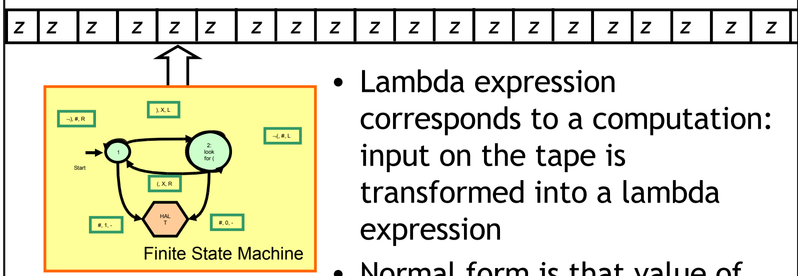
Universal Language

- Is Lambda Calculus a ***universal language***?
  - Can we compute any computable algorithm using Lambda Calculus?
- To prove it is **not**:
  - 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

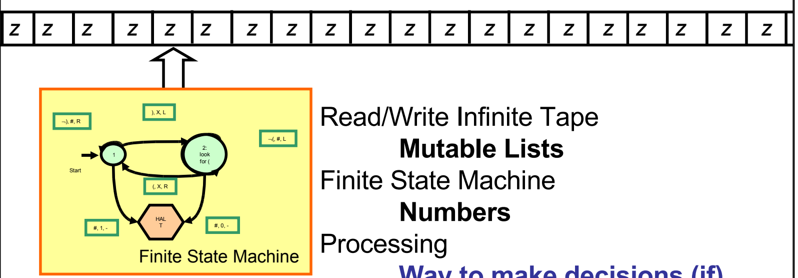
- A **Universal** Turing Machine can simulate every Turing Machine
- So, to show Lambda Calculus can simulate every Turing Machine, all we need to do is show it can simulate a Universal Turing Machine!

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
Processing
Way to make 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* → *M*

Don't search for T, search for if

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

$$\equiv \lambda xy.\ x$$

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

$$\mathbf{if} \equiv \lambda pca\ .\ pca$$

The Truth Is Out There

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

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

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

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

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

$$\rightarrow_\beta ???$$

Finding the Truth

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

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

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

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

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

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

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

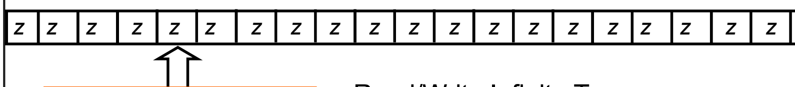
$$\rightarrow_\beta (\lambda y.\ \boldsymbol{M}\ ))\ \boldsymbol{N} \rightarrow_\beta \boldsymbol{M}$$

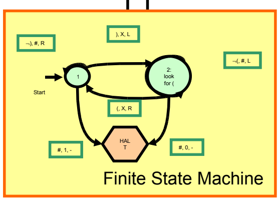
and and or?

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

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

Lambda Calculus is a Universal Computer?





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

...to be continued ...

Homework

- PS 9 Presentation Requests due Mon Apr 27