# $\lambda$ Calculus: Lecture 3

Madhavan Mukund

Chennai Mathematical Institute

madhavan@cmi.ac.in

PLC, 13 March 2017

# Recursive functions

Recursive functions [Gödel]

Initial functions

- Zero: $Z(n) = 0$.
- Successor: $S(n) = n+1$.
- Projection: $\Pi_i^k(n_1, n_2, \ldots, n_k) = n_i$

Composition Given $f : \mathbb{N}^k \to \mathbb{N}$ and
$g_1, g_2, \ldots, g_k : \mathbb{N}^h \to \mathbb{N}$,

$$f \circ (g_1, g_2, \ldots, g_k)(n_1, n_2, \ldots, n_h) =$$
$$f(g_1(n_1, n_2, \ldots, n_h), g_2(n_1, n_2, \ldots, n_h), \ldots, g_k(n_1, n_2, \ldots, n_h))$$

# Recursive functions . . .

Primitive recursion Given $g : \mathbb{N}^k \to \mathbb{N}$ and
$h : \mathbb{N}^{k+2} \to \mathbb{N}$

define $f : \mathbb{N}^{k+1} \to \mathbb{N}$ by primitive recursion as follows:

$$\begin{aligned} f(0, n_1, n_2, \ldots, n_k) &= g(n_1, n_2, \ldots, n_k) \\ f(n{+}1, n_1, \ldots, n_k) &= h(n, f(n, n_1, n_2, \ldots, n_k), n_1, \ldots, n_k) \end{aligned}$$

Minimalization
Given $g : \mathbb{N}^{k+1} \to \mathbb{N}$, define $f : \mathbb{N}^k \to \mathbb{N}$ by minimalization from $g$

$$f(n_1, n_2, \ldots, n_k) = \mu n.(g(n, n_1, n_2 \ldots, n_k) = 0)$$

where $\mu n.P(n)$ returns the least natural number $n$ such that $P(n)$ holds

# Encoding recursive functions . . .

- $\langle n \rangle \equiv \lambda fx.(f^n x)$.
- Successor $\langle succ \rangle \equiv \lambda nfx.(f(nfx))$ such that $succ\langle n \rangle \rightarrow^* \langle n+1 \rangle$.
- Zero $\langle Z \rangle \equiv \lambda x.(\lambda gy.y)$.
- Projection $\langle \Pi_i^k \rangle \equiv \lambda x_1 x_2 \ldots x_k.x_i$.

Composition is easy

# Recursive functions: Primitive recursion

- Evaluate $t(n)$ bottom up
  - Much like dynamic programming for recursive functions
- Define a function *step* that does the following

$$step(n, f(n)) = (n{+}1, f(n{+}1))$$

  i.e.

$$step(t(n)) = t(n{+}1)$$

- So, $t(n) = step^n(0, f(0)) = step^n(0, g)$ ...
- ... and $f(n) = snd(t(n)) = snd(step^n(0, g))$
- Requires constructions for building pairs and decomposing them using *fst* and *snd*

# Recursive functions: Minimalization

- To evaluate

$$f(n_1, n_2, \ldots, n_k) = \mu n.(g(n, n_1, n_2 \ldots, n_k) = 0)$$

  we go back to the idea of computing a while loop

```
n := 0;
while (g(n,n1,n2,...,nk) != 0) {n := n+1};
return n;
```

- Implement the while loop using recursion

```
f(n1,n2,...,nk) =   check(0,n1,n2..nk)
where
  check(n,n1,n2..nk){
    if (iszero(g(n,n1,n2,...,nk))  {return n;}
    else  {check(n+1,n1,n2,...,nk);}
  }
```

- Need a mechanism to encode booleans, if-then-else in $\lambda$-calculus. Also need a mechanism for recursion.

# Recursive definitions

Suppose $F = \lambda x_1 x_2 \ldots x_n E$, where where $E$ contains an occurrence of $F$

- Choose a new variable $f$
- Convert $E$ to $E^*$ replacing every $F$ in $E$ by $ff$
  - If $E$ is of the form $\cdots F \cdots F \cdots$ then $E^*$ is $\cdots (ff) \cdots (ff) \cdots$.

Now write

$$\begin{aligned} G &= \lambda f x_1 x_2 \ldots x_n . E^* \\ &= \lambda f x_1 x_2 \ldots x_n . \cdots (ff) \cdots (ff) \cdots \end{aligned}$$

Then

$$GG = \lambda x_1 x_2 \ldots x_n . \cdots (GG) \cdots (GG) \cdots$$

- $GG$ satisfies the equation defining $F$
- Write $F = GG$, where $G = \lambda f x_1 x_2 \ldots x_n . E^*$.

# Minimalization . . .

```
f(n1,n2,...,nk) =   check(0,n1,n2...nk)
where
  check(n,n1,n2...nk){
    if (iszero(g(n,n1,n2,...,nk))  {return n;}
    else  {check(n+1,n1,n2,...,nk);}
  }
```

Encoding Booleans

- $\langle True \rangle \equiv \lambda xy.x$
- $\langle False \rangle \equiv \lambda xy.y$
- $\langle if - b - then - x - else - y \rangle \equiv \lambda bxy.\ bxy$

# Minimalization . . .

- $\langle True \rangle \equiv \lambda xy.x$
- $\langle False \rangle \equiv \lambda xy.y$
- $\langle if - b - then - x - else - y \rangle \equiv \lambda bxy.\ bxy$

$$
\begin{aligned}
(\lambda bxy.bxy)\langle True\rangle fg &\rightarrow \lambda xy.((\lambda xy.x)xy)fg \\
&\rightarrow \lambda y.((\lambda xy.x)fy)g \\
&\rightarrow (\lambda xy.x)fg \\
&\rightarrow (\lambda y.f)g \\
&\rightarrow f
\end{aligned}
$$

$$
\begin{aligned}
(\lambda bxy.bxy)\langle False\rangle fg &\rightarrow \lambda xy.((\lambda xy.y)xy)fg \\
&\rightarrow \lambda y.((\lambda xy.y)fy)g \\
&\rightarrow (\lambda xy.y)fg \\
&\rightarrow (\lambda y.y)g \\
&\rightarrow g
\end{aligned}
$$

# Minimalization . . .

- Want to define $f : \mathbb{N}^k \to \mathbb{N}$ by minimalization from a $g : \mathbb{N}^{k+1} \to \mathbb{N}$
- Already have an encoding $\langle g \rangle$ for $g$

Define $F$ as follows:

$$F = \lambda n x_1 x_2 \ldots x_k.\ \begin{array}{l} \text{if } \langle \mathit{iszero} \rangle (\langle g \rangle\ n\ x_1\ x_2\ \ldots\ x_k) \\ \text{then } n \\ \text{else } F(\langle \mathit{succ} \rangle n)\ x_1\ x_2\ \ldots\ x_k \end{array}$$

- $\tilde{F}$ : the lambda term for $F$ after unravelling the recursive definition
- $\langle f \rangle$ is then $\tilde{F}\ \langle 0 \rangle$.

Still need to define $\langle \mathit{iszero} \rangle$

# Minimalization . . .

$$\langle iszero \rangle = \lambda n.n(\lambda z.\langle false \rangle)\langle true \rangle$$

$$\begin{array}{rcl} \langle iszero \rangle\ \langle 0 \rangle & = & (\lambda n.n(\lambda z.\langle false \rangle)\langle true \rangle)\ (\lambda fx.x) \\ & \rightarrow_\beta & (\lambda fx.x)(\lambda z.\langle false \rangle)\langle true \rangle \\ & \rightarrow_\beta & (\lambda x.x)\langle true \rangle \\ & \rightarrow_\beta & \langle true \rangle \end{array}$$

$$\begin{array}{rcl} \langle iszero \rangle\ \langle 1 \rangle & = & (\lambda n.n(\lambda z.\langle false \rangle)\langle true \rangle)\ (\lambda fx.fx) \\ & \rightarrow_\beta & (\lambda fx.fx)(\lambda z.\langle false \rangle)\langle true \rangle \\ & \rightarrow_\beta & (\lambda x.(\lambda z.\langle false \rangle)\ x)\langle true \rangle \\ & \rightarrow_\beta & (\lambda z.\langle false \rangle)\ \langle true \rangle \\ & \rightarrow_\beta & \langle false \rangle \end{array}$$

By induction, for $n > 0$ . . .

$$\langle iszero \rangle\ \langle n \rangle \rightarrow_\beta^* (\lambda z.\langle false \rangle)^n\ \langle true \rangle \rightarrow_\beta^* \langle false \rangle$$

# One step reduction

- Can have other reduction rules like $\beta$
- Observe that $\lambda x.(Mx)$ and $M$ are equivalent with respect to $\beta$-reduction, provided $x$ is not free in $M$.
- New reduction rule $\eta$

$$\lambda x.(Mx) \to_\eta M$$

- Given basic rules $\beta$, $\eta$, ..., we are allowed to use them "in any context"
- Define a one step reduction relation $\to$ inductively

$$\frac{M \to_x M'}{M \to M} \quad x \in \{\beta, \eta, \ldots\} \qquad \frac{M \to M'}{\lambda x.M \to \lambda x.M'} \qquad \frac{M \to M'}{MN \to M'N} \qquad \frac{N \to N'}{MN \to MN'}$$

# Normal forms

- Computation — a maximal sequence of reduction steps
- "Values" are expressions that cannot be further reduced: normal forms
- Allow reduction in any context $\Rightarrow$ multiple expressions may qualify for reduction in one step

Natural questions

- Does every term reduce to a normal form?
- Can a term reduce to more than one normal form, depending on order reduction strategy?
- If a term has a normal form, can we always find it?

# Normal forms . . .

Does every term reduce to a normal form?

- Consider $(\lambda x.xx)(\lambda x.xx)$
- $(\lambda x.xx)(\lambda x.xx) \rightarrow_\beta (\lambda x.xx)(\lambda x.xx)$
  - Reduction never terminates
- Call this term $\Omega$

# Normal forms . . .

Can a term reduce to more than one normal form, depending on order reduction strategy?

- Consider $\langle \mathit{False} \rangle \Omega = (\lambda yz.z)((\lambda x.xx)(\lambda x.xx))$
- Outermost reduction:
  $(\lambda yz.z)((\lambda x.xx)(\lambda x.xx)) \rightarrow \lambda z.z$
- Innermost reduction:
  $(\lambda yz.z)((\lambda x.xx)(\lambda x.xx)) \rightarrow (\lambda yz.z)((\lambda x.xx)(\lambda x.xx)) \rightarrow \cdots$
- Choice of reduction strategies may determine whether a normal form is reached . . .
- . . . but the question is, can more than one normal form be reached?

# Normal forms . . .

If a term has a normal form, can we always find it?

- We have seen how to encode recursive functions in $\lambda$-calculus
- Given a recursive function $f$ and an argument $n$, we cannot determine, in general, if computation of $f(n)$ terminates
- Computing $f(n)$ is equivalent to asking if $\langle f \rangle \langle n \rangle$ achieves a normal form

# Normal forms . . .

Can a term reduce to more than one normal form, depending on order reduction strategy?

- Define an equivalence relation $\leftrightarrow$ on $\lambda$-terms

$$M \leftrightarrow N \text{ iff } \exists P.\ P \rightarrow^* M, P \rightarrow^* N$$

$M \leftrightarrow N$ if both $M$ and $N$ can be obtained by reduction from a common "ancestor" $P$

- $\leftrightarrow$ is the symmetric transitive closure of $\rightarrow^*$

$$\frac{M \rightarrow^* N}{M \leftrightarrow N} \quad \frac{M \leftrightarrow N}{N \leftrightarrow M} \quad \frac{M \leftrightarrow N, N \leftrightarrow P}{M \leftrightarrow P}$$

- In general, for any reflexive, transitive relation $R$, can define the symmetric, transitive closure $\overset{R}{\leftrightarrow}$

# Church-Rosser Theorem

Diamond property or Church-Rosser property

- Let $R$ be any reflexive, transitive relation (such as $\rightarrow^*$)
- $R$ has the diamond property if, whenever $X\ R\ Y$ and $X\ R\ Z$ there is $W$ such that $Y\ R\ W$ and $Z\ R\ W$

Theorem [Church-Rosser]

*Let $R$ be Church-Rosser. Then $M \overset{R}{\leftrightarrow} N$ implies there exists $Z$, $M\ R\ Z$ and $N\ R\ Z$*

Proof By induction on the definition of $\overset{R}{\leftrightarrow}$

# Church-Rosser Theorem

Corollary [Church-Rosser]

*Let $R$ be a reduction relation that is Church-Rosser. Then a term can have at most one normal form with respect to $R$*

Proof By picture

# Church-Rosser Theorem

Is $\rightarrow^*$ Church-Rosser?

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

Two possible reductions

- $(\lambda x.xx)((\lambda x.x)(\lambda x.x)) \rightarrow$
  $((\lambda x.x)(\lambda x.x))((\lambda x.x)(\lambda x.x))$ (Outermost)
- $(\lambda x.xx)((\lambda x.x)(\lambda x.x)) \rightarrow ((\lambda x.xx)(\lambda x.x)$ (Innermost)

From second option, in one step we get

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

Can reach this term from the first option as well, but it requires two steps!

# Church-Rosser Theorem

Solution: Define a new notion of one step reduction $\twoheadrightarrow$ such that

- This new reduction is Church-Rosser.
- Its reflexive, transitive closure is equal to $\rightarrow^*$.

Define $\twoheadrightarrow$ as follows.

$$M \twoheadrightarrow M \qquad \frac{M \twoheadrightarrow M'}{\lambda x.M \twoheadrightarrow \lambda x.M'}$$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{MN \twoheadrightarrow M'N'} \qquad \frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{(\lambda x.M)N \twoheadrightarrow M'\{x \leftarrow N'\}}$$

- $\twoheadrightarrow$ combines nonoverlapping $\rightarrow$ reductions into one parallel step