Lambda Calculus

Rewriting Calculi

- Rewriting
- Lambda calculus
- Recursion theorem
- Combinatory logic
- Typed calculus

Lambda Terms

- Variables: x y z…
- Abstractions (function creation): $\lambda x.M$
- Applications: MN

- Shorthand:
  - $\lambda y,z.M$ for $\lambda y.\lambda z.M$
  - F(M,N) for (F(M))(N)

Beta

Apply an abstraction to a term

- $(\lambda x.M)N \to M[x \longmapsto N]$
- $(\lambda x.M[x,x,\ldots,x])N \to M[N,N,\ldots,N]$
    - replace all free occurrences of x in M with N

Conditionals as λ Expressions

Define

$T := \lambda y,z.y$

$F := \lambda y,z.z$

if X then y else z $:=$ $X(y,z)$

Church Numerals (Standard)

Defined inductively

I	:=	$\lambda x.x$
0	:=	$\lambda f.I$
n+1	:=	$\lambda f.\lambda x.f((nf)x)$

Numbers as λ Expressions
(Object-Oriented version)

Define

0	:=	$\lambda x.x$	
n+1	:=	$\lambda x.x(F,n)$	[inductive defn.]
s	:=	$\lambda n.\lambda x.x(F,n)$	[successor]
p	:=	$\lambda n.n(F)$	[predecessor]
z	:=	$\lambda n.n(T)$	[test for 0]

Note: p0 = F

Recursion via λ Expressions

Instead of

$f(x) := A[f(b)]$

use

$f(x) := E(E,a)$ *where* $E := \lambda zx.A[z(z,b)]$

Y Combinator

Let

$Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$

Then

Yg

is the fixpoint of g

which is the desired recursive function

Factorial

- A := Y λa,m,n.z(n)(m,s(a(m,p(n))))
- M := Y λb,m,n.z(n)(0,A(m,b(m,p(n))))
- ! := Y λf,n.z(n)(s0,M(n,z(p(n))))

Example (normal order evaluation)

Is the predecessor of 2 equal 0?

$zp2 \Rightarrow (p2)(T)$

$\Rightarrow 2(F)(T) = (\lambda x.x(F,1))(F)(T)$

$\Rightarrow F(F,1)(T) = (\lambda y,z.z)(F,1)(T)$

$\Rightarrow 1(T) = (\lambda x.x(F,0))(T)$

$\Rightarrow T(F,0) = (\lambda y,z.y)(F,0)$

$\Rightarrow F$

(Applicative Order)

Define

T	:=	$\lambda y,z.\ y(\ )$
F	:=	$\lambda y,z.\ z(\ )$
if X then y else z	:=	$X(\lambda.y,\ \lambda.z)$