Lambda terms are built up inductively in:

Variables - x, y, z

Applications - of form s t where s t are lambda terms f x

Abstractions - of form G.x.s

Free variables are a term for a variable which is not bound to any corresponding formal parameter

The free variables of a lambda terms denoted FV(t) can be defined recursively as follows:

FV(x) = {x}

FV(s t) = FV(s) U FV(t)

FV(Gx.s) = FV(s)\{x}

A term is closed if it does not contain any free variables

REDUCTION RULES

A conversion

B reduction

N conversion

Lambda Equality

We say that two terms s & t are equal if there is a finite sequence of a b n reductions which connects them

Congruence symbol is identity

Basic evaluation strategy in lambda calculus is reduction

Called a redex

An evaluation is completed if the expression cannot be redexed further ergo normal form

Strategy 1: Applicative Order Reduction

Call-by-value evaluation leftmost innermost index

Strategy 2: Normal order Reduction

Call-by-name evaluation leftmost outermost index

Church Rosser Theorem

If s → t1 and s → t2 then there is a term u such that t1 → u and t2 -->u then there is a term u such that t1 → u and t2 → u

We get the stronger form that t1 = t2 then there is a term u with t1 = u and t2 =u

So if a term reduces to, or in general is equal to, a term in normal form, that normal form is unique to ALPHA conversion

Lambda equality is non-trivial because two terms in normal form are not a-equivalent and unequal

If any reduction terminates then one arrived at by systematic reduction, the leftmost-outermost redexes will terminate