Untyped lambda calculus

Description of behaviour of functions on most basic level

Objects of set theory: sets only: $\{1 \cup (\in \setminus +) \cap \mathbb{R}\}$ is a valid expression
$= ?$ ... depends on implementation of $1, \in, +, \mathbb{R}$

Objects of lambda calculus: functions only

Basic operations within lambda calculus: Application and Abstraction

we use set theory as a tool to express meta-statements ("statements on" about λ-calculus)

Definition: The "set" Λ of all λ-terms (a set of "variables" (are also functions, most basic), $V = \{a, b, c, \dots\}$

inductive definition of Λ (these are all λ-terms)

(1) (Variable) If $u \in V$, then $u \in \Lambda$

(2) (Application) If M and $N \in \Lambda$, then $(MN) \in \Lambda$

(3) (Abstraction) If $u \in V$ and $M \in \Lambda$, then $(\lambda u.M) \in \Lambda$

meta-variables, "actual variables", represent arbitrary λ-terms

Short form: $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$ ... "abstract symbol" or a "grammar"
("or")

Examples: $\overset{(1)}{x}, \overset{(1)}{y}, \overset{(1)}{z}, \overset{(2)}{(xx)}, \overset{(2)}{(x(xz))}, \overset{(3)}{(\lambda x.(xz))}, \overset{(2)}{(y(\lambda x.(xz)))} \in \Lambda$

Notation: Elements of V: $a, b, c, x, x', x'', x_1, x_2, x_3 \dots$
Elements of Λ: $A, B, C, x, x', x'', x_1, x_2, x_3, \dots$

Definition: Syntactical identity: $\equiv$: $(xz) \equiv (xz)$ but $(xz) \not\equiv (xy)$ (where $x, y, z \in V$)
If $M \equiv N$ or $M \not\equiv N$ depends on M and N (where $M, N \in \Lambda$)

Definition: Multiset Sub of subterms of a λ-term: (identical elements can occur multiple times)

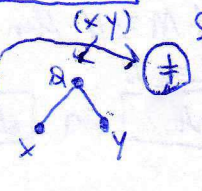
(1) (variable) $Sub(x) = \{x\}$ for all $x \in V$
(2) (Application) $Sub((MN)) = Sub(M) \cup Sub(N) \cup \{(M, N)\}$
(3) (Abstraction) $Sub((\lambda x.M)) = Sub(M) \cup \{(\lambda x.M)\}$

$L \in \Lambda$ is called a subterm of $M \in \Lambda$ if $L \in Sub(M)$
$L \in \Lambda$ is called a proper subterm of $M \in \Lambda$ if $L \in Sub(M)$ and $L \not\equiv M$

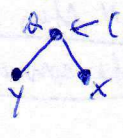
Lemma: (1) (Reflexivity) $\forall M \in \Lambda: M \in Sub(M)$
(2) (Transitivity) $\forall L, M, N \in \Lambda: L \in Sub(M) \wedge M \in Sub(N) \Rightarrow L \in Sub(N)$

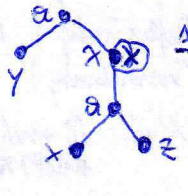
Example and "Tree" of subterms: $Sub((y(\lambda x.(xz)))) = \{(y(\lambda x.(xz))), y, (\lambda x.(xz)), x, z, x, z\}$

not a tree really a tree from graph theory because embedding is important



$Sub((xx)) = \{(xx), x, x\}$





subterms: nodes

Notation: • drop outermost parentheses: $MN \equiv (MN)$ ($\equiv$: extend definition of syntactical equality)

MN "stands for" (MN), $\boxed{MN = (MN)}$

• application is left associative: $\boxed{MNL \equiv ((MN)L)}$

• abstraction is right associative; use only one λ: $\boxed{\lambda xy.M = \lambda x.(\lambda y.M)}$

• application takes precedence over abstraction: $\boxed{\lambda x.MN = \lambda x.(MN)}$

Definition: free, and bound, binding, variables, set FV of free variables of λ-term

(1) (Variable) $FV(x) = \{x\}$, $(x \in V)$

(2) (Application) $FV(MN) = FV(M) \cup FV(N)$, $(M, N \in \Lambda)$

(3) (Abstraction) $FV(\lambda x.M) = FV(M) \setminus \{x\}$, $(M \in \Lambda, x \in V)$

x free in M if $x \in FV(M)$

x bound in M if $x \in B(x)$

x binding in M if $x \in Bi(x)$ (x occurs in M behind a λx)

Examples: $FV(\lambda x.xy) = FV(xy) \setminus \{x\} = (FV(x) \cup FV(y)) \setminus \{x\} = \{x,y\} \setminus \{x\} = \{y\}$

$FV(x(\lambda x.xy)) = FV(x) \cup FV(\lambda x.xy) = \{x\} \cup \{y\} = \{x,y\}$ ↕ free somewhere

Definition: (1) $B(x) = \{x\}$, (2) $B(MN) = B(M) \cup B(N)$, (3) $B(\lambda x.M) = B(M) \cup (\{x\}, FV(M))$

(1) $Bi(x) = \{x\}$, (2) $Bi(MN) = Bi(M) \cup Bi(N)$, (3) $Bi(\lambda x.M) = Bi(M) \cup \{x\}$

Definition: closed λ-term, combinator, set Λ^0 of closed λ-terms

$M \in \Lambda$ is called closed if $FV(M) = \emptyset$ or combinator

Definition: alpha conversion, renaming, $M^{x \to y}$, $=_\alpha$

For $M \in \Lambda$, $x, y \in V$ let $M^{x \to y} \in \Lambda$ denote the λ-term obtained by replacing every free occurrence of x in M by y: $x(\lambda x.xy)^{x \to y} \equiv y(\lambda x.xy)$

For $M \in \Lambda$, $x, y \in V$ with $y \notin FV(M)$ and y not a binding variable in M we define the relation renaming by $\lambda x.M \; [=_\alpha] \; \lambda y.M^{x \to y}$

$y \notin FV(M) \cup Bi(M)$ $\Leftrightarrow$ y does not occur in M

We say "$\lambda x.M$ has been renamed as $\lambda y.M^{x \to y}$"

Conditions on renaming: Should not change status of bound and free variables

If $y \in FV(M)$: $\lambda x.y^{x \to y}$ (y free) would become $\lambda y.y$ (y bound)

If y is a binding variable in M: $\lambda xy.x^{x \to y}$ would become $\lambda yy.y$ (bound to first λ / bound to second λ)

$\Leftrightarrow y \notin Bi(M)$

$((\lambda xy.x)5 \to_\beta \lambda y.5, (\lambda yy.y)5 \to_\beta \lambda y.y)$

↳ See examples 1.5.3

Alpha conversion: extend $=_\alpha$

(1) (Renaming) $\lambda x.M =_\alpha \lambda y.M^{x \to y}$ under above conditions ($y \notin FV(M)$, "$y \notin B(M)$")

β-conversion $\to$ (2) (Compatibility) If $M =_\alpha N$ then $ML =_\alpha NL$, $LM =_\alpha LN$, $\lambda z.M =_\alpha \lambda z.N$ for all $z \in V$

within subterms

(3a) (Reflexivity) $M =_\alpha M$, (3b) (Symmetry) If $M =_\alpha N$ then $N =_\alpha M$

(3c) (Transitivity) If $L =_\alpha M$ and $M =_\alpha N$ then $L =_\alpha N$

If $M =_\alpha N$ then M and N are said to be alpha convertible or alpha equivalent