

λ bezeichnet die hier Menge aller Lambdaterme

Applikation und Abstraktion

Applikation von λ -Terme: $t = t_1 t_2$ mit $t_1, t_2 \in \lambda$ Applikationen sind linksassoziativ: $t_1 t_2 t_3 = ((t_1 t_2) t_3)$

Abstraktion von λ -Terme: $t = (\lambda x. t_1)$ mit $t_1 \in \lambda$

Verschachtelte Abstraktionen können abgekürzt werden:

$$(\lambda x_1. (\lambda x_2. (\lambda x_3. x_1 x_2 x_3))) = (\lambda x_1 x_2 x_3. x_1 x_2 x_3)$$

gebundene Variablen GV und freie Variablen FV

Sei $t \in \lambda$:

- $t = x$ mit $x \in X$ (Menge der Variablen) $\Rightarrow GV(t) = \emptyset$ und $FV(t) = \{x\}$
- $t = t_1 t_2$ mit $t_1, t_2 \in \lambda \Rightarrow GV(t) = GV(t_1) \cup GV(t_2)$ und $FV(t) = FV(t_1) \cup FV(t_2)$
- $t = (\lambda x. t_1)$ mit $t_1 \in \lambda \Rightarrow GV(t) = \{x\} \cup GV(t_1)$ und $FV(t) = FV(t_1) / \{x\}$

Substitution

$t[x/s]$ mit $t, s \in \lambda$ und $x \in X$: Substitution aller frei Vorkommen von x in t durch s

Bsp.: $(\lambda x. xy(\lambda y. y)) [y : (\lambda z. z)] \Rightarrow (\lambda x. x(\lambda z. z)(\lambda y. y))$

β - Reduktion

β -Reduktion ist die Auflösung einer Applikation von einem Term t_2 auf einer Abstraktion $(\lambda x. t_1)$.

Voraussetzung: $GV(t_1) \cap FV(t_2) = \emptyset$.

Es werden alle freien Vorkommen von x in t_1 durch t_2 substituiert.

$$\overbrace{(\lambda x. t_1) t_2}^{\text{Applikation}} \Rightarrow_{\beta} t_1 [x/t_2]$$

Abstraktion

Zu beachten ist die implizite links Assoziativität

- nicht β -reduzierbar da implizite Klammerung: $(x(\lambda y. y)z) = ((x(\lambda y. y))z)$
- innerhalb Abstraktion reduzierbar: $(\lambda x. (\lambda y. yx)z) \Rightarrow_{\beta} (\lambda x. (yx) [y/z]) \Rightarrow (\lambda x. zx)$

α - Konversion

Umbenennung einer gebundenen Variable in einem Term.

$$(\lambda x. t_1) \Rightarrow_{\alpha} (\lambda x'. t_1 [x/x'])$$

Voraussetzung: x' kommt nicht in t_1 vor

Normalform

Normalform ist erreicht wenn keine weitere β -Reduktion mehr möglich ist.

Es ist möglich mehrere β -Reduktionen auf einen λ -Term anzuwenden. Alle Folgen von β -Reduktionen (\Rightarrow^*) führen zur gleichen Normalform. (Konfluenz von λ -Termen)

Zahlen, Funktionen und Wahrheitswerte im Lambda-Kalkül

- (Church) Zahlen: $\langle 0 \rangle = (\lambda xy.y)$, $\langle 1 \rangle = (\lambda xy.xy)$, $\langle 2 \rangle = (\lambda xy.x(xy))$, ... $\langle n \rangle = (\lambda xy.\underbrace{x(\dots(x(xy))\dots)}_n)$
- Wahrheitswerte: $\langle true \rangle = (\lambda xy.x)$, $\langle false \rangle = (\lambda xy.y)$
- Nachfolgerfunktion: $\langle succ \rangle = (\lambda z.(\lambda xy.x(zxy)))$ Bsp: $\langle succ \rangle \langle n \rangle \Rightarrow^* \langle n + 1 \rangle$
- Vorgängerfunktion: $\langle pred \rangle = (\lambda k.k(\lambda pu.u(\langle succ \rangle(p\langle true \rangle)))(p\langle true \rangle))(\lambda u.u\langle 0 \rangle\langle 0 \rangle)\langle false \rangle)$
- Ist Null: $\langle iszero \rangle = (\lambda k.k(\langle true \rangle\langle false \rangle)\langle true \rangle)$ Bsp: $\langle iszero \rangle \langle 0 \rangle = \langle true \rangle$
- if-then-else: $\langle ite \rangle = (\lambda bxy.bxy)$
 $\langle ite \rangle bxy \Rightarrow \begin{cases} x & \text{wenn } b \Rightarrow^* \langle true \rangle \\ y & \text{sonst} \end{cases}$
- Addition: $\langle Add \rangle = (\lambda zxy.\langle ite \rangle(\langle iszero \rangle x)y(\langle succ \rangle(z(\langle pred \rangle x)y)))$
 $\langle add \rangle = \langle Y \rangle \langle Add \rangle$
- Multiplikation: $\langle Mult \rangle = (\lambda zxy.\langle ite \rangle(\langle iszero \rangle x)\langle 0 \rangle(\langle add \rangle y(z(\langle pred \rangle x)y)))$
 $\langle mult \rangle = \langle Y \rangle \langle Mult \rangle$
- Fixpunktkombinator: $\langle Y \rangle = (\lambda h.((\lambda y.h(yy))(\lambda y.h(yy))))$