Vorlesung 4

Alexander Mattick Kennung: qi69dube

Kapitel 1

28. Mai 2020

1 4

1.0.1

$\lceil 2 \rceil = \lambda fa.ffa$

$\lceil 1 \rceil = \lambda fa.fa$

$pred\ n = \lambda fa.n(\lambda gh.h(gf))(\lambda u.a)(\lambda u.u)$

$pred\,\lambda f_0a_0.f_0f_0a_0 = \lambda fa.(\lambda f_0a_0.f_0f_0a_0)(\lambda gh.h(gf))(\lambda u.a)(\lambda u.u) \rightarrow_\beta \lambda fa.(\lambda a_0.(\lambda gh.h(gf)(\lambda gh.h(gf)a_0))(\lambda u.a)(\lambda u.u) \rightarrow_\beta$

$\lambda fa.((\lambda gh.h(gf)(\lambda gh.h(gf)(\lambda u.a)))(\lambda u.u) \rightarrow_\beta \lambda fa.((\lambda h_0.h_0((\lambda gh.h(gf)f))(\lambda u.a)))(\lambda u.u) \rightarrow_\beta \lambda fa.((\lambda h_0.h_0((\lambda h.h((\lambda u.a)ff)$

$\lambda fa.(\lambda h_0.h_0\lambda h.h((\lambda u.a)ff))(\lambda u.u) \rightarrow_\beta \lambda fa.(f\lambda h.h((\lambda u.a)f))(\lambda u.u) \rightarrow_\beta \lambda fa.f\lambda h.h(\lambda u.a)f(\lambda u.u) \rightarrow_\beta \lambda fa.f\lambda h.ha(\lambda u.u) \rightarrow_\beta$

$\lambda fa.f(\lambda u.u)a \rightarrow_\beta \lambda fa.fa \rightarrow_\delta \lceil 1 \rceil$

1.1

sub n m = $m(pred\ n)$

Dazu definiert $true = \lambda xy.x$ $False = \lambda xy.y$

Hilfskonstrukt "istNull" = $\lambda n.n\lambda x.(false)(true)$

wenn man 0 einsetzt erhält man $(\lambda n.n\lambda x.(false)(true))\lambda fa.a \rightarrow \lambda fa.a\lambda x.(false)(true) \rightarrow \lambda a.a(true) \rightarrow true.$

Wenn man einen wert ungleich null einsetzt erhält man:

$(\lambda n.n(false)(true))\lambda fa.\underbrace{f}_{nmal}a \rightarrow (\lambda fa.\underbrace{f}_{nmal}a)(false)(true) \rightarrow (\lambda a.\underbrace{\lambda x.false}_{nmal}a)(true) \rightarrow \underbrace{false}_{nmal}(true)$

Durch das $\lambda x.false$ werden alle nachfolgenden zeichen ignoriert $\lambda x.false(t) \rightarrow false$ somit "kollabiert" die "false, false,..., true" Kette von links nach rechts zu nur einem false.

Zweites Hilfskonstrukt: "und" $\lambda nm.nm(false)$, wenn n true ist, dann wird immer der wert von m gewählt und das letzte false verfällt. $\lambda xy.xm(false) \rightarrow \lambda y.m(false) \rightarrow m$

Wenn n false ist, dann ist egal, was m ist, das Ergebnis ist immer false $\lambda xy.ym(false) \rightarrow \lambda y.y(false) \rightarrow (false)$

le n m= istNull (sub n m)

eq n m= und (istNull (sub n m)) (istNull (sub m n)) für lt lohnt es sich ein "Nicht" zu definieren:

Nicht = $\lambda x.x(false)(true)$

Wenn man $true$ einsetzt: $(\lambda x.x(false)(true))(true) \to (true)(false)(true) \to \lambda xy.x(false)(true) \to \lambda y.(false)(true) \to (false)$

Wenn man $false$ einsetzt: $(\lambda x.x(false)(true))(false) \to (false)(false)(true) \to \lambda xy.y(false)(true) \to \lambda y.y(true) \to (true)$

daraus folgt für lt:

lt n m= und (le n m) (nicht (eq n m)) 3.

Zuerst wendet pred auf den eingesetzen wert $\lambda gh.h(gf)$ an, sodass diese funktion n-mal ineinander geschachtelt wird.

Diese funktion hat die Eigenschaft, dass sie eine zweite funktion nimmt, und diese auf f anwendet, bevor diese wiederolt angewandt wird.

Das zweite, was an unser n angwandt wird (das "a" in $\lambda fa.f \ldots fa$) ist eine konstante, die, egal was auf sie angewandt wird, immer a zurückliefert.

Dies führt dazu, dass das erste f in unserem n-hohen f-Turm zu einem a reduziert wird, da "const a" auf f angewandt wird $\lambda h.h(\text{``}consta''f) \to \lambda h.ha$

jetzt muss noch irgendwie der zweite parameter aufelöst werden, was mit der anwendung der identitätsfunktion $\lambda u.u$ bewerkstelligt wird.

Dadurch dass das h an erste stelle steht, wird die identitätsfunktion von $\lambda gh.h(gf)$ den gesamten turm hinaufgegeben.

Bei allen anwendungen außer der ersten, kann f einfach an konkateniert werden, da g keine konstante ist.

am Ende kann die identität entfernt werden, und der vorgänger steht da, da die konstante im ersten schritt schließlich ein f entfernt hat.