Inhaltsverzeichnis

Vorlesung 2

Alexander Mattick Kennung: qi69dube

Kapitel 1

15. Juni 2020

1 Terme

$\Sigma - Terme\ t ::= x | f(t_1, \ldots, t_n)\ (x \in V, f/n \in \Sigma)$

V Menge von Variablen.

$T_\Sigma(V)$ = Menge der Σ-Terme über V (ist nicht fix, kann sich u.u. z.B. verkleinern)

$FV(t)$ = Menge der in t (frei) vorkommenden Variablen.

$FV(x) = \{x\}$

$FV(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} FV(t_i)$

1.1 substitution

Substitution ist eine Abbildung $\sigma : V_0 \to T_\Sigma(V)$ für ein $V_0 \subseteq V, V_0$ endlich.

$[t_1/x_1, \ldots, t_n/x_n]\ V_0 = \{x_0, \ldots, x_1\}, \sigma(x_i) = t_i\ \ t\sigma = \begin{cases} x\sigma = \sigma(x) \\ \\ f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma) \end{cases}$

1.2 Kontext

$C(\cdot) = (\cdot) | f(t_1, \ldots, c(\cdot), \ldots, t_n),\ (f/n \in \Sigma)$

$f(t_1, \ldots, C(\cdot), \ldots, t_n)(g) = f(t_1, \ldots, C(g), \ldots, t_n)$

2 Operationelle Semantik von TES

$\to_0 \subseteq T_\Sigma(V) \times T_\Sigma(V)$

$R \subseteq T_\Sigma(V) \times T_\Sigma(V)$ heißt

- abgeschlossen bezüglich $C(\cdot)$, wenn $\forall t, s (tRs \implies C(t)RC(s))$
  - Bsp.: $x + y + = y + x \implies z * (x + y) = z(y + x)$ zeigt Kontextabschluss $C(\cdot) = z * (\cdot)$
- Kontextabgeschlossen $\iff$ R abgeschlossen für alle $C(\cdot)$
- stabil $\iff \forall t, s, \sigma (tRs \implies (t\sigma)R(s\sigma))$
  - z.B. $x + y = y + x \implies z^2 + xw = xw + z^2$

Einschrittreduktion $\to \subseteq T_\Sigma(V) \times T_\Sigma(V)$ = kontextabgeschlossener und stabiler Abschluss von $\to_0$

$\to = \{(C(s\sigma), C(t\sigma)) \mid t \to_0 s, C(\cdot)$Kontext, σ Substitution$\}$

Bew.: Wenn man einen Kontext von einem Kontext macht, erhält man einen Kontext (weil es nur eine Freistelle gibt). Wenn man substituiert dann ist die substitution entweder in s/t oder im Kontext, substitution im Kontext ändert nur in einen neuen kontext (es bleibt aber kontext).

Reduktion $\to^*$ (sprich "t reduziert zu s")

Konvertierbarkeit $\leftrightarrow^* = (\to \cup \to^-)^*$ ist die Äquivalenz zu $\to$.

t **normal** $\iff \neg\exists s(t \to s) \iff t \not\to$ s Normalform von t $\iff t \to^* s$ s normal

Lemma 2.1. *Sei* $R \subseteq T_\Sigma(v) \times T_\Sigma(V)$

1) R kontextabg. $\iff$ *R abgeschlossen bzg. aller* $f(t_1,\dots,t_{i-1},(\cdot),t_{i+1},\dots,t_n)$ *(Induktion über kontexte!):*

$(\cdot)$ *ist trivial.*

$tRs \implies C(t)RC(s) \implies f(t_1,\dots,C(t),\dots,t_n)Rf(t_1,\dots,C(s),\dots,t_n)$

2) R stabil $\implies$ *(R kontextabg.* $\iff$ *R abgeschlossen bezgl aller* $f(x_1,\dots,(\cdot),\dots,x_n)$*) (folgt direkt aus 1.)*

Beispiel 2.1.

$\Sigma = \{+/2, s/1, 0/0\}$

1) $s(x) + y \to_0 s(x+y)$

2) $0 + y \to_0 y$

3) $(x+y)+z \to_0 x+(y+z)$

Es gibt versch umklammerungsmöglichkeiten:

$(s(x)+s(y))+z \overset{1),C(\cdot)+z,\sigma=[s(y)/y]}{\to} s(x+s(y))+z \overset{1),C(\cdot),\sigma=[(x+s(y))/x]}{\longrightarrow} s((x+s(y))+z) \xrightarrow{3)} s(x+(s(y)+z) \xrightarrow{1)} s(x+s(y+z))$

$(s(x)+s(y))+z \overset{3),C(\cdot),\sigma=[s(x)/x,s(y)/y]}{\to} s(x)+(s(y)+z) \xrightarrow{1)} s(x)+s(y+z) \xrightarrow{1)} s(x+s(y+z))$

Unterschied Gleichungstheorie und TES: Gleichungstheorie ist eine **Umkehrbare** relation zwischen Termen!

3 Terminierung

Definition 3.1.

$R \subseteq X \times X$ wohlfundiert $\iff$ es existiert keine unendliche folge $x_0,\dots,x_n$ mit $x_0 R x_1 R \dots$

$(\mathbb{Z}, >)$ ist nicht wohlfundiert. $(0 > -1 > -2 > \dots$

$(\mathbb{Q}_+, >)$ ist nicht wohlfundiert $1 > \frac{1}{2} > \frac{1}{4} > \frac{1}{8} > \dots$

$(\mathbb{N}, >)$ ist wohlfundiert (es endet spätestens bei 0, induktion über Kettenanfänge)

Beweis 3.1. i.V. Die Kette $n_1 > n_2 > \dots$ ist endlich.

Annahme: es gibt eine unendliche Kette bei $n_0 > n_1 > \dots \implies n_1 > n_2 > \dots$ wäre auch unendlich $(\infty - 1 = \infty)$. Widerspruch zur Induktionsvorraussetzung!

Definition 3.2.

- schwach normalisierend $\iff$ t hat eine NF. $t \to \cdots \to s$ normal.
- stark normalisierend $\iff$ es gibt keine unendliche reduktionsfolge $\neg\exists t = t_0 \to t_1 \to \dots$ (unendlich). (es gibt keine zyklen)

TES $(\Sigma, \to_0)$ schwach/stark normalisierend (WN(SN)) $\iff$ alle t in $(\Sigma, \to_0)$ schwach/stark normalisierend.

Beispiel 3.1.

$f(x) \to_0 f(x)$

$g(x) \to 1$

$g(x)$ stark normalisierend einzige Reduktion $g(x) \to 1 \not\to$

$f(x)$ nicht schwach normalisierend einzige Reduktion $f(x) \to f(x) \to \dots$

$g(f(x))$ schwach normalisierend:$g(f(x)) \to 1 \not\to$(Haskell ausführung)

oder $g(f(x)) \to g(f(x)) \to \dots$ (deshalb nicht stark normalisierend, hier ML-ausführung)

4 Reduktionsordnungen

$\leq$ vs $<$: reflexiv vs. irreflexiv: $\forall x(\neg xRx)$.

R ist strikte Ordnung $\iff$ R transitiv und Irreflexiv (z.B.$>$).

Definition 4.1.

$R \subseteq T_\Sigma(V) \times T_\Sigma(V)$.

Reduktionsordnung $\iff$ R wohlfundierte, stabile, kontextabgeschlossen, strikte Ordnung.

(aus wohlfundiert folgt strikt, sonst könnte man eine unendliche Folge $xRxRxRxR\dots$).

Satz 1. *Sei $>$ Reduktionsordnung und $\forall t, s(t \to_0 s \implies t > s) \implies \to SN$ (also in jeder Ersetzungsregel wird nach anwendung der Term kleiner).*

Beweis 4.1. $>$ ist stabil und kontextabgeschlossen, und $\to_0 \subseteq > \implies \to \subseteq > \implies \to$ ist wohlfundiert, d.h. $\to$ ist SN.

(Weil $\to$ der kontextabg. und stabile abschluss von $\to_0$ ist, wenn $>$ wohlfundiert ist, dann kann es auch keine unendlichen Mengen in der Teilmenge $\to$ geben)

Beispiel 4.1.

$|t|$ = Größe von t. $t > s :\iff |t| > |s|$ (also die länge).

kontextabgeschlossen: $|t| > |s| \implies |C(t)| > |C(s)|$ (freiplatz kommt einmal vor.)

stabil? nicht immer $|x + 2y - x| > |y + y|$ aber: $\sigma = [100x/y]$ $|x + 2 * 100x - x| \not> |100x + 100x|$!

aber Ok, wenn in $t \to_0 s$ stets jede variable s höchstens so oft wie in t vorkommt.

$\emptyset$ ist eine Reduktionsordnung.

$\to$ SN $\implies \to^+$ Reduktionsordnung.

4.1 Polynomordnungen

Recall: Polynome die menge der Polynome über $\mathbb{N}$ d.h. mit natürlich zahligen koeffizienten (insbesondere also keine z.B. $-1x^2$).

$$\mathbb{N}[x_1,\ldots,x_n] = \left\{ \sum_{i_1,\ldots,i_n \in \mathbb{N}} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} | a_{i_1,\ldots,i_n} \in \mathbb{N}\ a_{i_1\ldots,i_n} = 0 \text{ fast immer} \right\}$$

z.B. $x^2y + 2y^2zx \in \mathbb{N}[x,y,z]$ ein summand wird “Monom” genannt. z.B. ist y^2zx ein monom und gehört zu $a_{121} = 2$

jedes $p \in \mathbb{N}[x_1,\ldots,x_n]$ definiert eine Funktion

$$\mathbb{N}^n \to \mathbb{N}\ \ (k_1,\ldots,k_n) \to p(k_1,\ldots,k_n) \in \mathbb{N}$$

p,q Polynom $\implies p+q, p \times q$ ist Polynom (nach Zusammenfassen gleichartiger Monome)

$\implies$ für $p \in \mathbb{N}[x_1,\ldots,x_n], q_1,\ldots q_n \in \mathbb{N}[y_1,\ldots,y_k]$

$\implies p(q_1,\ldots,q_n) \in \mathbb{N}[y_1,\ldots,y_k]$ (die eingesetzten polynome können von jeder Art “k” sein, das “buffert” auch unten evtl vorliegende $(x^2)^3 = x^6$ mit $a_{xyz} = 0$)

$k_1,\ldots,k_n \in \mathbb{N} \implies p(k_1,\ldots,k_n) \in \mathbb{N}$

Definition 4.2. Sei $\emptyset \neq A \subseteq \mathbb{N}$.

$p >_A q \iff \forall k_1,\ldots,k_n \in A(p(k_1,\ldots,k_n) > q(k_1,\ldots,k_n))$

Beispiel 4.2.

$x^2 >_{\mathbb{N}} x$ gilt nicht $1^2 \not> 1$ aber schon für $A = \{n \in \mathbb{N} | n \geq 2\}$

Lemma 4.1. $>_A$ *ist wohlfundiert.*

Beweis 4.2. Annahme: $p_0 >_A p_1 >_A \ldots$(unendlich) wähle $a \in A$; dann $p_0(a,\ldots,a) > p_1(a,\ldots,a) > \ldots$ in $\mathbb{N}$ WIDERSPRUCH ($>_{\mathbb{N}}$ ist wohlfundiert)

Definition 4.3. $p \in \mathbb{N}[x_1,\ldots,x_n]$ <u>streng monoton</u>:

$$\forall j \exists i_1,\ldots,i_n (i_j > 0 \wedge a_{i_1\ldots i_n} > 0)$$

(also wenn x_i im polynom struktur ist, muss es auch einen koeffizienten geben, der $\neq 0$ ist)

Lemma 4.2.

$$p \text{ streng monoton} \iff \forall k_1,\ldots,k_n,k_1',\ldots,k_n'((k_1,\ldots,k_n) > (k_1',\ldots,k_n') \implies p(k_1,\ldots,k_n) > p(k_1',\ldots,k_n'))$$

$\iff$ *1)* $\forall j(k_j \geq k'_j)$ *und 2)* $\exists j(k_j > k'_j)$ *(mindestens eins echt größer).*

Beweis 4.3. " $\implies$ "

$a_{i_1,\dots,i_n} k_1^{i_1} \dots k_n^{i_n} \geq a_{i_1,\dots,i_n} k_1^{i'_1} \dots k_n^{i'_n}$ stets, einmal ">"□

Definition 4.4. (monotone) Polynomielle Interpretion $\mathscr{A}$ besteht aus

- zu jedem $f/n \in \Sigma$ ein streng monotones $p_f \in \mathbb{N}[x_1, \dots, x_n]$
- $A \subseteq \mathbb{N}$ die unter p_f abgeschlossen ist

so dass $k_1, \dots, k_n \in A \implies p_f(k_1, \dots, k_n) \in A$

(Eine polynomordnung besteht aus einem polynom für jedes signatursymbol und einer auswahl natürlicher Zahlen)

$\rightarrow$ Polynomordnung $\succ_{\mathscr{A}}$ $t \succ_{\mathscr{A}} s \iff p_t \succ_{\mathscr{A}} p_s$

mit $p_x = x$ $p_{f(t_1,\dots,t_n)} = p_f(p_{t_1}, \dots, p_{t_n})$

Satz 2. $\succ_A$ *ist eine Reduktionsordnung!*

Korrolar 4.1. *Wenn* $t \rightarrow_0 s \implies t \succ_{\mathscr{A}} s$*, dann* $\rightarrow$ *SN.*

Beispiel 4.3.

$f(f(g(x))) \rightarrow_0 f(g(g(x)))$

$p_f(x) = x^2 + 1, p_g(x) = x$ also $f(f(g(x))) \equiv (x^2+1)^2 + 1 \succ_{\mathbb{N}} f(g(g(x))) = x^2 + 1$

oder einfach $p_f(x) = x^2, p_g(x) = x$ also, dann muss man jedoch $A = \mathbb{N} \setminus \{1, 0\}$

Lemma 4.3. *(Substitutionslemma):*

$\sigma = [t_1/x_1, \dots, t_n/x_n], p \in \mathbb{N}[x_1, \dots, x_n] \implies p_{t\sigma} = p_t(p_{t_1}, \dots, p_{t_n})$

Beweis 4.4. Induktion über t.

- $p_{x_i\sigma} = p_{t_i} = P_{x_i}(p_{t_1}, \dots, p_{t_n})$

$p_{f(s_1,\dots,s_k)\sigma} = p_f(p_{s_1}\sigma, \dots, p_{s_k}\sigma)$

$\underset{IV}{=} p_f(p_{s_1}(p_{t_1}, \dots, p_{t_n}), \dots)$

$\overset{substitution}{=} p_f(p_{s_1}, \dots, p_{s_k})(p_{t_1}, \dots, p_{t_n}) = p_{f(s_1,\dots,s_n)}(p_{t_1}, \dots, p_{t_n})$

Beweis 4.5. ($\succ_{\mathscr{A}}$ ist Reduktionsordnung)

- strikte Ordnung per definition
- wohlfundiert (es gibt keine endlos absteigende polynomfolge)
- $\succ_{\mathscr{A}}$ stabil: Sei $t \succ_{\mathscr{A}} s, \sigma = [t_1/x_1, \dots]$

zZ.: $t\sigma >_{\mathscr{A}} s\sigma$: Seien $k_1,\dots,k_n \in A$

$p_{t\sigma}(k_1,\dots,k_n) \underset{Lemma}{=} p_t(p_{t_1}(k_1,\dots,k_n),\dots) > p_s(p_{t_1}(k_1,\dots,k_n),\dots) \underset{Lemma}{=} P_{s\sigma}(k_1,\dots,k_n)$

- $>_{\mathscr{A}}$ kontextabgeschlossen: Sei $t >_{\mathscr{A}} s$, $C(\cdot) = f(x_1,\cdot,(\cdot)_i,\dots,x_n)$ (weil stabilität schon gezeigt, reicht das)

  zZ: $C(t) >_{\mathscr{A}} C(s)$ Seien $k_1,\dots,k_n \in A$

  $p_f(k_1,\dots,p_t(k_1,\dots,k_n),\dots,k_n) \overset{streng\ monoton}{<} p_f(k_1,\dots,p_s(k_1,\dots,k_n))$

Es ist beweisbar untentscheidbar, ob es für eine gegebene reduktionsordnung eine polynomordnung die deren Terminierung beweist, gibt. (halteproblem)

Beispiel 4.4.

$(x \oplus y) \oplus z \to_0 x \oplus (y \oplus z)$

$x \oplus (y \oplus z) \to_0 y \oplus y$

Gesucht ist also eine poly interpretation von "$\oplus$":

Hier: Linke seite muss mehr gewichtet werden als die Rechte.

$p_\oplus(x,y) = x^2 + y$

führt zu:

$(x^2+y)^2 + z >_{\mathscr{A}} x^2 + (y^2+z) = x^4 + 2x^2y + y^2 + z$

$\mathscr{A} = [1,\infty)$

$x^2 + y^2 + z \not>_{\mathscr{A}} y^2 + y$

Geht also nicht, wenn man $x^2 \to \infty$

Besser:

$p_\oplus(x,y) = x^2 + xy$

$(x^2+xy)^2 + (x^2+xy)z = x^4 + 2x^3y + x^2y^2 + x^2z + xyz >_{\mathscr{A}} x^2 + x(y^2+yz) = x^2 + xy^2 + xyz$

$\mathscr{A} = \mathbb{N}_{\geq 1}$

$x^2 + xy^2 + xyz >_{\mathscr{A}} y^2 + yy = 2y^2$

$\mathscr{A} = \mathbb{N}_{\geq 2}$

(Wichtig, man darf keine variablen "verlieren" wenn man noch umformungsschritte hat!)

5 Konfluenz

Beispiel 5.1. Gruppen

$x \cdot (y \cdot z) \overset{\to_0}{=} (x \cdot y) \cdot z$

$x \cdot e \overset{\to_0}{=} x$

$x \cdot x^{-1} \overset{\to_0}{=} e$

$y \cdot (x \cdot x^{-1}) \to y \cdot e \to y \not\to$ ist eine NF

oder

$y \cdot (x \cdot x^{-1}) \to (y \cdot x) \cdot x^{-1} \not\to$ ist eine NF

(Knuth-bendix algorithmus[1] würde zur konfluenz führen: regel von einer der beiden NF zur anderen)

Definition 5.1.

- t,s **zusammenführbar** (zf) $\iff \exists u(t \to^* u \ {}^*\!\leftarrow s)$ (u.U auch mit null schritten)
- TES T heißt **konfluent** (CR, church/Rosser) $\iff \forall t, s, s'(t \to^* s \wedge t \to^* s' \implies s, s' \ zf)$

$$\begin{tikzcd}
 & t \arrow[dl, "*"'] \arrow[dr, "*"] & \\
s \arrow[dr, dotted, "*"'] & & s' \arrow[dl, dotted, "*"] \\
 & q &
\end{tikzcd}$$

- T heißt **lokal konfluent** (WCR, weakly Church/Rosser) $\iff \forall t, s, s'(t \to s \wedge t \to s' \implies s, s' \ zf)$ (also in nur einem schritt zusammenführbar)

$$\begin{tikzcd}
 & t \arrow[dl] \arrow[dr] & \\
s \arrow[dr, dotted, "*"'] & & s' \arrow[dl, dotted, "*"] \\
 & q &
\end{tikzcd}$$

z.B.: ist oben 5.1 weder stark noch schwach CR

Satz 3. *Sei t konfluent* $\implies$

1) $s \leftrightarrow^* t \iff s, t \ zf$

2) s, s' *NF von* $t \implies s = s'$

Beweis 5.1.

1) $\impliedby$ klar $\implies$

Haben $s = t_0 \leftrightarrow t_1 \leftrightarrow \cdots \leftrightarrow t_n = t$

Induktion über n:

$n = 0 : s = t$ klar

$n \to n+1$: Nach I.V. $s = t_0 \to^* q^* \leftarrow t_n \leftrightarrow t_{n+1}$

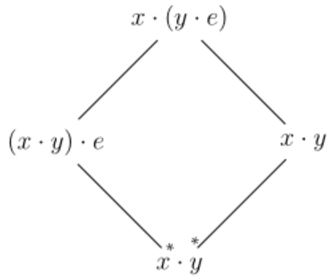
Fall 1: $t_n \leftarrow t_{n+1}$ fertig (weil t_n über q mit s zusammenführbar)

Fall 2: $t_n \to t_{n+1}$ dann gibt es ein r, dass über $\to^*$ mit t_{n+1} und q erreichbar ist (wegen konfluenz)

$$\begin{tikzcd}
 & t_n \arrow[r] \arrow[d, "*"] & t_{n+1} \arrow[d, dashed, "*"] \\
t_0 \arrow[r, "*"] & q \arrow[r, dashed, "*"] & r
\end{tikzcd}$$

2) $s \leftrightarrow^* s' \implies s, s' \ zf : s \to^* u^* \leftarrow s'$ weil s, s' NF, braucht man genau 0 schritte: $s = u = s'$

hier ein bsp für eine konfluente form:



Satz 4. *(Newman's Lemma)*

$$SN \wedge WCR \implies CR$$

(also lokale konfluenz und stark normalisierend, führt zur vollen konfluenz, starke konfluenz ist i.a untentscheidbar (und so auch SN, deshalb widerspricht dieser Satz dem nicht...)) Beweis, später

Beispiel 5.2.

Regeln

$l_1 \to r_1$

$l_2 \to r_2$

Terme

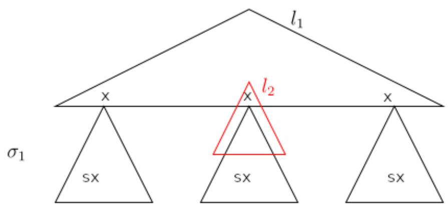
$C_1(l_1\sigma_1) = t = C_2(l_2\sigma_2)$

$C_1(l_1\sigma_1) \to C_1(r_1\sigma_1)$

$C_2(l_2\sigma_2) \to C_2(r_2\sigma_2)$

$$\begin{array}{ccccc} l_1 & C_1(l_1\sigma_1) & = t = & C_2(l_2\sigma_2) & l_2 \\ \downarrow_0 & \downarrow & & \downarrow & \downarrow_0 \\ r_1 & C_1(r_1\sigma_1) & & C_2(r_2\sigma_2) & r_2 \end{array}$$

C_1 kann ignoriert werden wegen kontextabgeschlossen.



Weil l_2 in l_1 hineinragt, ist nach anwendung von $l_2 \to r_2$ kein l_1 mehr für die zweite Regel vorhanden (die eine anwendung zerschiest die prämisse einer zweiten)

Definition 5.2. Unifikation

t, s Terme $t \doteq s$

σ Unifikator von t,s ($\sigma \in Unif(t,s)$) $\iff$ $t\sigma = s\sigma$ (syntaktisch)

t, s unfiz $\iff$ $unif(t,s) \neq \emptyset$

σ allgemeiner als $\sigma' \iff \exists\tau(\sigma' = \sigma\tau)$

σ allgemeinster Unifikator (mgu) von t, s $\sigma = mgu(t,s) \iff \sigma \in Unif(t,s) \land \forall\sigma' \in Unif(t,s)$($\sigma$allgemeiner als σ')

mgu existiert, wenn t,s unifizierbar, eindeutig bis auf isomorphismus (injektive umbennennung)

Beispiel 5.3. unifikation

$k(r(x), x) \doteq k(z, r(z))$

decomp $r(x) \doteq z, x \doteq r(z)$

elim $r(r(z)) \doteq z, x \doteq r(z)$

occurs.

$f(h(x), z), f(y, g(x))$

$\sigma = [h(x)/y, g(x)/z]$

Definition 5.3. kritisches Paar nach Knuth-Bendix

Seien $l_1 \to_0 r_1, l_2 \to_0 r_2$,

$l_1 = C(t)$ t nichttrivial (d.h. t keine Variable, konstanten gehen aber...)

und $FV(l_2) \cap FV(l_1) = \emptyset$

$\sigma = mgu(t, l_2)$

also, wenn man $r_1\sigma \leftarrow l_1\sigma = C(t)\sigma = (C\sigma(t\sigma) = C\sigma(l_2) \to C\sigma(r_2\sigma) = C(r_2)\sigma$

$\implies (r_1\sigma, C(r_2)\sigma)$ **kritisches Paar**

Lemma 5.1. $(r_1\sigma, C(r_2\sigma))$ *kritisches Paar*

$\implies r_1\sigma \leftarrow l_1\sigma = C(l_2)\sigma \to C(r_2)\sigma$ *(kritische Paare sind divergente Redukte eines gemeinsamen ursprungs)*

Korrolar 5.1. $T\ WCR \implies$ *alle Paare sind zf*

Satz 5. *alle kritischen Paare zf* $\implies$ WCR *(Critical Pair Lemma)*

Aufwand ist $O(n^3)$ *(paare und dann jede Regel für kontext C(t) einsetzen, mal die anzahl der Schritte, den jede reduktion selbst benötigt)*

Beispiel 5.4. (Gruppe)

$(l_1 \to_0 r_1) = (x \cdot (y \cdot z)) \to_0 (x \cdot y) \cdot z)$

(in frische variablen umbennenen)

$(l_2 \to_0 r_2) = (x' \cdot e \to x')$

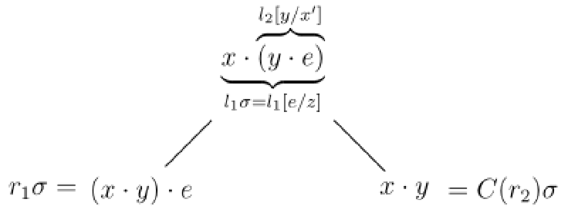
Jetzt: wähle einen Teilterm aus, und mach das "t" draus:

$C(\cdot) = x \cdot (\cdot)$

$t = y \cdot z \; \sigma = mgu(t, l_2) = [y/x', e/z]$

$\rightarrow$ kritisches Paar $(r_1\sigma, C(r_2)\sigma) = ((x \cdot y) \cdot e,\; x \cdot y)$

(die r_1, r_2 sind oben definiert. . .)



Beispiel 5.5.

$l_1 \rightarrow_0 r_1 = (x \cdot (y \cdot z)) \rightarrow_0 (x \cdot y) \cdot z)$

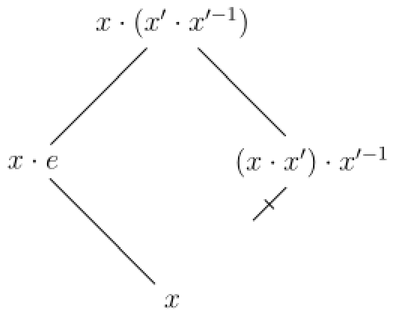
$l_2 \rightarrow_0 r_2 = (x' \cdot x^{-1'} \rightarrow_0 e)$

$C(\cdot) = x \cdot (\cdot)$

$t = y \cdot z$

$\sigma = mgu(t, l_2) = [x'/y, x^{-1'}/z]$

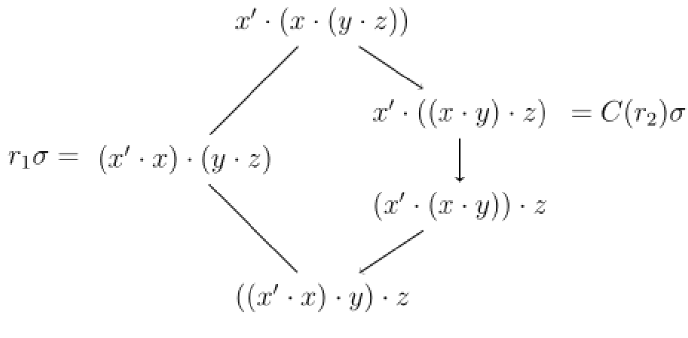
$(x \cdot x') \cdot x^{'-1} \leftarrow x \cdot (x' \cdot x^{-1'}) \rightarrow x \cdot e$ nicht z.f.



Beispiel 5.6. $l_1 \rightarrow_0 r_1 = (x \cdot (y \cdot z) \rightarrow_0 (x \cdot y)z) = l_2 \rightarrow_0 r_2$

$t = y \cdot z \; \sigma = mgu(t, l_2) = mgu((y \cdot z), x' \cdot (y' \cdot z')) = [y/x', y' \cdot z'/z]$

$(x \cdot y) \cdot (y' \cdot z') \leftarrow x \cdot (y \cdot (y' \cdot z')) \rightarrow x \cdot ((y \cdot y') \cdot z')$



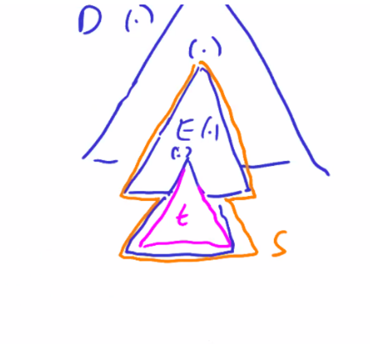
sind z.f.

<u>ACHTUNG</u>

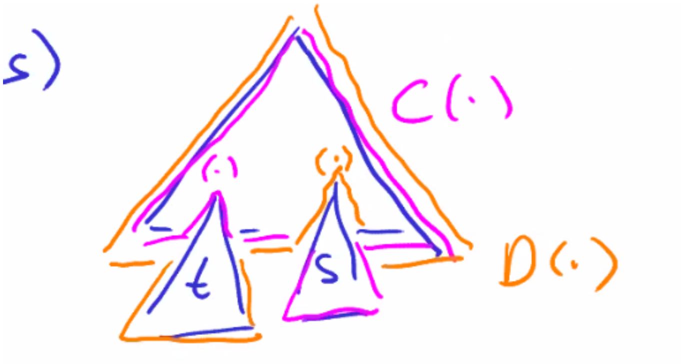
wenn man das umbennenen der variablen vergisst, dann krigt man $y \cdot z$ und $x \cdot (y \cdot z) \longrightarrow \perp$ occurs!!

Beweis 5.2. Critical Pair Lemma5

Notation $C(\cdot) \sqsubseteq D(\cdot) \iff \exists E(\cdot)(C(\cdot) = D(E(\cdot)))$ (also C liegt unter D, wenn man in D einen weiteren kontext einführen kann, um ihn zu C zu verwandeln! Wie bei mgu auch)

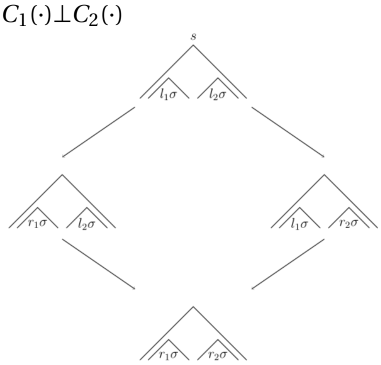


$C(\cdot) \perp D(\cdot) \iff C(\cdot) \not\sqsubseteq D(\cdot) \wedge D(\cdot) \not\sqsubseteq C(\cdot)$ (also keiner ist subset des anderen, sie sind orthogonal)



Sei $l_1 \rightarrow_0 r_1, l_2 \rightarrow r_2$ anwendbar auf s

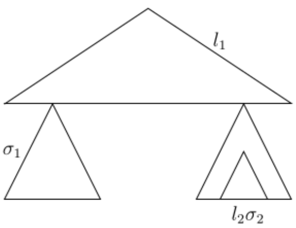
Fall 1:$C_1(\cdot) \perp C_2(\cdot)$



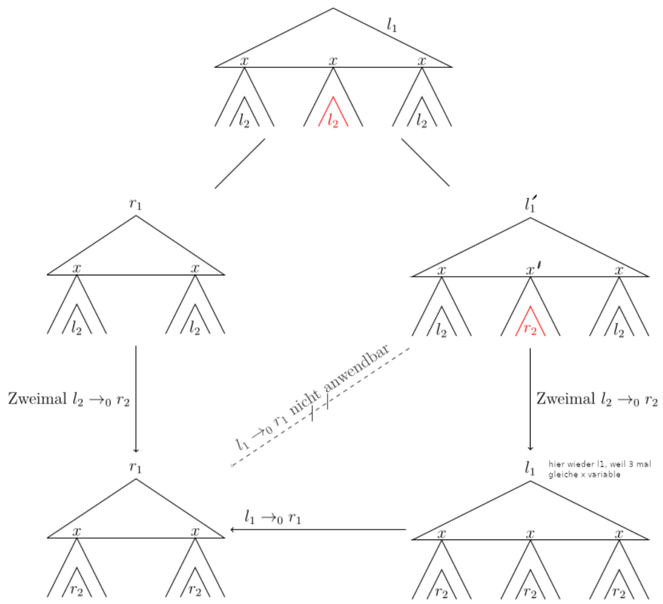
Beide Terme stören sich nicht, ich kann immer beide Regeln in beliebiger Reihenfolge anwenden

Fall 2: o.b.d.A $C_2(\cdot) \sqsubseteq C_1(\cdot)$ mit $C_1(\cdot) = (\cdot)$ (man kann sich den äußersten einfach wegdenken, der Teilbaum unter einem echten $C_1 \neq (\cdot)$ ist equivalent zu einem normalen Baum mit wurzel direkt unter C_1)

ohne einschrenkung $l_2\sigma = l_2$, weil l_2 sowieso vollkommen unter unserer substitution liegt, also auch im nachhinein gemacht werden kann (es stört den Rest des Terms nicht).

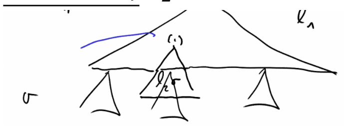


unterfall 2a) C_2 ist echt unterhalb von l_1



auf der rechten seite ist $l_1 \to r_1$ nicht mehr anwendbar, weil $l_1 \to r_1$ fordert, dass es 3 gleiche argumente gibt.

unterfall 2b) C_2 ist nicht unterhalb von l_1, d.h. $(\cdot)$ von C_2 liegt in l_1:



Dies ist gleich der situation des Kritischen paares 5.1: Der einzige ort, wo es schiefgehen kann ist also, wenn das kritische paar nicht zf ist.

Es reicht also: Für $\sigma \in Unif(t, l_2)$ ist $(r_1\sigma, C_2(r_2)\sigma)$ zf.

Gilt nach Annahmen für $\sigma = mgu(t, l_2)$ (alle kritischen paare sind zf), dann $\sigma' = \sigma\tau$ für ein τ

$r_1\sigma\tau, C_2(r_2)\sigma\tau$ unsere Reduktionsrelation ist stabil, also ist $r_1\sigma, C_2(r_2)\sigma$ zf, so auch alle substitutionen.

Satz 6. *wohlfundiert Induktion*

$R \subseteq X \times X$ wohlfundiert $\Longrightarrow$

Wenn $\forall x(\forall y(xRy \Longrightarrow P(y))) \Longrightarrow P(x)$ (1)

(wenn für alle nachfolger von x P(y) gilt, dann gilt auch P(x))

dann gilt $\forall x(P(x))$ (2)

dies heißt wohlfundierte Induktion.

Beweis 5.3. Kontraposition:

zeige $(1) \wedge \neg(2) \Longrightarrow R$ nicht wohlfundiert.

Per $\neg(2)$ ex. x_0 mit $\neg P(x_0)$

$\overset{\Longrightarrow}{(1)}$ ex. x_1 mit x_0Rx_1 $\neg P(x_1)\ldots$

d.h. $x_0Rx_1Rx_2\ldots$, R nicht wf.

(dependent choice, viel harmloser als ZFC's auswahlaxiom)

Beispiel 5.7. 1) $X = \mathbb{N}$ $R = \{(n+1, n) | n \in \mathbb{N}\}$

wohlfundierte Relation (bzw vollständige Relation als wohlfundierte…):

$P(0)$

$\forall n(P(n) \implies P(n+1))$

zusammen liefert das $\forall n(P(n))$

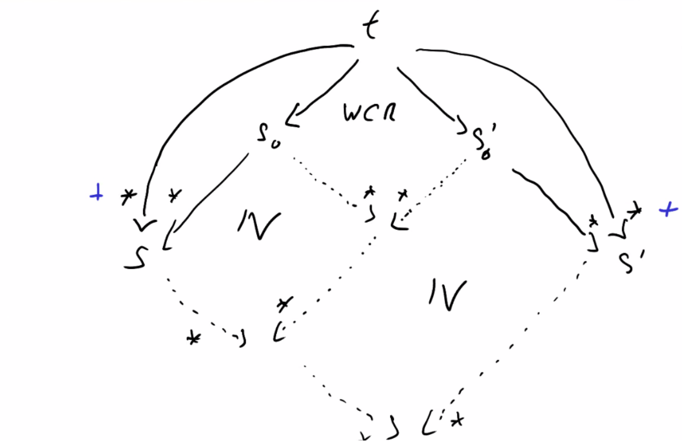
Beispiel 5.8. $X = \mathbb{N}$ $R =>$: Course-of-values-Induktion (man nimmt also für alle echt kleineren n die Aussage an)

Beweis 5.4. Newman's Lemma 4 $SN\&WCR \implies CR$

beweis per wohlfundierter Relation über $\rightarrow$ (ist wf wegen SN).

o.E. $t \rightarrow^+ s, s'$ (weil in 0 schritten reduzieren trivialerweise sofortig zf ist)

Idee: man teilt die schritte zwischen s und s' in jeweils zwei paare von beiden seiten (s_0, s_0') dann Induktionsvorraussetzung anwenden, woraus man folgern kann, dass dies für jeden schritt möglich ist:



6 Der λ-Kalkül

Beispiel haskell

```
twice f x = f $ f x
-- Eigentlich ``Schönfinkelisierung'' nach dem echten Erfinder.
map:: (a-> b)-> (List a-> List b)
map f [] = []
map f (x:xs) =(f x):map(f xs)
```

ungetypter λ-Kalkül = LISP.

getypter λ-Kalküle

Church (Ein zuerst unvollständiges system), Kleene, Rosser (haben beide R, Curry)

7 Der ungetypte λ-Kalkül

Definition 7.1. $\lambda x.t$ "die Funktion, die x auf t abbildet $x \mapsto t$ (wobei üblicherweise $x \in FV(T)$ ist)"

ts Anwendung von s auf t. (Applikation)

<u>Terme</u> t,s gegeben durch:

$$t, s ::= x | ts | \lambda x.t \, (x \in V)$$

Wobei das Zweite als (λ)-Notation.

Beispiel 7.1.

- "$\lambda x.3 + x$" (3 und plus ist technisch gesehen nicht definiert...)
- $\lambda x.xx$ (Haskell würde typfehler liefern, wenn man x als funktion auf sich selbst andwendet, es gibt aber sinnvole kontexte für solche dinge: Wenn x eine berechenbare funktion ist, dann muss es einen bestimmten, Gödelnummerierbaren, funktionsraum geben. Das erste x wäre dann die interpretation als funktion, und die zweite die Gödelnummer)
- $\lambda x.\lambda y.x =: f$, dann wäre z.B. $fxy = (\lambda y.x)y = x$

Definition 7.2. Freie Variablen und konventionen

Kontexte

$$C(\cdot) = (\cdot) | tC(\cdot) | C(\cdot)s | \lambda x.C(\cdot)$$

Kongruenz = Kontextabgeschlossene Äquivalenz.

Notation $\lambda x_1 \ldots \lambda x_n.t = \lambda x_1 \ldots x_n.t$

$tsu = (ts)u$

Scope von λ so weit wie möglich.

$\lambda x.xx = \lambda x.(xx)$ im gegensatz zu $(\lambda x.x)x$

Freie Variablen:

$FV(x) = \{x\}$

$FV(ts) = FV(t) \cup FV(s)$

$FV(\lambda x.t) = FV(t) \setminus \{x\}$

Definition 7.3. Substitution

- $x\sigma = \sigma(x)$
- $(ts)\sigma = (t\sigma)(s\sigma)$
- $(\lambda x.t)\sigma = \lambda y.(t\sigma')$ wobei y eine **frische variable** ist

  (also $y \in FV(\sigma(z)), (z \in FV(t) \setminus \{x\} \iff z \in FV(\lambda x.t))$), sonst könnte x "gefangen werden" $\lambda x.y[x/y] \neq \lambda x.x$!!

  Lösung, wie bei $\forall/\exists$ in GLOIN. (capture-avoiding substitution, liefert hier $\lambda x.y[x/y] = \lambda t.x$, mit $\sigma' = \sigma[x \to t]$)

  de-Broujin indizes. ($\lambda x.\lambda y.xy = \lambda\lambda.2\ 1$) oder nominale Mengen[3]

Definition 7.4. $t =_\alpha s$ (sprich "α-äquivalent")

$\iff$ t geht aus s durch **Umbennenung** gebundener Variablen hervor (ohne Variableneinfang!).

Formal: $=_\alpha$ ist die von

$$\lambda x.t =_\alpha \lambda y.t[y/x]\ (y \notin FV(t) \setminus \{x\})$$

erzeugte Kongruenz

Beispiel 7.2.

$\lambda x.xy =_\alpha \lambda z.zy \neq_\alpha \lambda y.yy$

Lemma 7.1. *$=_\alpha$ ist stabil*

Beweis 7.1. Es reicht: erzeugende Relation ist stabil:

$$R = \{(\lambda x.t, \lambda y.t[y/x]) | y \notin FV(\lambda x.t)\}$$

Sei also $y \notin FV(\lambda x.t)$

zZ: $(\lambda x.t)\sigma R(\lambda y.t[y/x])\sigma$

Daraus folgt (2) $\lambda x'.t\sigma'\ \lambda y'.t[y/x]\sigma''$

Wobei $\sigma' = \sigma[x \mapsto x']$ und $\sigma'' = \sigma[y \mapsto y']$ und x', y' frisch

$[y/x]\sigma'' = \sigma'[y'/x']$

$x \to y \to y' = x \to x' \to y'$

somit ist die Rechte seite $\lambda y'.t\sigma[y'/x']$ mit $y' \notin FV(\lambda x'.t\sigma')$ frisch.

Die Rechte seite is talso gleich der linken in (2)

Satz 7. *β-Reduktion. Operationale Semantik ("Wie sich ein program während der Ausführung verändert")*

Im imperativen gibt es Kontexte: $\eta, (x := 1; c) \to \eta[x \mapsto 1]; c$ (η Umgebung, wie in GLOIN)

In λ-Kalkül gibt es sowas nicht: β-Reduktion als kontextabgeschlossene Umformung

$(\lambda x.3 + x)3 \to 3 + 4$($\to$ *wenn + bekannt ist*)

$\lambda-$Kalkül ist im wesentlichen ein TES (nicht 100% wegen alpha-equiv und gebundenen Variablen)

(β) $(\lambda x.t)x \to_0 t$

$\Longrightarrow$ *Einschrittreduktion* $\to$

$C((\lambda x.t)s) \to C(t[s/x])$

$(\lambda x.t)s$ *heißt β-Redex.[hier nicht: (η) $\lambda x.yx \to_0 y$, beliebt in theoriebetrachtung, aber nicht in programmiersprachen (wenn man Seiteneffekte/IO hat, macht (η) viel kaputt, weil damit die "ausführung" von x auf y umgangen wird)]*

Beispiel 7.3.

- $(\lambda x.xx)(yx) \to_\beta yx(yx)$
- $(\lambda xy.x(yx))zu \to_\beta \lambda y.z(yz)u \to_\beta z(uz)$
- $\omega := \lambda x.xx$, $\omega\omega = (\lambda x.xx)\omega \to_\beta = \omega\omega \to_\beta \dots$ Nicht terminierend.
- Booleans "$x \times x \to x$"

  $true := \lambda xy.x$ $false := \lambda xy.y$

- Paare: "$Paar \equiv Fkt$" $Bool \to x$

  - $fst := \lambda p.ptrue$
  - $snd := \lambda p.pfalse$
  - $pair := \lambda xy.\lambda z.zxy$ wobei "z eine von true/false ist"

  Dies liefert uns:

  $\underline{\underline{fst\ \underline{(pair\ x\ y)}}} \to_\beta fst(\lambda xy.\lambda zxy)xy \to_{2\times\beta} fst(\lambda z.zxy) = (\lambda p.p\ true)\lambda z.zxy \to_\beta (\lambda z.zxy)true \to_\beta true\ xy = (\lambda xy.x)xy \to_\beta (\lambda y.x)y \to_\beta x$

7.1 Rekursion

$fact = \lambda n.$if n=0 then 1 else n*fact(n-1)

Dieser Aufruf besteht aus einer primitiven rekursionsfunktion F und der funktion selbst.$fact = F\ fact$

$F = \lambda f.\lambda n.$if n=0 then 1 else n*f(n-1) F nennt man auch ein Funktional.

$fact = F\ fact$ nennt man Fixpunktgleichung. (rekursive Funktionen sind Fixpunktgleichen)

Fixpunktkombinator fix:

fix F = F(fix F)

Satz 8. *λ-Kalkül Fixpunktkombinator*

1) Jedes t hat einen Fixpunkt s, d.h. $s \to_\beta ts$ (also die Reduktion liefert wieder ts auf dem wider reduziert werden kann, ad absurdum)

2) Es existiert ein Fixpunktkombinator Y, d.h. $Yt \to_\beta s$ s ist Fixpunkt von t

$Yt \to_\beta s \overset{(1)}{\to}_\beta ts$

Beweis 7.2.

1) $s = W_t W_t, W_t = \lambda x.t(xx)$ (wie oben bei $\omega\omega$, bloß mit t ausenrum):

$s = W_t W_t = (\lambda x.t(xx))W_t \to_\beta t(W_t W_t) = ts$

2) $Y = \lambda f.W_f W_f$ wenn man das jetzt auf ein f anwendet erhält man genau $fs = s$

Beispiel 7.4. Der Fall von oben $\lambda x.xx = \omega$ und dann $\omega\omega$ hat die funktion $t = \omega$ terminiert deshalb nicht.

$\lambda x.((\lambda y.y)(xx))$ jetzt $t = \lambda y.y$ (also rekursion über die Identitätsfunktion) und s wäre dann $\omega\omega \to_\beta t(\omega\omega) \to_\beta \omega\omega$

7.2 Auswertungsstrategie

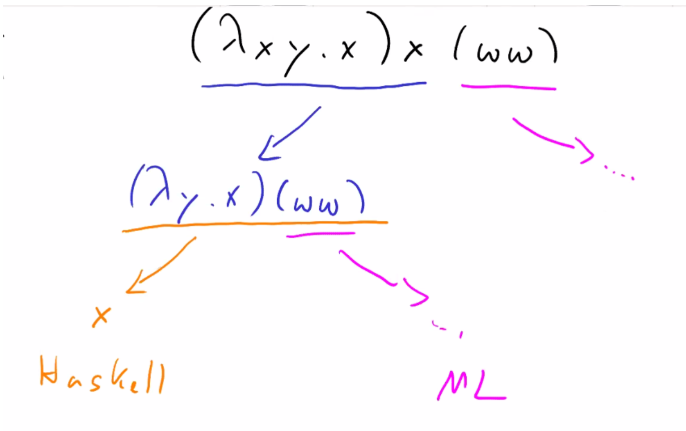
Beispiel 7.5.

$(\lambda xy.x)x(\omega\omega)$ Wenn man probiert zuerst $\omega\omega$ zu reduzieren, läuft man undendlich weiter.

Wenn man den rechten reduziert erhält man:

$(\lambda y.x)(\omega\omega)$ wo man entweder wieder ad absurdum $(\omega\omega)$ reduzieren kann (ML, leftmost-innermost, applikativ), oder das ganze zerlegen in:

$\lambda y.x\omega\omega = x$ (Haskell, leftmost-outermost, normal/standard)



Definition 7.5. applikative (leftmost-innermost) Reduktion $\rightarrow_a$

induktiv definiert durch:

1a)$(\lambda x.t)s \rightarrow_a t[s/x]$ **wenn** t,s normal (innermost, eager).

2a) $(\lambda x.t \rightarrow_a \lambda x.t')$, wenn $t \rightarrow_a t'$ (eine echte prog. sprache macht aber niemals Termreduktionen unter einem lambda)

3a)$ts \rightarrow_a t's$ wenn $t \rightarrow_a t'$

4a)$ts \rightarrow_a ts'$, wenn $s \rightarrow_a s'$ und t normal.

Definition 7.6. normale (leftmost-outermost) Reduktion $\rightarrow_n$

1n) $(\lambda x.t)s \rightarrow_n t[s/x]$ immer (outermost reinziehen)

2n) $\lambda x.t \rightarrow_n \lambda x.t'$, wenn $t \rightarrow_n t'$

3n) $ts \rightarrow_n t's$, wenn $t \rightarrow_n t'$ und t keine λ–Abstraktion. (wenn es eine wäre, dann 1. Regel)

4n) $ts \rightarrow_n ts'$ wenn $s \rightarrow_n s'$ und t normal und keine λ-Abstraktion.

Beispiel 7.6. $(\lambda y.x)(\omega\omega)$

mit normaler Reduktion:

$(\lambda y.x)(\omega\omega) \rightarrow_{1n} x$

mit Applikativer Reduktion:

$(\lambda y.x)(\omega\omega) \rightarrow_{4.a} (\lambda y.x)(\omega\omega)$

Satz 9. *Standardisierungssatz:*

Sei $t \rightarrow^* s$ *s Normalform* $\implies t \rightarrow_n^* s$

(Nebenbemerkung: $(\lambda x.fxx)t \rightarrow_n ftt \rightarrow_n fst \rightarrow_n fss$ *und* $(\lambda x.fxx)t \rightarrow_a (\lambda x.fxx)s \rightarrow_a fss$ *also geht es mit applikativen schneller, weil man funktionen nur 1 mal evaluieren muss)*

8 Der einfach getypte λ-Kalkül

Typen α, β:

$\alpha \rightarrow \beta$ Funktion von α nach β

Typvariablen a,b,...

z.B. $\lambda x.x : a \to a$

Definition 8.1.

Gegebene Menge **V** var.

Typvariablen **B** von Basistypen.

(**bool**, **Int**, ...) sind typen $\alpha, \beta, \ldots$

definiert durch

$$\alpha, \beta ::= a|\mathbf{b}|\alpha \to \beta \ (a \in \mathbf{V}, \mathbf{b} \in \mathbf{B})$$

z.B. $a \to (b \to a) = (a \to b) \to a = a \to b \to a$ Terme: Church: $\lambda x : \alpha.t$ nur typkorrekte Terme. (also termbildung und typisierung)

Curry: $\lambda x.t$, Term kann typisierbar sein oder nicht.

$\omega = \lambda x.xx$ z.B. nicht typisierbar ($\lambda \to$, weil typ voll rekursiv ist)

wir benützen curry.

$x\lambda y.y$? weil x unbekannt/untypisiert.

Definition 8.2. kontexte

Ein Kontext ist eine endliche Menge Γ von Typisierungsannahmen $x : \alpha (x \in V)$ "x hat typ α"

Schreibweise: $\Gamma, x : \alpha = \Gamma \cup \{x : \alpha\}$

d.h. Γ ist eine endliche partielle Abbildung. (von variablennamen auf typen)

Typisierungsurteile (typing judgements) $\Gamma \vdash t : \alpha$ "in Kontext Γ hat t Typ α" z.B. $f : a \to b, x : a \vdash fx : b$

Herleitbarkeit induktiv:

(Ax) $\dfrac{}{\Gamma \vdash x : \alpha}$ $(x : \alpha) \in \Gamma$

$(\to_e)$ $\dfrac{\Gamma \vdash t : \alpha \to \beta \qquad \Gamma \vdash s : \alpha}{\Gamma \vdash ts : \beta}$

$(\to_i)$ $\dfrac{\Gamma[x \mapsto \alpha] \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \to \beta}$ (also wenn x schon einen typ hat, wird dieser Überschrieben, shadowing)

Rechts: sonst könnte $x : \alpha$ zum clash führen! (in der Realität könnte man das lösen, indem man aus Γ eine List statt eine Menge baut)

Beispiel 8.1.

$$\dfrac{\dfrac{\dfrac{Ax \dfrac{}{\vdash x : a \to b, y : a \vdash x : a \to b, x : a \to b, y : a \vdash y : a} AX}{\vdash x : a \to b, y : a \vdash xy : b} \to_i}{\vdash x : a \to b \vdash \lambda y.xy : a \to b} \to_i}{\vdash \lambda xy.xy : (a \to b) \to (a \to b)}$$

$$\rightarrow_e \dfrac{\rightarrow_i \dfrac{x : a \rightarrow \vdash x : a \rightarrow x : a \rightarrow \vdash x : a}{x :\vdash xx}}{\vdash \lambda x.xx :}\ \text{CIRCULAR DEPENDENCY}$$

Berechnungsprobleme:

- gilt $\vdash t : \alpha$? (typcheck)
- finde (existiert?) α mit $\vdash t : \alpha$ (Typinferenz)
- finde (existiert?) t mit $\vdash t : \alpha$ (Type inhabitation)

Beispiel 8.2. $a \rightarrow a$ inhabited (Identitätsfunktion $\lambda x.x$, bildet typ auf sich selbst ab, nach curry-howard tautologie)

a nicht inhabited (also für sich stehend hat ein wert nicht irgendeinen typ, nicht obdA gültig)

$(a \rightarrow a) \rightarrow a$ nicht inhabited (das erste ist eine Tautologie, also immer wahr, a selbst ist aber nicht immer wahr)

denn: wäre $\vdash t : (a \rightarrow a) \rightarrow a$, dann $t(\lambda x.x) : a$ widerspruch! (dependent Types a'la idris/agda, Programmsynthese, automatisches Beweisen)

Eigenschaften:

c frisch $\dfrac{\phi(c)}{\forall(\phi)}\ \forall I$

$\dfrac{\phi \vdash \psi}{\phi \rightarrow \psi} \rightarrow I$

c frisch $\dfrac{\phi(c) \vdash \psi(c)}{\forall x(\phi \rightarrow \psi)}$ herleitbar:

Beweis:

$\forall I\ \dfrac{\dfrac{\phi \vdash \psi}{\phi(c) \rightarrow \psi(c)}}{\forall x(\phi \rightarrow \psi)}$ c frisch

Regel zulässig $\iff$ durch ihre Hinzunahme wird nichts neu herleitbar.

Lemma 8.1. *(Weakening)*

(wk) $\dfrac{\Gamma \vdash t : \alpha}{\Gamma' \vdash t : \alpha}\ \Gamma \subseteq \Gamma'$

(also ein größerer Kontext ändert nichts an der Herleitbarkeit)

Beweis 8.1. Induktion über Herleitung von $\Gamma \vdash t : \alpha$

$(Ax)\Gamma \vdash x : \alpha, x : \alpha \in \Gamma \implies x : \alpha \in \Gamma' \implies \Gamma' \vdash x : \alpha$ $(\rightarrow_i)\ \dfrac{\Gamma[x \mapsto \alpha] \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \rightarrow \beta}$

(*) Nach IV. (prämisse ist kleineres Gamma als konklusion) $\Gamma'[x \mapsto \alpha] \vdash t : \beta$

da $\Gamma[x \mapsto \alpha] \subseteq \Gamma'[x \mapsto \alpha]$

Sei $y : \beta \in \Gamma[x \mapsto \alpha]$ (also ein beta ist links, so muss es auch rechts sein)

Fall 1: $y \neq x \implies y : \beta \in \Gamma \implies y : \beta \in \Gamma' \implies y : \beta \in \Gamma'[x \mapsto \alpha]$

Fall 2: $y = x \implies \beta = \alpha \implies y : \beta = x : \alpha \in \Gamma[x \mapsto \alpha]$

per (*) $\rightarrow_i \Gamma' \vdash \lambda x.t : \alpha \rightarrow \beta$ (die prämisse gilt, also kann man auch die gleiche folgerung machen)

Lemma 8.2. *Inversion*

(man kann alle Regeln auch umdrehen)

- *1)* $\Gamma \vdash x : \alpha \implies (x : \alpha) \in \Gamma$ *(ax inversion)*

- *2)* $\Gamma \vdash ts : \beta \implies$ *es existiert* α *mit* $\Gamma \vdash t : \alpha \to \beta$ *(*$\to_e$ *inversion, also wenn es eine Anwendung gibt, dann muss es eine Funktion dazu gegeben haben)*

- *3)* $\Gamma \vdash \lambda x.t : \gamma \implies \gamma$ *hat die Form* $\alpha \to \beta$ *und* $\Gamma[x \mapsto \alpha] \vdash t : \beta$ *(*$\to_i$ *inversion, also der type des input einer Funktion muss herleitbar sein)*

Beweis 8.2. Regeln sind syntaxgerichtet

8.1 Typinferenz

$\lambda x.x : a \to a$

$\lambda x.x : (a \to b) \to (a \to b)$

Offensichtlich ist das erst besser als das zweite. Es muss also eine "Algemeinheitshierarchie" geben (most general typing)

Definition 8.3. Terminologie/Notation:

$TV(\alpha)$ = Menge der in α vorkomenden Typvariablen.

$TV(\Gamma) = \bigcup_{(x:\alpha)\in\Gamma} TV(\alpha)$

Typsubstitution = Substitution von Typen für Typvariablen σ Lösung von $\Gamma \vdash t : \alpha$, wenn $\Gamma\sigma \vdash t : \alpha\sigma$ herleitbar.

allgemeinste Lösung (wie bei mgu $\sigma' = \sigma\theta$ dann ist σ das allgemeinere, wenn das $\forall\sigma'$ gilt dann ist σ die allgemeinste Lösung)

Prinzipaltyp von $\Gamma \vdash t$ = allgemeinste Lösung von $\Gamma \vdash t : a$ a frisch ($a \notin TV(\Gamma)$) (prinizipaltyp ist allgemeinste Lösung mit frischen typen und eindeutig modulo Umbennenung)

$\Gamma \vdash t$ typisierbar $\iff \Gamma \vdash t : a$ hat eine Lösung. (a frisch)

Satz 10. *Algorithmus W nach HINDLEY/MILNER*

Berechne zu $\Gamma \vdash t : \alpha$ *("Ziel")*

$PT(\Gamma; t; \alpha)$ *Menge von Typgleichungen* $a \doteq \beta$ *mit* $PT(\Gamma; t; \alpha)$ *unfizierbar* $\iff \Gamma \vdash t : \alpha$ *hat Lösung.*

$mgu(PT(\Gamma; t; \alpha))$ *liefert allgemeinste Lösung von* $\Gamma \vdash t : \alpha$ *(liefert und ist nicht gleich, weil der PT mehr variablen substituiert als notwendig)*

$\implies mgu(PT((); t; a))(a)$ = *Prinzipaltyp von t (a frisch, t geschlossen)*

Implizit geht man bei diesen Regeln immer von $x \in \Gamma$ *aus*

- $PT(\Gamma; x; \alpha) = \{\alpha \doteq \beta | x : \beta \in \Gamma\}$ *(nach Ax inversionslemma)*
- $PT(\Gamma; ts; \alpha) = PT(\Gamma; t; a \to \alpha) \cup PT(\Gamma; s; a)$ ***global***$(a\ frisch)$ *nach* $(\to_e)$
- $PT(\Gamma; \lambda x.t; \alpha) = PT(\Gamma[x \mapsto a]; t, b) \cup \{a \to b \doteq \alpha\}$ *mit a,b* ***global*** *frisch* $(\to_i)$ *invers. (wir fitten also input auf a und output auf b)*

Das global ist notwendig, um einfang bei unifikation zu vermeiden. Lösung z.b. über [3]

Beispiel 8.3.

$\vdash \lambda xy.xy$

$PT(\emptyset, \lambda xy.xy; a) = PT(x : b, \lambda y.xy, c) \cup \{a \doteq b \to c\} =$

$PT(x : b, y : d; xy; e) \cup \{a \doteq b \to c, c \doteq d \to e\}$

$PT(x : b, y : d; x; f \to e) \cup PT(x : b, y : d; y; y : f) \cup \{a \doteq b \to c, c \doteq d \to e\}$

$= \{b \doteq f \to e, y \doteq f, a \doteq b \to c, c \doteq c = d \to e\}$

jetzt hat man gleichungen, die man unifizieren muss:

$mgu = [f/d, f \to e/c, f \to e/b, (f \to e) \to (f \to e)/a]$ wir haben oben mit a angefangen, also ist der endtyp $\lambda xy.xy :$ $(f \to e) \to (f \to e)$ ist Prinzipaltyp.

$PT(x : a; x\lambda z.z; c) = PT(x : a; x; b \to c) \cup PT(x : a, \lambda z.z; b) = \{a \doteq b \to c\} \cup PT(x : a, z : d; z; e) \cup \{b \doteq d \to e\} = \{a \doteq b \to c, d \doteq e, b \doteq d \to e\}$

$mgu = [e \to e/b, e \to e \to c/a, c/c]$ "$e \to e \to c/a, c/c$" also ist Prinzipaltyp.

$PT(\emptyset; \lambda x.xx, a) = PT(x : b; xx; c) \cup \{a \doteq b \to c\} = PT(x : b; x; d \to c) \cup PT(x : b; x; d) \cup \{a \doteq b \to c\} = \{b \doteq d \to c, b \doteq d, \dots\}$

Unifikation liefert occurs $\perp$ nach substitution [d/b]

Satz 11. (Γ, t) *typisierbar* $\iff$ $PT(\Gamma; t; a)$ *(a frisch) unifizierbar; dann* $mgu(PT(\Gamma; t; \alpha))|_{TV(\Gamma) \cup \{a\}}$ *Prinzipaltyp von* $(\Gamma; t)$

Beweis 8.3. Zeige allgemeiner:

$$PT(\Gamma; t; \alpha) \text{ unifizierbar } \iff \Gamma \vdash t : \alpha \text{ lösbar}$$

dann

$$mgu(PT(\Gamma; t; \alpha))|_{TV(\Gamma, \alpha)}$$

allgemeinste Lösung von $\Gamma \vdash t : \alpha$

Zeige dazu:

$$\Gamma\sigma \vdash t : \alpha\sigma \iff \sigma \text{ ist erweiterbar zu } \sigma' \in Unif(PT(\Gamma; t; \alpha))$$

d.h. $\sigma'|_{TV(\Gamma, \alpha} = \sigma$ sigma ist also erweiterbar.

per Induktion über t:

"$\Leftarrow$": per Typregeln(8.2).

z.B. $t = \lambda x.s$:

haben $\sigma' \in Unif(PT(\Gamma[x \mapsto a]; s; b) \cup \{\alpha \doteq a \to b\})$

Nach IV. $\underbrace{\Gamma[x \mapsto a]\sigma'}_{=\Gamma[x \mapsto a]\sigma} \vdash s : \underbrace{b\sigma'}_{b\sigma}$

Per $(\to_e)$ $\dfrac{\Gamma[x \mapsto a]\sigma \qquad s : b\sigma}{\Gamma\sigma \vdash \lambda x.s : a\sigma' \to b\sigma'}$

Daraus folgt $\alpha\sigma' = \alpha\sigma$

" $\implies$ " Per Inversion:

1.$\Gamma\sigma \vdash x : \alpha\sigma \overset{inversion}{\implies} x : \beta \in \Gamma, \alpha\sigma = \beta\sigma$

$\implies \sigma \in Unif(\underbrace{PT(\Gamma; x; \alpha)}_{\alpha \doteq \beta})$

2. $\Gamma\sigma \vdash ts : \alpha\sigma \overset{Inversion}{\implies}$

es existiert ein γ mit $\Gamma\sigma \vdash t : \gamma \to \alpha\sigma, \Gamma\sigma \vdash s : \gamma$

Setze $\sigma' = \sigma[a \mapsto \gamma]$ (a global frisch) $\implies \Gamma\sigma' \vdash t : (a \to \alpha)\sigma', \Gamma\sigma' \vdash s : a\sigma'$

$\overset{IV}{\implies} \sigma'$ erweitert zu $\underbrace{\sigma'' \in Unif(PT(\Gamma; t; a \to \alpha))}_{\text{gemeinsame TV sind nur die aus} TV(\Gamma;\alpha;a)}$ und $\sigma'' \in Unif(PT(\Gamma; s; a))$

$\implies \sigma'' \in Unif(PT(\Gamma, ts, \alpha))$

3. $\Gamma\sigma \vdash \lambda x.s : \alpha\sigma \overset{Inversion}{\implies}$

$\alpha\sigma = \beta \to \gamma\ \Gamma\sigma[x \mapsto \beta] \vdash s : \gamma$

Setze $\sigma' = \sigma[a \mapsto \beta, b \mapsto \gamma]$, a,b global frisch

$\implies \Gamma[x \mapsto a]\sigma' \vdash s : b\sigma'$

$\overset{IV}{\implies} \sigma'$ erweitert zu $\sigma'' \in Unif(PT(\Gamma[x \mapsto a]; s; b))$

$\overset{(a \to b)\sigma'' = \alpha\sigma''}{\implies} \sigma'' \in Unif(PT(\Gamma; \lambda x.s; \alpha))$

(Das letzte folgt daraus, dass $\alpha\sigma = \beta \to \gamma$ ist und wir $\sigma' = \sigma[a \mapsto \beta, b \mapsto \gamma]$ haben)

8.2 Subjektreduktion

Satz 12. $\Gamma \vdash t : \alpha, t \to_\beta s \implies \Gamma \vdash s : \alpha$

⚠ *"$\Longleftarrow$" gilt **NICHT**: z.B. $t = (\lambda x.y)(\lambda x.xx) \to_\beta y$*

Lemma 8.3. *Substitution*

$\Gamma[x \mapsto \alpha] \vdash t : \beta, \Gamma \vdash s : \alpha \implies \Gamma \vdash t[s/x] : \beta$

Beweis: induktion über t.

Beweis 8.4. $t = C((\lambda x.u)v), s = C(v[u/x])$

Induktion über $C(\cdot)$

z.B.: $C(\cdot) = (\cdot)$ Per inversion:

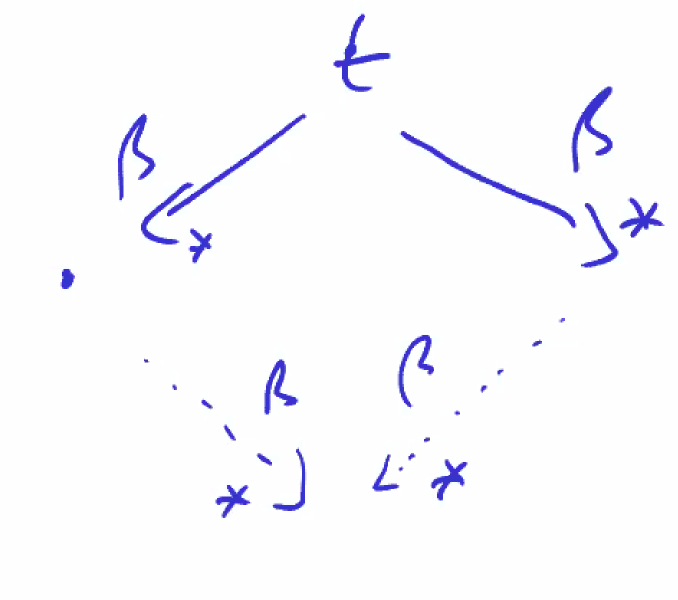
$\Gamma \vdash \lambda x.u : \beta \to \alpha, \Gamma \vdash v : \beta$

$\Gamma[x \mapsto \beta] \vdash u : \alpha$

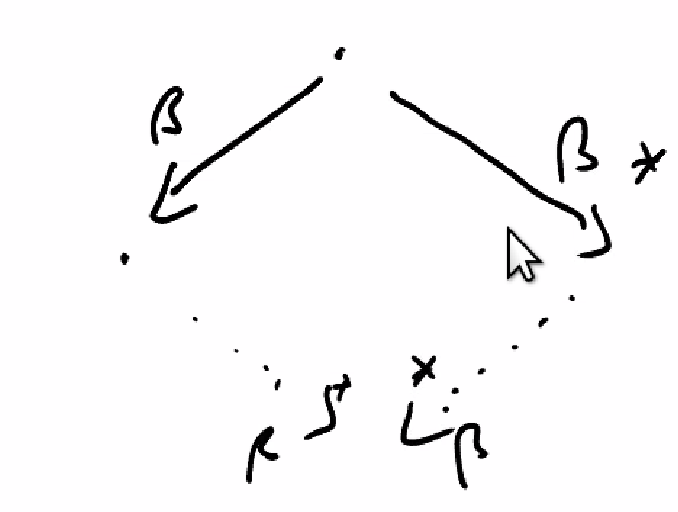
$\overset{subst-lemma}{\implies} \Gamma \vdash \underbrace{u[v/x]}_{s} : \alpha$

9 Church Rosser des λ-Kalkül

(Dies ist der Ursprüngliche Church-Rosser beweis, der Name wird jetzt mittlerweile für alle TES benutzt).

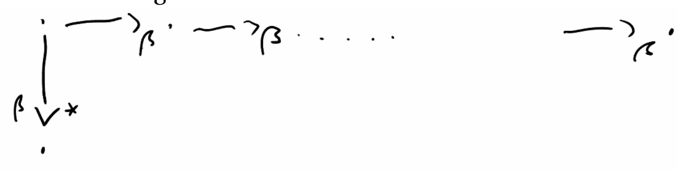


Lemma 9.1. *Streifenlemma (strip-lemma)*



Effektiv ein mittelweg zwischen WCR und CR.

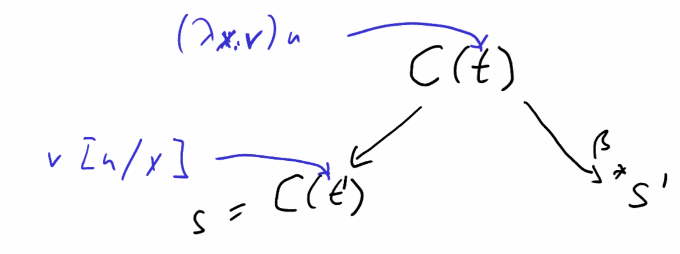
Daraus folgt CR:



oben hat man die n-beta-reduktionen. Unten macht man jeweils immer einen Schritt und kaskadiert nach links.

Dies geht für jedes TES.

Speziell für lambda sähe das so aus.



markierte Terme: $t, s ::= x | ts | \lambda x.t | (\underline{\lambda} x.t)s$ (also genau gleich, man kann nur **beta-reduzierbare** terme markieren).

Diese unterstriche müssen wieder “unlocked” werden.

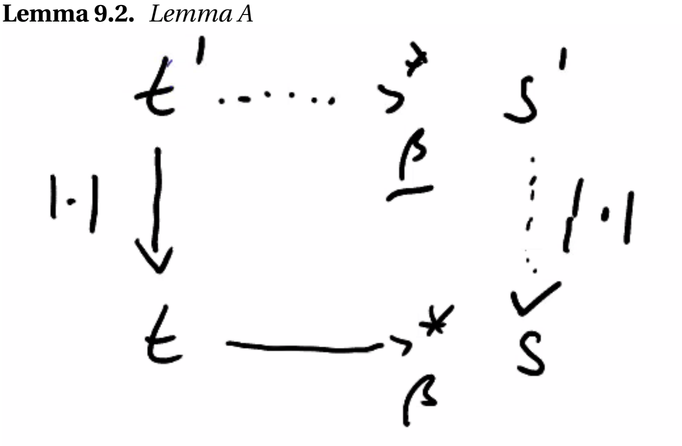
$|t|$: Entfernt _ aus t, z.B. $|(\underline{\lambda} x.t)s| = (\lambda x.|t|)|s|$

$\phi(t)$ Reduziere alle unterstrichenen Redexe:

- $\phi(x) = x$
- $\phi(ts) = \phi(t)\phi(s)$
- $\phi(\lambda x.t) = \lambda x.\phi(t)$
- $\phi((\underline{\lambda} x.t)s) = \phi(t)[\phi(s)/x]$

Syntaktic sugar $t \xrightarrow{|\cdot|} |t|$

Lemma 9.2. *Lemma A*



Dabei $(\lambda x.t)s \to_{\underline{\beta}} t[s/x]$ $(\underline{\lambda} x.t)s \to_{\underline{\beta}} t[s/x]$ *Es ignoriert also die Unterstriche.*

Beweis 9.1. Reduziere auf $t \to_\beta s$, Induktion über Kontexte

Wir bekommen also nicht mehr dazu, oder verlieren ausdruckskraft.

Lemma 9.3. *Syntaktisches Substitutionslemma*

$$u[v/y][s/x] = u[s/x][v[s/x]/y]$$

nicht simultan! Das v wird vom x beeinflusst.

wenn $y \notin FV(s), x \neq y$ *(bei dem ersten hätte man eine doppelsubstitution, beim zweiten wird die zweite reduktion void)*

Beweis 9.2. Induktion über u (weil in diesen Reinsubstituiert wird)

Der interessante Fall ist hier der Induktionsanfang, weil im I.S nur weitergereicht wird.

Sei n eine Variable. (LHS/RHS =Left/Right handed side)

$u \notin \{x, y\}$✓

$u = x : LHS = s, RHS = s$ weil $y \notin FV(s)$

$u = y : LHS = v[s/x], RHS = v[s/x]$

Lemma 9.4. *Lemma B*

a) $\phi(t[s/x]) = \phi(t)[\phi(s)/x]$

b) ϕ *bewahrt* β

b)
$$\begin{tikzcd} t \arrow[r, "*"', "\underline{\beta}"] \arrow[d] & s \arrow[d] \\ \varphi(t) \arrow[r, dotted, "*", "\beta"'] & \varphi(s) \end{tikzcd}$$
„φ bewahrt β"

Beweis 9.3. .

a) Induktion über t, interessant nur $t = (\underline{\lambda} y.u)v$

$\phi(((\underline{\lambda} y.u)v)[s/x]) \overset{(1)}{=} \phi((\underline{\lambda} y.u[s/x])v[s/x]) = \phi(u[s/x])[\phi(v[s/x])/y]$

$\overset{IV}{=} \phi(u)[\phi(s)/x][\phi(v)[\phi(s)/x])/y]$

$\overset{synt.Subst+(1)}{=} \underbrace{\phi(u)[\phi(v)/y]}_{\phi((\underline{\lambda} y.u)v)}[\phi(s)/x]$

$= \phi((\underline{\lambda} y.u)v)[\phi(s)/x]$

(1) o.E. $y \neq x$ $y \notin FV(s)$ (im zweifelsfall umbennenen)

b) Reduziere auf $t \rightarrow_{\underline{\beta}} s$, Induktion über Kontexte:

$(\cdot)$: Fälle nach Markierung:

Fall 1: markiert:

$$\begin{bmatrix} (\underline{\lambda} x.u)v & \rightarrow_{\underline{\beta}} & u[v/x] \\ \phi & & \phi \\ \phi(u)[\phi(v)/x] & = & \phi(u)[\phi(v)/x] \end{bmatrix}$$

Fall 2: unmarkiert:

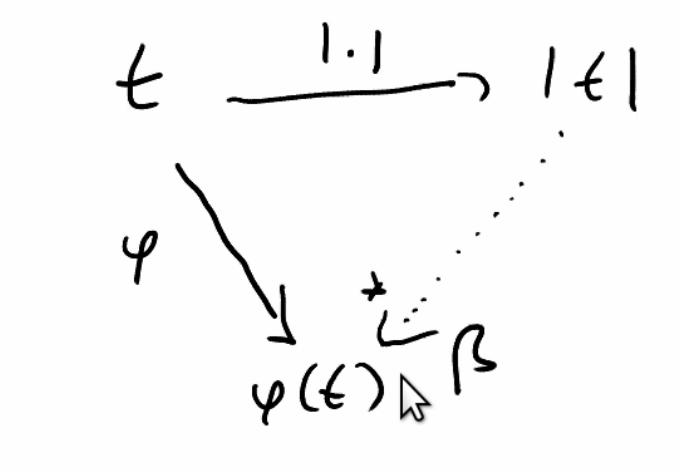
$$\begin{bmatrix} (\lambda x.u)v & \rightarrow_{\underline{\beta}} & u[v/x] \\ \phi & & \phi \\ \lambda x.\phi(u)\phi(v) & \rightarrow_{\beta} & \phi(u)[\phi(v)/x] \end{bmatrix}$$

Beispiel mit nichtleerem kontext:

Kontext $(\underline{\lambda} x.C(\cdot))s$ und $t \rightarrow_{\beta} s$

$\phi(\underline{\lambda} x.C(t))s = \phi(C(t))[\phi(s)/x] \xrightarrow[\beta]{*}{}^{IV,C\ ist\ kleinerer\ Kontext} \phi(C(s))[\phi(s)/x]$✓

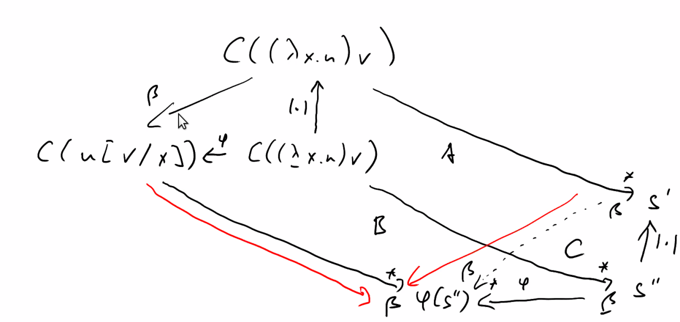
Lemma 9.5. *Lemma C*



Beweis 9.4. Induktion über t, interessanter Fall $t = (\underline{\lambda} x.u)\, v$ (sonst passiert ja nichts):

$$\begin{bmatrix} (\underline{\lambda} x.u)\, v & \xrightarrow{|\cdot|} & (\lambda x.|u|)|v| \\ \phi & & \phi \\ \phi(u)[\phi(v)/x] & {}_{\beta}\!\leftarrow & (\lambda x.\phi(u))\phi(v) \end{bmatrix}$$

$C((\lambda x.u)\, v)$



10 Curry-Howard Isomorphismus

Typen = Propositionen (=Formel) "Types are propositions".

Terme/Programme = Beweise (u.U. ist ein Typ nicht bewohnt, also nicht Beweisbar)

Coq "man hat einen Term definiert, der diesen Typ hat"

<u>minimale Logik:</u>

$$\phi, \psi ::= a | \phi \to \psi \ (a \in V)$$

Deduktion:

$(\to_E)\ \dfrac{\phi \to \psi \qquad \phi}{\psi}$

$(\to_I)\ \dfrac{[\phi] \vdash \psi}{\phi \to \psi}$

Sequenzenkalkül: $\overset{Annahme}{\underbrace{\Gamma}_{MengeFormeln}} \vdash \underbrace{\phi}_{Formel}$ "linke Regel" $(\to_E)$

$$(\to_E)\ \frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

$$(\to_I)\ \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}$$

$$(Ax)\ \frac{}{\Gamma \vdash \phi}$$

Satz 13. *C-H-I:*

$\vdash \phi \iff \phi$ *bewohnt*

Beweis 10.1. .

$\Longleftarrow$ trivial: lösche Terme aus Herleitung von $\vdash t : \phi$ ergibt Herleitung $\vdash \phi$ (damit erhält man die kombination der ϕ als typen, dies sind trivialerweise die Herleitung)

$\Longrightarrow$ Zu Γ definiert Kontext $\bar{\Gamma}$:

$\Gamma = \{\phi_1, \ldots, \phi_n\} \implies \bar{\Gamma} = \{x_1 : \phi_1, \ldots, x_n : \phi_n\}$

Also jeder Term wird als Typ gewertet und bekommt eine Variable.

Zeige $\Gamma \vdash \psi \implies \exists t(\bar{\Gamma} \vdash t : \psi)$

per Induktion über Herleitungen von $\Gamma \vdash \psi$

(Ax): $\phi \in \Gamma \implies \psi = \phi_i$ für ein $x_i : \phi_i \in \bar{\Gamma} \overset{(AX)}{\Longrightarrow} \bar{\Gamma} \vdash x_i : \phi_i$

$(\to_E)$

$$(\to_E)\ \frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

I.V. haben t,s mit

$$\frac{\bar{\Gamma} \vdash t : \phi \to \psi \qquad \bar{\Gamma} \vdash s : \phi}{\bar{\Gamma} \vdash ts : \psi}$$

$(\to_I)$ Beweis

$$(\to_I)\ \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \psi \to \phi}$$

I.V. haben (bei $\phi = \phi_{n+1}$)

$$(\to_I)\ \frac{\bar{\Gamma}, x_{i+1} : \phi_{i+1} \qquad \vdash t : \psi}{\bar{\Gamma} \vdash \lambda x_i.t : \phi_{i+1} \to \psi}$$

⚠**"→" ist intuitionistisch!**

z.B. $((a \to b) \to a) \to a$ ist klassisch gültig, aber in miminaler (intuitionistischer) Logik nicht herleitbar.

$$(\to_I)\ \frac{(\to_E)\ \dfrac{(\text{Ax})\ \dfrac{}{(a \to a) \vdash (a \to a) \to a} \qquad \dfrac{(\text{Ax})\ \dfrac{}{(a \to a) \to, a \vdash a}}{(a \to a) \to a \vdash a \to a}}{(a \to a) \to a \vdash a}}{\vdash ((a \to a \to)a) \to a}$$

Auf Typebene:

$\underbrace{((a \to a) \to a) \to a}_{\lambda f.f(\lambda x.x)}$

Man hat eine funktion, die wieder eine funktion erhält also das innere $(a \to a) \to a$ und hat als Ergebnis wieder ein a. Dazu muss man anwenden. Die innerer funktion muss den typ $a \to a$ haben. Das einzig mögliche hierfür ist die identitätsfunktion $\lambda x.x$

Deshalb haben wir eine funktion f in den wir eine funktion (identität) haben.

11 Induktive Datentypen

```
data Nat where
    0: ()-> Nat
    Suc: Nat->Nat
```

definiert Signatur $\Sigma_{Nat} = \{0/0, Suc/1\}$

Die Semantik von Nat ist definiert als

$[\![Nat]\!] = \{0, Suc(0), Suc(Suc(0)), \ldots\}$ also ein Herbrandmodell.

Definition 11.1. Ein Σ-Modell (Σ-Algebra) $\mathfrak{M}$ besteht aus

- Menge M (Träger)
- zu f/n∈ Σ
- $\mathfrak{M}[\![f]\!] : M^n \to M$

Interpretion von Termen unter <u>Umgebung</u> $\eta : V \to M$:

$$\mathfrak{M}[\![t]\!]\eta \in M$$

$$\mathfrak{M}[\![x]\!]\eta = \eta(x)\ x \in V$$

$$\mathfrak{M}[\![f(t_1, \ldots, t_n)]\!]\eta = \mathfrak{M}[\![f]\!](\mathfrak{M}[\![t_1]\!]\eta, \ldots, \mathfrak{M}[\![t_n]\!]\eta)$$

Beispiel 11.1. $[\![Nat]\!]$ ist eine Σ_{Nat}-Algebra per:

$$[\![Nat]\!][\![0]\!] = 0$$

$$[\![Nat]\!][\![succ]\!](x) = x + 1$$

Literatur

[1] Knuth-bendix algorithm for creating CR TES from terminating TES

https://en.wikipedia.org/wiki/Knuth%E2%80%93Bendix_completion_algorithm

[2] dependant choice

https://de.wikipedia.org/wiki/Axiom_der_abh%C3%A4ngigen_Auswahl

[3] nominale Mengen

https://www.tcs.ifi.lmu.de/mitarbeiter/martin-hofmann/publikationen-pdfs/c43-nominalrenamingsets.pdf