Lambda-Kalkül

Andreas Abel
Ludwig-Maximilians-Universität München

Rohfassung vom 17. November 2010

Skript zur Vorlesung *Lambda-Kalkül*. Weiteres Material ist bereitgestellt unter `http://www.tcs.ifi.lmu.de/lehre/WS06-07/Lambda/` und `http://www.tcs.ifi.lmu.de/lehre/WS`

Kommentare und Verbesserungsvorschläge bitte an `abel@tcs.ifi.lmu.de`.

Dieses Skript ist in Bearbeitung. Die Präsentation des Stoffes ist unvollständig und höchstwahrscheinlich fehlerbehaftet. Zitate verwandter Arbeiten fehlen. Bitte dieses Dokument nicht zitieren.

Inhaltsverzeichnis

Teil I

Der ungetypte Lambda-Kalkül

Rohfassung vom 17. November 2010

Kapitel 1

Syntax und Gleichheit

In diesem Kapitel definieren wir die Syntax dem λ-Kalküls und die β-Gleichheit. Anschaulich entspricht dieser Gleichheitsbegriff der Gleichheit von $4 + 1$ und 5 in der Zahlentheorie. Mit β-Reduktion geben wir der Gleichheit eine Richtung, und mit Hilfe der Konfluenz erhalten wir eine Methode, die Gleichheit zweier Terme systematisch zu bestimmen.

1.1 Variablen, Terme, α-Äquivalenz

Sei V eine abzählbare Menge von Variablennamen.

Definition 1.1 (Lambda-terme) λ-Terme *sind Bäume, erzeugt durch folgende Grammatik:*

$$\begin{array}{lcll} r, s, t & ::= & x & \textit{Variable } x \in \mathsf{V} \\ & | & \lambda x t & \textit{Abstraktion der Variable } x \textit{ in Term } t \\ & | & r\, s & \textit{Applikation von Term } r \textit{ auf Term } s \end{array}$$

Dabei ist x ein Blattknoten, beschriftet mit Variable x, $\lambda x t$ ist ein unärer Knoten, der mit λ und einer Variable x beschriftet ist und ein Kind t besitzt, und $r\, s$ ein binärer Knoten mit Kindern s und t. Die Menge der λ-Terme bezeichnen wir mit Λ.

Lineare Notation. Wir schreiben Terme in linearer Form, zur Disambiguierung verwenden wir Klammern. Dabei ist Applikation links-assoziativ, d.h., $r\, s\, t$ bedeutet $(r\, s)\, t$, Anwendung der Funktion r auf das Argument s, wobei das Ergebnis dieser Anwendung eine Funktion ist, die dann auf t angewendet wird. Der umgeklammerte Term $r\, (s\, t)$ hat eine völlig andere Bedeutung.

Terme wie $x\, s\, \lambda y t$ sind zu lesen als $(x\, s)\, (\lambda y t)$. Die Notation $\lambda x r\, s$ ist zweideutig und wird nicht verwendet. (Könnte als $(\lambda x r)\, s$ oder als $\lambda x (r\, s)$ gelesen werden.)

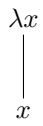
Ein Punkt öffnet eine Klammer, die sich so weit rechts schließt wie syntaktisch sinnvoll, d.h.

$$\begin{array}{lcll} \lambda x.\, y\, z & = & \lambda x(y\, z) & \\ \lambda x.\, \lambda y.\, x & = & \lambda x(\lambda y(x)) & = \quad \lambda x(\lambda y\, x) \\ (\lambda x.y\, z)\, v & = & (\lambda x(y\, z))\, v & \end{array}$$

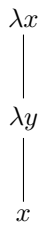
Nach Definition wäre auch $x\,.\,y\,z = x\,(y\,z)$, wir werden die Punkt-Notation nur nach einer Abstraktion verwenden.

Beispiel 1.2 *Hier einige Beispiele für λ-Terme:*

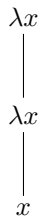
1. $\mathsf{I} = \lambda x x$, *die Identitätsabbildung.*



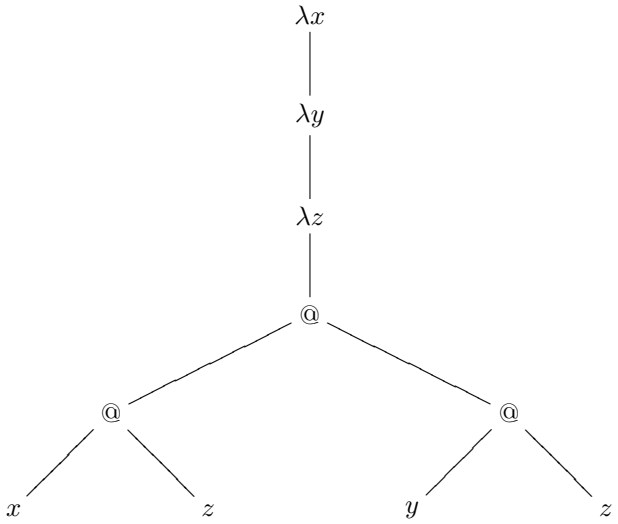
2. $\mathsf{K} = \lambda x \lambda y\, x$, *die erste Projektion.*



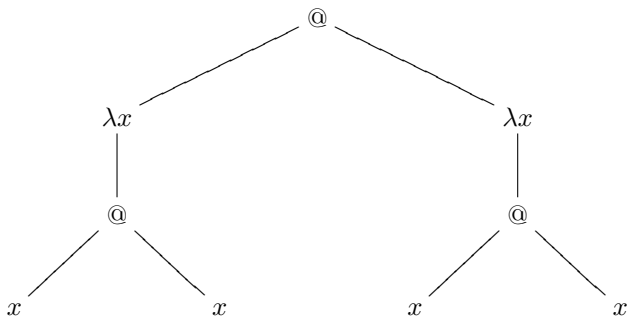
3. $\mathsf{K}^* = \lambda x \mathsf{I} = \lambda x \lambda x\, x$, *die zweite Projektion.*



4. $\mathsf{S} = \lambda x \lambda y \lambda z.\, x\, z\, (y\, z)$.



5. $\Omega = (\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$.



Vektor-Notation. Anwendung einer Funktion r auf mehrere Argumente $s_1, s_2, \ldots s_n$ ($n \geq 0$) kürzen wir mit der Vektornotation ab:

$$r\, \vec{s} := r\, s_1\, s_2\, \ldots s_n = (\ldots (r\, s_1)\, s_2 \ldots)\, s_n.$$

Mehrfach-Abstraktion einer Liste von Variablen $\vec{x} = x_1, x_2, \ldots, x_n$ ($n \geq 0$) kann wie folgt geschrieben werden:

$$\lambda \vec{x} t := \lambda \vec{x}.t := \lambda x_1 x_2 \ldots x_n.t := \lambda x_1 (\lambda x_2 \ldots (\lambda x_n t))$$

Definition 1.3 (Höhe) *Die Höhe* $\mathsf{h}(t) \in \mathbb{N}$ *eines Terms* t *wird definiert durch (strukturelle) Rekursion über* t.

$$\begin{array}{lcl} \mathsf{h}(x) & = & 0 \\ \mathsf{h}(\lambda x t) & = & 1 + \mathsf{h}(t) \\ \mathsf{h}(r\, s) & = & 1 + \max\{\mathsf{h}(r), \mathsf{h}(s)\} \end{array}$$

Die Größe $|t| \in \mathbb{N}$ eines Terms t ist die Anzahl seiner Knoten und wird ebenfalls definiert durch Rekursion über t.

$$\begin{array}{lcl} |x| & = & 1 \\ |\lambda x t| & = & 1 + |t| \\ |r\, s| & = & 1 + |r| + |s| \end{array}$$

Definition 1.4 (Freie und gebundene Variablen) *Wir definieren die (endliche) Menge $\mathsf{FV}(t) \subset \mathsf{V}$ der* freien Variablen *eines Terms t durch Rekursion über t wie folgt.*

$$\begin{array}{lcl} \mathsf{FV}(x) & = & \{x\} \\ \mathsf{FV}(\lambda x t) & = & \mathsf{FV}(t) \setminus \{x\} \\ \mathsf{FV}(r\, s) & = & \mathsf{FV}(r) \cup \mathsf{FV}(s) \end{array}$$

Eine Variable x, die in einem Term t in einer Abstraktion $\lambda x r$ vorkommt, bezeichnen wir als gebunden.

Abstraktion $\lambda x t$ bindet also x in t. In dem Term $x\, \lambda x x$ kommt x sowohl frei als auch gebunden vor. In dem Term $\lambda x.\, y\, \lambda x.x$ wird x durch das innere λ gebunden, nicht durch das äußere.

Definition 1.5 (Geschlossener Term) *Ein Term t heißt geschlossen, falls $\mathsf{FV}(t) = \emptyset$. Die Menge der geschlossenen Terme bezeichnen wir mit Λ^0.*

Bemerkung 1.6 (Zur Substitution (Barendregt, 1984, 2.1.10)) *Bei der Substitution müssen wir Vorsicht walten lassen, damit keine freien Variablen gefangen werden; sonst ergeben sich Inkonsistenzen. Z.B., modulo $=_\beta$, $(\lambda x y.\, y\, x)\, r\, s = ((\lambda y.\, y\, x)[r/x])\, s = (\lambda y.\, y\, r)\, s = (y\, r)[s/y] = s\, r$, insbesondere $(\lambda x y.\, y\, x)\, y\, x = x\, y$. Naive Anwendung der Substitution lässt uns aber auch $(\lambda x y.\, y\, x)\, y\, x = ((\lambda y.\, y\, x)[y/x])\, x = (\lambda y.\, y\, y)\, x = (y\, y)[x/y] = x\, x$ herleiten. Damit wäre $x\, y = x\, x$ und die Theorie inkonsistent.*

Definition 1.7 (Substitution) *Gegeben Terme s, t und eine Variable x definieren wir die* Substitution von s für x in t, $t[s/x]$, *(sprich: "t mit s für x") rekursiv wie folgt:*

$$\begin{array}{rcll} x[s/x] & = & s & \\ y[s/x] & = & y & \text{falls } x \neq y \\ (t_1\, t_2)[s/x] & = & (t_1[s/x])\, (t_2[s/x]) & \\ (\lambda y r)[s/x] & = & \lambda y'.\, r[y'/y][s/x] & \text{falls } x \neq y \text{ und } y \in \mathsf{FV}(s) \\ (\lambda y r)[s/x] & = & \lambda y.\, r[s/x] & \text{falls } x \neq y \text{ und } y \notin \mathsf{FV}(s) \\ (\lambda x r)[s/x] & = & \lambda x r & \end{array}$$

Die Bedeutung eines Termes hängt nur von der Bedeutung seiner freien Variablen ab. Terme, die sich nur in den Namen von gebundenen Variablen unterscheiden wollen wir als identisch betrachten. Falls y nicht in t vorkommt, bezeichnen wir mit $t\{y/x\}$ die Ersetzung *aller Vorkommen* (gebunden, frei, in Abstraktionen) von x in t durch y.

Zwei Terme t und t' bezeichnen wir als α-äquivalent, falls sie sich nur in den Namen gebundener Variablen unterscheiden, z.B. $\lambda xx =_\alpha \lambda yy$, $\lambda x \lambda xx =_\alpha \lambda y \lambda xx$, aber $\lambda x.\, y\, x \neq_\alpha \lambda x.\, x\, x$. Formal definieren wir die α-Äquivalenz als kleinste kompatible Äquivalenzrelation über dem Axiom $\lambda xt =_\alpha \lambda x'.t\{x'/x\}$ (wobei x' nicht in t vorkomme).

Definition 1.8 (α-Äquivalenz) $=_\alpha$ *ist die kleinste Relation auf* Λ*, die unter den folgenden Regeln abgeschlossen ist.*

$$\text{EQ}_\alpha\text{-AX}\ \frac{}{\lambda xt =_\alpha \lambda y.\, t\{y/x\}}\ y \textit{ kommt in } t \textit{ nicht vor}$$

$$\text{EQ}_\alpha\text{-REFL}\ \frac{}{t =_\alpha t} \qquad \text{EQ}_\alpha\text{-SYM}\ \frac{t' =_\alpha t}{t =_\alpha t'} \qquad \text{EQ}_\alpha\text{-TRANS}\ \frac{t_1 =_\alpha t_2 \qquad t_2 =_\alpha t_3}{t_1 =_\alpha t_3}$$

$$\text{EQ}_\alpha\text{-APP-L}\ \frac{r =_\alpha r'}{r\, s =_\alpha r'\, s} \qquad \text{EQ}_\alpha\text{-APP-R}\ \frac{s =_\alpha s'}{r\, s =_\alpha r\, s'}$$

$$\text{EQ}_\alpha\text{-}\xi\ \frac{t =_\alpha t'}{\lambda xt =_\alpha \lambda xt'}$$

Wir sagen auch $=_\alpha$ ist durch diese Regeln *induktiv definiert.*

Bemerkung 1.9 *Da die Relation* $=_\alpha$ *transitiv ist, können die Regeln* EQ_α-APP-L *und* EQ_α-APP-R *durch die Regel*

$$\text{EQ}_\alpha\text{-APP}\ \frac{r =_\alpha r' \qquad s =_\alpha s'}{r\, s =_\alpha r'\, s'}$$

ersetzt werden.

Konvention 1.10 (Barendregt (Barendregt, 1984, 2.1.12+13))

1. *α-kongruente Terme werden als identisch betrachtet, also z.B. $\lambda xx = \lambda yy$.*

2. *Wenn eine Anzahl Terme $t_1, \ldots, t_n$ in einem mathematischen Kontext (z.B. Definition, Satz) auftreten, so seien alle ihre gebundenen Variablen von allen ihren freien Variablen verschieden.*

Durch Umbenennung gebundener Variablen können wir diese *Variablenkonvention* immer sicher stellen.

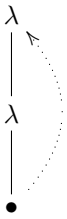
Definition 1.11 (Subterm) *Ein Term s heisst* Subterm *von t, falls s ein Unterbaum von t ist, modulo α.*

1.2 Lokal Namenlose Repräsentation

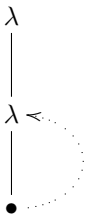
Anstatt Terme mit benamsten Variablen modulo α zu betrachten, kann man auch Terme *namenlos* repräsentieren, so dass α-äquivalente Terme schon identisch sind. Gebundene (= lokale) Variablen werden statt durch Namen durch Rückverweise auf ihren Bindungsort repräsentiert. Freie Variablen werden weiterhin durch einen Namen verkörpert. Man spricht daher von einer *lokal namenlosen Repräsentation*.

Beispiel 1.12 *1.* $\mathsf{I} = \lambda x x.$

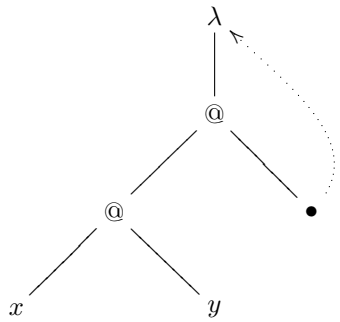
2. $\mathsf{K} = \lambda x \lambda y\, x.$



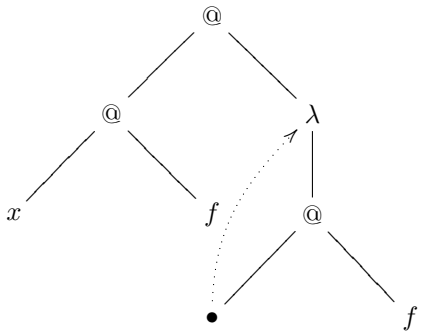
3. $\mathsf{K}^* = \lambda x \mathsf{I} = \lambda x \lambda x\, x.$



4. $\lambda z.\, x\, y\, z$



5. $x\, f\, (\lambda x.\, x\, f)$



Beobachtung: Rückverweise zielen stets auf einen (indirekten) Vaterknoten. Deswegen kann man sie durch die Anzahl der auf dem Weg zum Bindungsort übersprungenen λ-Knoten repräsentieren. Diese Zahl bezeichnet man als *de Bruijn-Index*.

Beispiel 1.13

1. $\mathsf{I} = \lambda 0$
2. $\mathsf{K} = \lambda\lambda 1.$
3. $\mathsf{K}^* = \lambda\lambda 0.$
4. $\lambda z.\, x\, y\, z = \lambda.\, x\, y\, 0.$
5. $x\, f\, \lambda x.\, x\, f = x\, f\, \lambda.\, 0\, f.$
6. $\lambda x.\, x\, \lambda y.\, y\, x = \lambda.\, 0\, \lambda.\, 0\, 1.$ *Man beachte, dass diesselbe gebundene Variable x einmal als de Bruijn-Index 0 und einmal als 1 erscheint. Das macht die Notation mittels de Bruijn-Indizes für das menschliche Auge schwer lesbar.*

Ist ein de Bruijn-Index zu groß, dann zeigt er auf keinen Bindungsort. Der Term ist dann nicht *wohlgeformt*, z.B. $\lambda 1$. Ein nicht-wohlgeformter Term hat *freie Indices*.

Definition 1.14 (Freie Indices) *Die Menge $\mathsf{FI}(t)$ der freien Indices eines lokal namenlosen Terms t ist rekursiv gegeben durch:*

$$\begin{array}{lcl} \mathsf{FI}(x) & = & \emptyset \\ \mathsf{FI}(i) & = & \{i\} \\ \mathsf{FI}(r\, s) & = & \mathsf{FI}(r) \cup \mathsf{FI}(s) \\ \mathsf{FI}(\lambda t) & = & \{i - 1 \mid i \in \mathsf{FI}(t), i > 0\} \end{array}$$

Ist $\mathsf{FI}(t) = \emptyset$, ist t wohlgeformt.

Definition 1.15 (Substitution) *Gegeben lokal namenlose Terme s, t mit $\mathsf{FI}(s) = \emptyset$ und einen de Bruijn-Index i definieren wir die* kollabierende Substitution von s für i in t, $t[s/i]$, *rekursiv wie folgt:*

$$\begin{array}{rcl} x[s/i] & = & x \\ j[s/i] & = & \begin{cases} s & \text{falls } i = j \\ j & \text{falls } i > j \\ j-1 & \text{falls } i < j \end{cases} \\ (t_1\, t_2)[s/i] & = & (t_1[s/i])\,(t_2[s/i]) \\ (\lambda r)[s/i] & = & \lambda.\, r[s/i+1] \end{array}$$

Da wir für den Index i substituieren, verschwindet er und alle grösseren Indizes "fallen herunter", d.h. werden um eins erniedrigt. (Denken Sie an einen Stapel Bretter, aus dem Sie das $i$te von unten herausziehen.)

Die typische Anwendung der kollabierenden Substitution ist die Auflösung von β-Redexen: $(\lambda t)\, s \longrightarrow_\beta t[s/0]$. Dabei bleiben wir im Bereich der wohlgeformten Terme: $\mathsf{FI}((\lambda t)\, s) = \emptyset$ impliziert $\mathsf{FI}(s) = \emptyset$ und $\mathsf{FI}(t) \subseteq \{0\}$, daher $\mathsf{FI}(t[s/0]) = \emptyset$.

Im Folgenden verwenden wir informell wieder benamste λ-Terme modulo α-Äquivalenz und Variablenkonvention.

1.3 Substitution und β-Gleichheit

Mit der Variablenkonvention können wir nun Substitution so definieren:

Definition 1.16 (Substitution) *Gegeben Terme s, t und eine Variable x definieren wir die* Substitution von s für x in t, $t[s/x]$, *(sprich: "t mit s für x") rekursiv wie folgt:*

$$\begin{array}{rcll} x[s/x] & = & s & \\ y[s/x] & = & y & \text{falls } x \neq y \\ (t_1\, t_2)[s/x] & = & (t_1[s/x])\,(t_2[s/x]) & \\ (\lambda y r)[s/x] & = & \lambda y.\, r[s/x] & \text{o.B.d.A. } x \neq y \text{ und } y \notin \mathsf{FV}(s) \end{array}$$

In der letzten Gleichung verwenden wir die Variablenkonvention, d.h. die freien Variablen von s und x sind von den gebundenen Variabeln von $\lambda y r$ verschieden, damit ist insbesondere $x \neq y$ und $y \notin \mathsf{FV}(s)$.

Bemerkung 1.17 *In der Literatur finden viele unterschiedliche Schreibweisen der Substitution, z.B., $t\{s/x\}$, $[s/x]t$, $t[x := s]$, $t[x \backslash s]$, $t[x \to s]$, $t[x \leftarrow s]$, ...*

Lemma 1.18 (Eigenschaften von Substitution)

1.	$t[x/x]$	$=$	t	
2.	$t[s/x]$	$=$	t	falls $x \notin \mathsf{FV}(t)$
3.	$t[y/x][s/y]$	$=$	$t[s/x]$	falls $y \notin \mathsf{FV}(t)$
4.	$t[s/y][r/x]$	$=$	$t[r/x][s[r/x]/y]$	falls $y \notin \mathsf{FV}(r)$
5.	$t[s/y][r/x]$	$=$	$t[r/x][s/y]$	falls $y \notin \mathsf{FV}(r)$ und $x \notin \mathsf{FV}(s)$

Beweis. Jeweils durch Induktion über t. Aussage 4 benötigt Aussage 2 im Fall $t = x$. □

Definition 1.19 (β-Gleichheit)

$$\text{EQ}_\beta\text{-AX}\ \frac{}{(\lambda x t)\, s =_\beta t[s/x]}$$

$$\text{EQ}_\beta\text{-REFL}\ \frac{}{t =_\beta t} \qquad \text{EQ}_\beta\text{-SYM}\ \frac{t' =_\beta t}{t =_\beta t'} \qquad \text{EQ}_\beta\text{-TRANS}\ \frac{t_1 =_\beta t_2 \qquad t_2 =_\beta t_3}{t_1 =_\beta t_3}$$

$$\text{EQ}_\beta\text{-APP-L}\ \frac{r =_\beta r'}{r\, s =_\beta r'\, s} \qquad \text{EQ}_\beta\text{-APP-R}\ \frac{s =_\beta s'}{r\, s =_\beta r\, s'}$$

$$\text{EQ}_\beta\text{-}\xi\ \frac{t =_\beta t'}{\lambda x t =_\beta \lambda x t'}$$

Bemerkung 1.20 *Wieder können wir* EQ$_\beta$-APP-L *und* EQ$_\beta$-APP-R *durch folgende Regel ersetzen:*

$$\text{EQ}_\beta\text{-APP}\ \frac{r =_\beta r' \qquad s =_\beta s'}{r\, s =_\beta r'\, s'}$$

Beispiel 1.21 $(\lambda x.\, x\, x\, x)\, \lambda y y =_\beta \lambda y y$.

PROBLEM: Wie zeigen wir $\lambda x \lambda y x \neq_\beta \lambda x \lambda y y$?

1.4 Relationen und Hüllen

Die α- und β-Gleichheit sind Beispiele von kompatiblen Äquivalenzrelationen auf λ-Termen.

Definition 1.22 (Kompatible Relation) *Eine Relation $R \subseteq \Lambda \times \Lambda$ heißt* kompatibel (mit allen Termkonstruktoren)*, falls*

1. *$r\ R\ r'$ impliziert $r\, s\ R\ r'\, s$ für alle $r, r', s \in \Lambda$,*
2. *$s\ R\ s'$ impliziert $r\, s\ R\ r\, s'$ für alle $r, s, s' \in \Lambda$, und*
3. *$t\ R\ t'$ impliziert $\lambda x t\ R\ \lambda x t'$ für alle $t, t' \in \Lambda$ und alle $x \in \mathsf{V}$.*

Statt kompatibel sagt man auch *abgeschlossen unter allen Termkonstruktoren*, oder *kontektuell abgeschlossen.*

Wir definieren nun verschiedene Abschlüsse von Relationen konstruktiv.

Definition 1.23 (Kompatible Hülle) *Sei $R \subseteq \Lambda \times \Lambda$. Die kompatible Hülle R^{c} ist induktiv definiert wie folgt:*

$$\text{IN}\ \frac{t\ R\ t'}{t\ R^{\mathsf{c}}\ t'} \qquad \text{APP-L}\ \frac{r\ R^{\mathsf{c}}\ r'}{r\ s\ R^{\mathsf{c}}\ r'\ s} \qquad \text{APP-R}\ \frac{s\ R^{\mathsf{c}}\ s'}{r\ s\ R^{\mathsf{c}}\ r\ s'} \qquad \xi\ \frac{t\ R^{\mathsf{c}}\ t'}{\lambda x t\ R^{\mathsf{c}}\ \lambda x t'}$$

Eine Relation R' heisst kompatibel (mit allen Termkonstruktoren)*, falls sie unter* APP-L*,* APP-R*, und ξ abgeschlossen ist.*

Definition 1.24 (Transitive Hülle) *Sei M eine Menge und $R \subseteq M \times M$ eine Relation auf M. Wir definieren die Relation R^+ wie folgt induktiv:*

$$\text{START}^+\ \frac{t\ R\ t'}{t\ R^+\ t'} \qquad \text{STEP}^+\ \frac{t_1\ R\ t_2 \qquad t_2\ R^+\ t_3}{t_1\ R^+\ t_3}$$

Lemma 1.25 *R^+ ist die transitive Hülle von R.*

Beweis. Wir zeigen zuerst, dass R^+ transitiv ist. Gegeben $t_1\ R^+\ t_2$ und $t_2\ R^+\ t_3$ zeigen wir $t_1\ R^+\ t_3$ durch Induktion über die Herleitung von $t_1\ R^+\ t_2$.

Fall

$$\text{START}^+\ \frac{t_1\ R\ t_2}{t_1\ R^+\ t_2}$$

Hier gilt sofort $t_1\ R^+\ t_3$ nach Regel STEP^+.

Fall

$$\text{STEP}^+\ \frac{t_1\ R\ t \qquad t\ R^+\ t_2}{t_1\ R^+\ t_2}$$

Nach Induktionsvoraussetzung gilt $t\ R^+\ t_3$. Mit Regel STEP^+ erhalten wir daraus $t_1\ R^+\ t_3$.

Nun zeigen wir noch, dass R^+ die kleinste transitive Relation oberhalb von R ist. Vermöge START^+ gilt sofort $R \subseteq R^+$.

Sei nun R' eine transitive Relation oberhalb von R. Wir zeigen $R^+ \subseteq R'$. Angenommen $t_1\ R^+\ t_2$, zeigen wir $t_1\ R'\ t_2$ durch Induktion über R^+.

Fall

$$\text{START}^+\ \frac{t_1\ R\ t_2}{t_1\ R^+\ t_2}$$

Da R' oberhalb von R ist, gilt $t_1\ R'\ t_2$.

Fall

$$\text{STEP}^+\ \frac{t_1\ R\ t \qquad t\ R^+\ t_2}{t_1\ R^+\ t_2}$$

Wie eben ist $t_1\ R'\ t$, und nach Induktionsvoraussetzung gilt $t\ R'\ t_2$. Da R' transitiv ist, folgt $t_1\ R'\ t_2$. □

Bemerkung 1.26 *Es gibt einige andere, jedoch gleichwertige Arten, die transitive Hülle zu definieren.*

Definition 1.27 (Reflexive und symmetrische Hülle) *Die* reflexive Hülle $R^=$ *einer Relation* $R \subseteq M \times M$ *ist die Relation* $R \cup \{(t,t) \mid t \in M\}$. *Die symmetrische Hülle ist die Relation* $R \cup \{(t',t) \mid t\ R\ t'\}$.

Lemma 1.28 (Reflexiv-transitive Hülle) *Sei* M *eine Menge und* $R \subseteq M \times M$ *eine Relation auf* M. *Die* reflexiv-transitive Hülle R^* *von* R *kann man auch wie folgt induktiv definieren:*

$$\textsc{start}^* \ \frac{}{t\ R^*\ t} \qquad \textsc{step}^* \ \frac{t_1\ R\ t_2 \qquad t_2\ R^*\ t_3}{t_1\ R^*\ t_3}$$

Lemma 1.29 (Vererbung von Abschlusseigenschaften)

1. *Die transitive Hülle einer symmetrischen Relation ist wieder symmetrisch.*
2. *Die reflexive/symmetrische/transitive Hülle einer kompatiblen Relation ist wieder kompatibel.*
3. *Die reflexive Hülle einer symmetrischen/transitiven/kompatiblen Relation ist symmetrisch/transitiv/kompatibel.*

Beweis. 1. Sei R symmetrisch und $t\ R^+\ t'$. Wir zeigen $t'\ R^+\ t$ durch Induktion über die Herleitung von $t\ R^+\ t'$.

Fall

$$\textsc{start}^+ \ \frac{t\ R\ t'}{t\ R^+\ t'}$$

Da R symmetrisch ist, gilt $t'\ R\ t$ und mit Regel $\textsc{start}^+$ auch $t'\ R^+\ t$.

Fall

$$\textsc{step}^+ \ \frac{t_1\ R\ t_2 \qquad t_2\ R^+\ t_3}{t_1\ R^+\ t_3}$$

Es gilt $t_2\ R^+\ t_1$ nach Symmetrie von R und $\textsc{start}^+$, ausserdem nach Induktionshypothese, $t_3\ R^+\ t_2$. Da R^+ transitiv ist, gilt somit auch $t_3\ R^+\ t_1$.

□

Bemerkung 1.30 *Die symmetrische Hülle einer transitiven Relation ist im Allgemeinen nicht transitiv, ein Beispiel ist die transitive Relation* $<$ *auf* $\mathbb{N}$, *deren symmetrische Hülle die Ungleichheit* $\neq$ *ist.*

Bemerkung 1.31 *Achtung! Die kompatible Hülle einer transitiven Relation ist in der Regel* nicht *transitiv. Beispiel: Sei* R *das durch die Regeln* $\textsc{eq}_\beta$-AX *und* $\textsc{eq}_\beta$-TRANS *aufgespannte Fragment der* β*-Gleichheit. In* R^c *gelten* $((\lambda x x)\,\lambda y y)\,\lambda z z\ R^\mathsf{c}\ (\lambda y y)\,\lambda z z\ R^\mathsf{c}\ \lambda z z$, *aber nicht* $((\lambda x x)\,\lambda y y)\,\lambda z z\ R^\mathsf{c}\ \lambda z z$.

Bemerkung 1.32 *Es empfiehlt sich also, Hüllen in der folgenden Reihenfolge zu bilden:*

1. *kompatibel*
2. *symmetrisch*
3. *transitiv*
4. *reflexiv.*

1.5 Reduktion

In der Mathematik berechnen wir arithmetische Ausdrücke durch eine Richtung der Gleichheit:

$$\begin{array}{rcl}(5+3)*4 & = & 8*4\\ & = & 32\end{array}$$

Berechnung ist paradoxerweise ein Informations*verlust*. So kann ich aus $8*4$ deterministisch 32 gewinnen, jedoch nicht umgekehrt. Die Strukturierung der 32 als $8*4$ ist durch die Berechnung verlorengegangen.

In derselben Weise richten wir dir β-Gleichheit, um λ-Terme zu berechnen:

Definition 1.33 (β-Kontraktion) *Die* β-Kontraktionsrelation β *ist gegeben durch das Axiomenschema* $(\lambda x t)\, s\ \beta\ t[s/x]$. *Die linke Seite* $((\lambda x t)\, s)$ *bezeichnen wir als* Redex *(*red*ucible* ex*pression), und die rechte* $(t[s/x])$ *als sein* Redukt.

Lemma 1.34 (Von der Kontraktion zur Gleichheit) $=_\beta$ *ist die reflexiv-transitiv-symmetrische Hülle der kompatiblen Hülle der* β*-Kontraktionsrelation* $\beta \subseteq \Lambda \times \Lambda$.

Definition 1.35 (β-Reduktion) *Die* Ein-Schritt-β-Reduktion $\longrightarrow_\beta$ *ist die kompatible Hülle der* β-Kontraktion $t\ \beta\ t'$.

Der Wert eines λ-Terms wird berechnet, indem wir den Term reduzieren so lange wir können. Entweder, wir reduzieren endlos, oder wir erreichen eine Normalform.

Beispiel 1.36

$$\begin{array}{rcl}\mathsf{S}\,\mathsf{K}\,\mathsf{K} & \equiv & (\lambda x \lambda y \lambda z.\, x\, z\, (y\, z))\,\mathsf{K}\,\mathsf{K}\\ & \longrightarrow & (\lambda y \lambda z.\, \mathsf{K}\, z\, (y\, z))\,\mathsf{K}\\ & \longrightarrow & (\lambda y \lambda z.\, (\lambda y.\, z)\, (y\, z))\,\mathsf{K}\\ & \longrightarrow & (\lambda y \lambda z.\, z)\,\mathsf{K}\\ & \longrightarrow & \lambda z z\end{array}$$

$$\begin{array}{rcl}\Omega & \equiv & (\lambda x.\, x\, x)\, (\lambda x.\, x\, x)\\ & \longrightarrow & (\lambda x.\, x\, x)\, (\lambda x.\, x\, x)\\ & \longrightarrow & \ldots\end{array}$$

Definition 1.37 (Normalform) *Der Term t heißt* Normalform *bezüglich einer Reduktionsrelation $\longrightarrow$, falls er sich nicht weiter (echt) reduzieren lässt, d.h., falls $t \longrightarrow t'$, dann $t \equiv t'$. Wir sagen auch t* ist normal. *Der Term t* hat Normalform t', *falls $t \longrightarrow^* t'$ und t' normal.*

Falls $\longrightarrow$ nicht reflexiv ist, dann ist t normal gdw. es kein t' gibt mit $t \longrightarrow t'$. Hierfür schreiben wir kürzer $t \not\longrightarrow$.

Bemerkung 1.38 *$t \not\longrightarrow_\beta$ gdw. t keinen β-Redex enthält. Dies folgt aus den Kongruenz-Regeln (*APP *und ξ) und kann formal durch Induktion nach t bewiesen werden.*

1.6 Literaturverweise

Die lokal namenlose Repräsentation von λ-Termen geht zurück auf Andy Gordon (1994). Eine komplett namenlose Repräsentation, in der freie Indizes die freien Variablen repräsentieren, hat de Bruijn (1972) im System AUTOMATH verwendet. Diese Methode wird ausführlich in Pierce's Buch (2002) erläutert. Weitere Repräsentationstechniken sind *abstrakte Syntax höherer Ordnung* (Higher Order Abstract Syntax – HOAS) (Harper et al., 1993) und die *nominale* Technik (Gabbay & Pitts, 1999).

1.7 Übungen

Übung 1.39 *Schreiben Sie rekursives Prädikat* wf*, das testet, ob ein lokal namenloser Term wohlgeformt ist, ohne erst die Menge der freien Indizes zu berechnen.*

Übung 1.40 (Syntaxgerichtete Definition der α-Äquivalenz) *Sei $=^\alpha$ die kleinste Relation, die unter den folgenden Regeln abgeschlossen ist.*

$$\text{EQ}_\alpha\text{-VAR}\ \frac{}{x =^\alpha x} \qquad \text{EQ}_\alpha\text{-APP}\ \frac{r =^\alpha r' \qquad s =^\alpha s'}{r\,s =^\alpha r'\,s'}$$

$$\text{EQ}_\alpha\text{-ABS}\ \frac{t\{y/x\} =^\alpha t'\{y/x'\}}{\lambda x t =^\alpha \lambda x' t'}\ y \text{ kommt nicht in } t, t' \text{ vor}$$

Beweisen Sie, dass $=^\alpha$ diesselben Terme identifiziert wie $=_\alpha$. Anleitung:

1. *Zeigen Sie die Korrektheit von $=^\alpha$, also die Aussage $t =^\alpha t' \implies t =_\alpha t'$, durch (Regel-)Induktion über die Herleitung von $t =^\alpha t'$.*
2. *Zeigen Sie, dass $=^\alpha$ eine Äquivalenzrelation (reflexiv, transitiv, symmetrisch) ist.*
3. *Zeigen Sie schliesslich die Vollständigkeit, also die Aussage $t =_\alpha t' \implies t =^\alpha t'$, durch Induktion über die Herleitung von $t =_\alpha t'$.*

Kapitel 2

Der λ-Kalkül als Berechnungsmodell

Der Lambda-Kalkül ist eine Turing-vollständige Programmiersprache, d.h. jede berechenbare Funktion lässt sich im Lambda-Kalkül repräsentieren. Weitere Beispiele für Turing-mächtige Sprachen sind Turing-Maschinen, WHILE- und GOTO-Sprachen, kombinatorische Algebren, jede vernünftige Programmiersprache, aber auch z.B. das Textsatzsystem TEX und die Seitenbeschreibungssprache POSTSCRIPT.

Im diesem Kapitel legen wir β-Gleichheit zugrunde, d.h., wir verwenden $=$ für $=_\beta$ und $\longrightarrow$ für $\longrightarrow_\beta$. Syntaktische Gleichheit bezeichnen wir mit $\equiv$.

2.1 Programmieren im λ-Kalkül

Im Folgenden werden wir einige elementare Datentypen und die wichtigsten Anwendungen dieser Datentypen im Lambda-Kalkül definieren.

2.1.1 Church-Booleans

Die ersten Datenobjekte, den wir betrachten, sind die Wahrheitswerte T und F und ihre Verwendung mit if. Da im Lambda-Kalkül "alles Funktion" ist, müssen wir die drei Konstrukte durch Funktionen realisieren. Wollen wir Datentypen im Lambda-Kalkül repräsentieren, müssen wir folgende Schlüssel-Frage beantworten:

Wie kann ein Objekt des betrachteten Typs verwendet werden?

Es geht also nicht um die *Konstruktion (Einführung)*, sondern um die *Verwendung (Elimination)* eines Datenobjekts. Ein Datenobjekt im Lambda-Kalkül wird bestimmt durch seine Verwendungsmöglichkeiten.

Ein Wahrheitswert steht für eine *Entscheidung*, welche von zwei alternativen Berechnungen fortgesetzt werden soll. Das aus Programmiersprachen bekannte

`if-then-else` ist also die allgemeinste Verwertung eines boolschen Wertes. Alle anderen Funktionen, wie z.B. logische Verknüpfungen, können mit Hilfe von if ausgedrückt werden.

Der Wert T ist eine Funktion, die angewandt auf zwei Berechnungen, den *then*-Zweig und den *else*-Zweig, stets den *then*-Zweig wählt. Der Wert F verfolgt immer den *else*-Zweig.

$$\begin{array}{lcl} \mathsf{T} & \equiv & \lambda t \lambda e.\, t \\ \mathsf{F} & \equiv & \lambda t \lambda e.\, e \\ \mathsf{if} & \equiv & \lambda b \lambda t \lambda e.\, b\, t\, e \end{array}$$

Der Destruktor if tut nichts Besonderes, er wendet den übergebenen Wahrheitswert b auf t, denn *then*-Zweig, und e, den *else*-Zweig an. Die Negationsfunktion können wir nun wie folgt definieren:

$$\mathsf{not} \equiv \lambda b.\ \mathsf{if}\ b\ \mathsf{F}\ \mathsf{T}$$

Die Definition is korrekt, wie aus den folgenden Auswertungssequenzen ersichtlich:

$$\begin{array}{l} \mathsf{not}\ \mathsf{T} \equiv (\lambda b.\ \mathsf{if}\ b\ \mathsf{F}\ \mathsf{T})\ \mathsf{T} \\ \longrightarrow \mathsf{if}\ \mathsf{T}\ \mathsf{F}\ \mathsf{T} \equiv (\lambda b \lambda t \lambda e.\ b\ t\ e)\ \mathsf{T}\ \mathsf{F}\ \mathsf{T} \\ \longrightarrow (\lambda t \lambda e.\ \mathsf{T}\ t\ e)\ \mathsf{F}\ \mathsf{T} \\ \longrightarrow (\lambda e.\ \mathsf{T}\ \mathsf{F}\ e)\ \mathsf{T} \\ \longrightarrow \mathsf{T}\ \mathsf{F}\ \mathsf{T} \equiv (\lambda t \lambda e.\ t)\ \mathsf{F}\ \mathsf{T} \\ \longrightarrow (\lambda e.\ \mathsf{F})\ \mathsf{T} \longrightarrow \mathsf{F} \end{array}$$

$$\begin{array}{l} \mathsf{not}\ \mathsf{F} \equiv (\lambda b.\ \mathsf{if}\ b\ \mathsf{F}\ \mathsf{T})\ \mathsf{F} \\ \longrightarrow^4 (\mathsf{F}\ \mathsf{F}\ \mathsf{T}) \equiv (\lambda t \lambda e.\ e)\ \mathsf{F}\ \mathsf{T} \\ \longrightarrow^2 \mathsf{T} \end{array}$$

Die Funktion not kann noch knapper definiert werden durch den Term $\lambda b.\ b\ \mathsf{F}\ \mathsf{T}$, welcher sich auch ergibt, wenn man die ursprüngliche Definition normalisiert. Auch folgende Definition ist denkbar:

$$\mathsf{not} \equiv \lambda b \lambda t \lambda e.\ b\ e\ t$$

Boolsche Funktionen lassen sich durch (geschachtelte) if-Ausdrücke implementieren. Deswegen sind sie auch im Lambda-Kalkül problemlos kodierbar. Z.B. Konjunktion:

$$\mathsf{and} \equiv \lambda b \lambda c.\ b\ c\ \mathsf{F}$$

Der Ausdruck $\mathsf{and}\ b\ c$ liesst sich: Wenn b, dann c, sonst falsch.

2.1.2 Paare

Ein *Paar* ist der Zusammenschluss zweier Komponenten. Bei der Elimination eines Paares erhälten wir eine der Komponenten zurück. (Wollen wir beide Komponenten, müssen wir das Paar zweimal verwenden.) Welche der beiden Komponenten wir erhalten wollen, geben wir durch einen Wahrheitwert an. Ein

Paar ist also eine Funktion, die ein Bit erwartet und eine Komponente zurückliefert.

$$\begin{array}{lcl} \mathsf{Pair} & \equiv & \lambda c_\mathsf{T} \lambda c_\mathsf{F} \lambda b.\ b\ c_\mathsf{T}\ c_\mathsf{F} \\ \mathsf{fst} & \equiv & \lambda p.\ p\ \mathsf{T} \\ \mathsf{snd} & \equiv & \lambda p.\ p\ \mathsf{F} \end{array}$$

Die Selektoren fst und snd tun nichts weiter, als entweder die Nachricht T oder F an das Paar p zu übergeben. Wieder sind also alle Verwendungsmöglichkeiten im Konstruktor Pair enthalten. Korrektheit wird ersichtlich durch die Reduktionsfolgen:

$$\begin{array}{l} \mathsf{fst}\ (\mathsf{Pair}\ t_1\ t_2) \equiv (\lambda p.\ p\ \mathsf{T})\ (\mathsf{Pair}\ t_1\ t_2) \\ \longrightarrow (\mathsf{Pair}\ t_1\ t_2)\ \mathsf{T} \equiv (\lambda c_\mathsf{T} \lambda c_\mathsf{F} \lambda b.\ b\ c_\mathsf{T}\ c_\mathsf{F})\ t_1\ t_2\ \mathsf{T} \\ \longrightarrow^3 \mathsf{T}\ t_1\ t_2 \longrightarrow^2 t_1 \end{array}$$

$$\begin{array}{l} \mathsf{snd}\ (\mathsf{Pair}\ t_1\ t_2) \equiv (\lambda p.\ p\ \mathsf{F})\ (\mathsf{Pair}\ t_1\ t_2) \\ \longrightarrow^4 \mathsf{F}\ t_1\ t_2 \longrightarrow^2 t_2 \end{array}$$

2.1.3 Church-Ziffern

Auch unendliche Datentypen wie die natürlichen Zahlen lassen sich im Lambda-Kalkül repräsentieren. (Dabei ist jede Zahl ein endliches Gebilde.) Eine natürliche Zahl n gibt allgemein an, *wie oft* eine Prozedur ausgeführt wird bzw. eine Funktion angewendet wird. Die Repräsentation der Zahl n im Lambda-Kalkül, die Church-Ziffer $\underline{n}$ ist also eine Funktion, die eine beliebige Funktion f als Argument nimmt und n-mal iteriert. Dabei ist 0-fache Iteration die Identität. Die n-te Iterierte einer Funktion f können wir allgemein so definieren.

$$\begin{array}{lcl} f^0 & \equiv & \lambda x.\ x \\ f^{n+1} & \equiv & f \circ f^n \equiv \lambda x.\ f\ (f^n\ x) \end{array}$$

Damit erhalten wir die Church-Ziffern nach folgendem Schema:

$$\begin{array}{lcl} \underline{0} & \equiv & \lambda f \lambda x.\ x \\ \underline{1} & \equiv & \lambda f \lambda x.\ f\ x \\ \underline{2} & \equiv & \lambda f \lambda x.\ f\ (f\ x) \\ \vdots & & \\ \underline{n} & \equiv & \lambda f \lambda x.\ f^n\ x \end{array}$$

Die Applikation $\underline{n}\ f$ einer Church-Ziffer ist also f^n. Tabelle 2.1 listet einige einfache Funktionen über natürlichen Zahlen auf.

Die Vorgänger-Funktion pred mit $\mathsf{pred}\ \underline{n} = \underline{n-1}$ ist nicht direkt implementierbar. Man benötigt Paare.

$$\begin{array}{lcl} \mathsf{x} & \equiv & \mathsf{Pair}\ \underline{0}\ \underline{0} \\ \mathsf{f} & \equiv & \lambda p.\ \mathsf{Pair}\ (\mathsf{snd}\ p)\ (\mathsf{succ}\ (\mathsf{snd}\ p)) \\ \mathsf{pred} & \equiv & \lambda n.\ \mathsf{fst}\ (n\ \mathsf{f}\ \mathsf{x}) \end{array}$$

Übung 2.1 *Definieren Sie Multiplikation* mult *auf Church-Ziffern.*

Funktion	Beschreibung	Spezifikation	Implementation
succ	Nachfolger	succ $\underline{n} = \underline{n+1}$	$\lambda n \lambda f \lambda x.\ f\ (n\ f\ x)$
add	Addition	add $\underline{n}\ \underline{m} = \underline{n+m}$	$\lambda n \lambda m \lambda f \lambda x.\ n\ f\ (m\ f\ x)$
iszero	Test auf 0	iszero $\underline{0} = \mathsf{T}$ iszero $\underline{n+1} = \mathsf{F}$	$\lambda n.\ n\ (\lambda\_.\ \mathsf{F})\ \mathsf{T}$

Tabelle 2.1: Funktionen über natürliche Zahlen.

Übung 2.2 *Definieren Sie Exponentiation* exp *auf Church-Ziffern.*

Übung 2.3 *Definieren Sie Subtraktion* sub *und euklidische Division*[1] div *auf Church-Ziffern.*

2.1.4 Divergenz

Nicht alle Terme des ungetypten Lambda-Kalküls lassen sich auf Normalform reduzieren. Einfachstes Beispiel für einen *divergenten* Term ist $\Omega \equiv (\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$, der durch Selbstanwendung entsteht. Ω ist invariant unter Reduktion:

$$\Omega \equiv (\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow [\lambda x.\ x\ x/x](x\ x) \equiv (\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \equiv \Omega$$

2.1.5 Funktionen mit mehreren Argumenten und Currying

Funktionen mit mehreren Argumenten (z.B. Pair, add) haben wir mit Hilfe von mehrfacher Lambda-Abstraktion definiert. Streng genommen ist z.B. Pair eine Funktion, die die erste Komponente als Argument nimmt und eine Funktion zurückgibt, die wieder ein Argument, die zweite Komponente, erwartet, und dann ein Paar erzeugt. Diese Darstellung hat den Vorteil, dass man Pair auch *teilweise* anwenden kann. So kann man eine Funktion Pair0 definieren, die nur ein Argument t erwartet und dann ein Paar mit konstanter erster Komponente $\underline{0}$ und zweiter Komponente t erzeugt.

$$\begin{aligned} \mathsf{Pair0} &\equiv \mathsf{Pair}\ \underline{0} \equiv (\lambda c_\mathsf{T} \lambda c_\mathsf{F} \lambda b.\ b\ c_\mathsf{T}\ c_\mathsf{F})\ \underline{0} \longrightarrow \lambda c_\mathsf{F} \lambda b.\ b\ \underline{0}\ c_\mathsf{F} \\ \mathsf{Pair0}\ t &\equiv (\lambda c_\mathsf{F} \lambda b.\ b\ \underline{0}\ c_\mathsf{F})\ t \longrightarrow \lambda b.\ b\ \underline{0}\ t =_\beta \mathsf{Pair}\ \underline{0}\ t \end{aligned}$$

Im Fall von Paaren mag das nicht besonders sinnvoll sein, aber nehmen wir an, wir hätten eine Exponentiationsfunktion power definiert mit

$$\begin{aligned} \mathsf{power}\ \underline{n}\ \underline{m} &= \underline{m^n} \\ \mathsf{power} &\equiv \lambda n \lambda m \ldots \end{aligned}$$

[1] Das ist herkömmliches Teilen mit Rest, wie aus der Grundschule bekannt.

Dann können wir Quadratur $\mathsf{sqr}\ \underline{n} = \underline{n^2}$ sehr knapp definieren durch teilweise Applikation von power:

$$\mathsf{sqr} \equiv \mathsf{power}\ \underline{2}$$

Eine Funktion, die teilweise Applikation erlaubt, nennt man ge*curry*t (nach Haskell Curry).

Alternativ können wir im Lambda-Kalkül Funktionen mit mehreren Argumenten über *Paare* definieren, z.B. die Additionsfunktion

$$\mathsf{add} \equiv \lambda p.\ \lambda f \lambda x.\ (\mathsf{fst}\ p)\ f\ ((\mathsf{snd}\ p)\ f\ x)$$

$$\begin{array}{lcl} \mathsf{add}\ (\mathsf{Pair}\ n\ m) & \longrightarrow & \lambda f \lambda x.\ \mathsf{fst}\ (\mathsf{Pair}\ n\ m)\ f\ (\mathsf{snd}\ (\mathsf{Pair}\ n\ m)\ f\ x) \\ & \longrightarrow^2 & \lambda f \lambda x.\ n\ f\ (m\ f\ x) \end{array}$$

Im Lambda-Kalkül sieht diese Variante etwas umständlich aus, jedoch ist dies in den meisten Programmiersprachen die gängige Form. Etwas eleganter wird sie mit der gebräuchlichen Notation $(a_1, \ldots, a_n)$ für n-Tupel. Im SML können wir add dann folgendermaßen implementieren.

```
add = fn (n, m) => fn f => fn x => n f (m f x)
```

Das Transformieren einer über Tupel definierte Funktion mit mehreren Argumenten in eine mit iterierter λ-Abstraktion nennt man *currying*, die Gegentransformation *uncurrying*.

Übung 2.4 *Definieren Sie eine λ-Funktion* curry*, die eine Funktion f als Argument nimmt, welche wiederum ein Paar von Argumenten erwartet, und eine Funktion zurückliefert, die zwei einzelne Argumente erwartet, aber sonst das Gleiche leistet wie f.*

Übung 2.5 *Definieren Sie auch eine Funktion* uncurry.

2.2 Fixpunkte

Satz 2.6 (Existenz von Fixpunkten) *Zu jedem Term s gibt es einen Term r, so dass $r = s\,r$.*

Beweis. Wir können so einen Term r sogar *uniform* angeben. Sei

$$\mathsf{Y} \equiv \lambda f.\ (\lambda x.\ f\ (x\,x))\ (\lambda x.\ f\ (x\,x)).$$

Dann ist $\mathsf{Y}\,s$ so ein r. Y ist der *Standard-Fixpunktkombinator.* □

Satz 2.7 (Turing, 1937) *Zu jedem Term s gibt es ein r, so dass $s\,r \longrightarrow^* r$.*

Beweis. $r \equiv \Phi\,s$ wobei Φ der *Turing'sche Fixpunktkombinator* ist:

$$\Phi \equiv (\lambda f \lambda x.\ f\ (x\,x\,f))\ (\lambda f \lambda x.\ f\ (x\,x\,f)).$$

□

Im λ-Kalkül lassen sich also beliebige rekursive Funktionen implementieren.

2.3 Berechenbarkeit

Im Folgenden rekapitulieren wir den Begriff der Berechenbarkeit einer Funktion über natürlichen Zahlen und beweisen, dass jede berechenbare Funktion im λ-Kalkül ausdrückbar ist, d.h., *λ-definierbar* ist.

An dieser Stelle betrachten wir nur totale Funktionen.

Definition 2.8 (λ-definierbar) *Eine* (p-stellige) numerische Funktion *($p \in \mathbb{N}$) ist eine (totale) Abbildung $f \in \mathbb{N}^p \to \mathbb{N}$. Sie heisst* λ-definierbar*, falls es einen Term $r \in \Lambda$ gibt, so dass*

$$r\,\underline{n_1}\ldots\underline{n_p} = \underline{f(n_1,\ldots,n_p)}$$

für alle $n_1,\ldots,n_p \in \mathbb{N}$.

Wir schreiben auch kürzer $r\,\underline{\vec{n}} = \underline{f(\vec{n})}$.

Bemerkung 2.9 *Da $\longrightarrow$ konfluent und eine Church-Ziffer normal ist, ist f λ-definiert durch r gdw. $r\,\underline{\vec{n}} \longrightarrow^* \underline{f(\vec{n})}$.*

Definition 2.10 (Basisfunktionen) *Basisfunktionen sind Null-, Nachfolger- und Projektionsfunktionen π_i^p, wobei $1 \leq i \leq p$:*

$$\begin{array}{llllll} \mathit{zero} & \in & \mathbb{N}^0 \to \mathbb{N} & \mathit{zero}() & = & 0 \\ \mathit{succ} & \in & \mathbb{N}^1 \to \mathbb{N} & \mathit{succ}(n) & = & n+1 \\ \pi_i^p & \in & \mathbb{N}^p \to \mathbb{N} & \pi_i^p(n_1,\ldots,n_p) & = & n_i \end{array}$$

Die Basisfunktionen sind total.

Definition 2.11 (Kombinierte Funktionen)

1. *Ist f eine p-stellige und $g_1,\ldots,g_p$ q-stellige partielle numerische Funktionen, so ist $f \circ \vec{g}$ die q-stellige partielle numerische Funktion definiert durch*
$$(f \circ \vec{g})(\vec{n}) = f(g_1(\vec{n}),\ldots,g_p(\vec{n})).$$

2. *Ist f eine p-stellige und g eine $p+2$-stellige partielle numerische Funktion, so ist $\mathit{PrimRec}_{f,g}$ eine $p+1$-stellige partielle numerische Funktion definiert durch*
$$\begin{array}{lll} \mathit{PrimRec}_{f,g}(0,\vec{n}) & = & f(\vec{n}) \\ \mathit{PrimRec}_{f,g}(m+1,\vec{n}) & = & g(m,\mathit{PrimRec}_{f,g}(m,\vec{n}),\vec{n}) \end{array}$$

3. *Ist f eine $p+1$-stellige partielle numerische Funktion, dann ist μf eine p-stellige partielle numerische Funktion definiert durch:*
$$(\mu f)(\vec{n}) = \min\{m \mid f(\vec{n},m) = 0\}$$

Definition 2.12 (μ-rekursive Funktion) *Die Mengen $\mathcal{T}_p$ ($p \in \mathbb{N}$) der totalen μ-rekursiven p-stelligen numerischen Funktionen sind simultan-induktiv wie folgt definiert.*

$$\mathcal{T}\text{-ZERO}\ \frac{}{zero \in \mathcal{T}_0} \qquad \mathcal{T}\text{-SUCC}\ \frac{}{succ \in \mathcal{T}_1} \qquad \mathcal{T}\text{-PROJ}\ \frac{}{\pi_i^p \in \mathcal{T}_p}\ 1 \le i \le p$$

$$\mathcal{T}\text{-COMP}\ \frac{f \in \mathcal{T}_p \qquad g_1, \ldots, g_p \in \mathcal{T}_q}{f \circ \vec{g} \in \mathcal{T}_q} \qquad \mathcal{T}\text{-PRIMREC}\ \frac{f \in \mathcal{T}_p \qquad g \in \mathcal{T}_{p+2}}{PrimRec_{f,g} \in \mathcal{T}_{p+1}}$$

$$\mathcal{T}\text{-MUREC}\ \frac{f \in \mathcal{T}_{p+1}}{\mu f \in \mathcal{T}_p}\ \forall \vec{n} \exists m.\ f(\vec{n}, m) = 0$$

Wir zeigen nun, dass alle μ-rekursiven Funktionen λ-definierbar sind.

Lemma 2.13 *Die Basisfunktionen sind λ-definierbar.*

Beweis. Die definierenden Terme sind:

$$\begin{array}{lcl} \mathsf{zero} & \equiv & \underline{0} \\ \mathsf{succ} & \equiv & \lambda n \lambda s \lambda z.\, s\,(n\,s\,z) \\ \mathsf{proj}_i^p & \equiv & \lambda x_1 \ldots x_p.\, x_i \end{array}$$

□

Für die Implementation der primitiven Rekursion benützen wir die Notation $\langle r, s\rangle$ für $\mathsf{Pair}\, r\, s$ und $\lambda\langle x, y\rangle.t$ für $\lambda p.\ t[(\mathsf{snd}\, p)/y][(\mathsf{fst}\, p)/x]$.

Lemma 2.14 *Die λ-definierbaren Funktionen sind abgeschlossen unter Komposition.*

Beweis.

$$\mathsf{comp}_q^p \equiv \lambda f\, g_1 \ldots g_p\, x_1 \ldots x_q.\ f\,(g_1\,\vec{x}) \ldots (g_p\,\vec{x})$$

□

Lemma 2.15 *Die λ-definierbaren Funktionen sind abgeschlossen unter primitiver Rekursion.*

Beweis. Primitive Rekursion lässt sich mittels Paaren durch *Iteration* simulieren, also durch ein LOOP-Programm. Iteration ist wiederum in Church-Ziffern eingebaut. Zuerst also das LOOP-Programm:

$\mathsf{primrec}_{f,g}^p(m', n_1, \ldots, n_p)$
  $\langle m, r\rangle := \langle 0, f(\vec{n})\rangle$
  LOOP m'
    $\langle m, r\rangle := \langle m+1,\ g(m, r, \vec{n})\rangle$
  RETURN r

Da es die LOOP-Konstruktion uns nicht ermöglicht, abzufragen, in welchen Schleifendurchlauf wir uns gerade befinden, wir die Information aber für den Aufruf von g brauchen, beschaffen wir uns sie mittels einer Variable m, die wir bei jedem Druchlauf hochzählen. Im Paar $\langle m, r\rangle$ enthält die Variable r jeweils das Ergebnis der primitiven Rekursion über m.

Ebenso erzeugen wir in der λ-Kalkül-Version ein Paar $\langle m, r\rangle$ durch Iteration über m'. Nach Beendigung der Iteration geben wir die zweite Komponente zurück. Wir definieren:

$$\begin{array}{lcl} \mathsf{primrec}^p & \equiv & \lambda f\, g\, m'\, n_1 \ldots n_p.\ \mathsf{snd}\,(m'\ \mathsf{step}\ \mathsf{start}) \\ & & \text{wobei}\quad \mathsf{start} \equiv \langle \mathsf{zero},\ f\,\vec{n}\rangle \\ & & \qquad\quad \mathsf{step} \equiv \lambda\langle m, r\rangle.\ \langle \mathsf{succ}\ m,\ g\, m\, r\,\vec{n}\rangle \end{array}$$

Durch Induktion über $m \in \mathbb{N}$ beweisen wir simultan

$$\begin{array}{lcl} \underline{m}\ \mathsf{step}\ \mathsf{start} & = & \langle \underline{m},\ \mathsf{primrec}^p\ f\ g\ \underline{m}\ \vec{n}\rangle \\ \mathsf{primrec}^p\ f\ g\ \underline{m}\ \vec{n} & = & \begin{cases} f\,\vec{n} & \text{falls } m = 0 \\ g\,\underline{m'}\,(\mathsf{primrec}^p\ f\ g\ \underline{m'}\ \vec{n})\,\vec{n} & \text{falls } m = m'+1 \end{cases} \end{array}$$

Basisfall $m = 0$.

$$\begin{array}{lcl} \underline{0}\ \mathsf{step}\ \mathsf{start} & = & \mathsf{start} \\ & = & \langle \underline{0},\ f\,\vec{n}\rangle \\ \mathsf{primrec}^p\ f\ g\ \underline{0}\ \vec{n} & = & \mathsf{snd}\,(\underline{0}\ \mathsf{step}\ \mathsf{start}) \\ & = & f\,\vec{n} \end{array}$$

Schrittfall $m = m' + 1$.

$$\begin{array}{lcl} \underline{m'+1}\ \mathsf{step}\ \mathsf{start} & = & \mathsf{step}^{m'+1}\ \mathsf{start} \\ & = & \mathsf{step}\ (\underline{m'}\ \mathsf{step}\ \mathsf{start}) \\ & = & \mathsf{step}\ \langle \underline{m'},\ \mathsf{primrec}^p\ f\ g\ \underline{m'}\ \vec{n}\rangle \\ & = & \langle \underline{m'+1},\ g\,\underline{m'}\ (\mathsf{primrec}^p\ f\ g\ \underline{m'}\ \vec{n})\,\vec{n}\rangle \\ \mathsf{primrec}^p\ f\ g\ \underline{m'+1}\ \vec{n} & = & \mathsf{snd}\,(\underline{m'+1}\ \mathsf{step}\ \mathsf{start}) \\ & = & g\,\underline{m'}\ (\mathsf{primrec}^p\ f\ g\ \underline{m'}\ \vec{n}) \end{array}$$

In diesem Beweis haben wir primitive Rekursion auf Iteration zurückgeführt. Man kann primitive Rekursion aber auch mit allgemeiner Rekursion implementieren:

$$\begin{array}{lcl} \mathsf{primrec}^p\ f\ g\ m\ \vec{n} & = & \mathsf{iszero}\ m \\ & & \quad (f\,\vec{n}) \\ & & \quad (g\,(\mathsf{pred}\,m)\,(\mathsf{primrec}^p\ f\ g\ (\mathsf{pred}\,m)\ \vec{n})\,\vec{n}) \end{array}$$

Rekursion kam man mit einem Fixpunkt-Kombinator abbilden, zusätzlich braucht man noch Test auf Null (iszero) und Vorgänger (pred). Da der Vorgänger aber selbst schon eine Iteration über m benötigt, wäre das ziemlich ineffizient. Wie wir gesehen haben, kann man ja die ganze primitive Rekursion auf eine Iteration zurückführen. □

Lemma 2.16 *Die λ-definierbaren Funktionen sind abgeschlossen unter μ-Rekursion (Minimierung).*

Beweis. Wir implementieren $(\mu f)(\vec{n}) = g(0)$, wobei g rekursiv definiert ist:

$$g(m) = \begin{cases} m & \text{falls } f(\vec{n}, m) = 0 \\ g(m+1) & \text{andernfalls} \end{cases}$$

Im λ-Kalkül sieht das dann so aus:

$$\mathsf{murec}^p \equiv \lambda f\, n_1 \ldots n_p.\ (\mathsf{Y}(\lambda g\, m.\ \mathsf{iszero}\ (f\, \vec{n}\, m)\ m\ (g\ (\mathsf{succ}\ m))))\ \mathsf{zero}$$

□

Satz 2.17 (μ-rekursiv impliziert λ-definierbar) *Wenn $f \in \mathcal{T}_p$, dann ist f λ-definierbar.*

Beweis. Durch Induktion über $f \in \mathcal{T}_p$, unter Benutzung der Lemmata. □

Kapitel 3

Reduktion

3.1 β-Reduktion

Definition 3.1 (β-Reduktion) *Wir definieren die* Ein-Schritt-β-Reduktion $\longrightarrow_\beta \subseteq \Lambda \times \Lambda$ *induktiv durch die folgenden Regeln:*

$$\text{RED}_\beta\text{-AX}\ \frac{}{(\lambda x t)\, s \longrightarrow_\beta t[s/x]}$$

$$\text{RED}_\beta\text{-}\xi\ \frac{t \longrightarrow_\beta t'}{\lambda x t \longrightarrow_\beta \lambda x t'} \qquad \text{RED}_\beta\text{-APP-L}\ \frac{r \longrightarrow_\beta r'}{r\, s \longrightarrow_\beta r'\, s} \qquad \text{RED}_\beta\text{-APP-R}\ \frac{s \longrightarrow_\beta s'}{r\, s \longrightarrow_\beta r\, s'}$$

Einen Term der Form $(\lambda x t)\, s$ *bezeichnen wir als* Redex *(*red*ucible* ex*pression), und* $t[s/x]$ *als sein* Redukt*.*

$\longrightarrow_\beta$ ist also die kompatible Hülle der *β-Kontraktion* $t\ \beta\ t'$, die gegeben ist durch das Schema RED_β-AX.

Aus Lemma 1.29 folgt:

Korollar 3.2 *Die Relationen* $\longrightarrow_\beta^+$ *und* $\longrightarrow_\beta^*$ *sind kompatibel.*

3.1.1 Substitutivität

Definition 3.3 (Substitutivität) *Eine Relation* $R \subseteq \Lambda \times \Lambda$ *auf* λ*-Termen heisst* substitutiv*, falls* $t\ R\ t'$ *impliziert, dass* $t[s/x]\ R\ t'[s/x]$ *für alle* $s \in \Lambda$ *und* $x \in \mathsf{V}$*.*

Lemma 3.4 *Ist eine Relation R substitutiv, dann auch ihre reflexive, transitive, symmetrische, und kompatible Hülle.*

Lemma 3.5 *Die β-Kontraktion ist substitutiv.*

Beweis. Sei $(\lambda xt)\, s \;\beta\; t[s/x]$. Dann gilt $((\lambda xt)\, s)[r/y] = (\lambda x.\, t[r/y])\, (s[r/y]) \;\beta\; t[r/y][s[r/y]/x] = t[s/x][r/y]$ nach den Rechenregeln für Substitution. □

Korollar 3.6 *Die Relationen* $\longrightarrow_\beta$, $\longrightarrow_\beta^+$, $\longrightarrow_\beta^*$, *und* $=_\beta$ *sind substitutiv.*

3.2 Konfluenz

3.2.1 Konfluenzeigenschaften und deren Beziehung

Definition 3.7 (Konfluenz) *Sei M eine Menge und $\longrightarrow \subseteq M \times M$ eine Relation auf M.*

1. $\longrightarrow$ *hat die* Diamanteigenschaft *falls gilt: Wenn* $t_0 \longrightarrow t_1$ *und* $t_0 \longrightarrow t_2$ *dann gibt es ein* t_3 *mit* $t_1 \longrightarrow t_3$ *und* $t_2 \longrightarrow t_3$.

2. $\longrightarrow$ *hat die* Streifeneigenschaft *falls gilt: Wenn* $t_0 \longrightarrow t_1$ *und* $t_0 \longrightarrow^* t_2$ *dann gibt es ein* t_3 *mit* $t_1 \longrightarrow^* t_3$ *und* $t_2 \longrightarrow^* t_3$.

3. $\longrightarrow$ *ist* konfluent, *falls gilt: Wenn* $t_0 \longrightarrow^* t_1$ *und* $t_0 \longrightarrow^* t_2$ *dann gibt es ein* t_3 *mit* $t_1 \longrightarrow^* t_3$ *und* $t_2 \longrightarrow^* t_3$.

4. $\longrightarrow$ *ist* lokal konfluent, *falls gilt: Wenn* $t_0 \longrightarrow t_1$ *und* $t_0 \longrightarrow t_2$ *dann gibt es ein* t_3 *mit* $t_1 \longrightarrow^* t_3$ *und* $t_2 \longrightarrow^* t_3$.

Alternativ sagt man für konfluent *auch* Church-Rosser *und für* lokal konfluent *auch* schwach Church-Rosser.

Trivialerweise ist jede konfluente Relation auch lokal konfluent. Auch Diamant- und Streifeneigenschaft implizieren direkt die lokale Konfluenz. Jede konfluente Relation hat die Streifeneigenschaft.

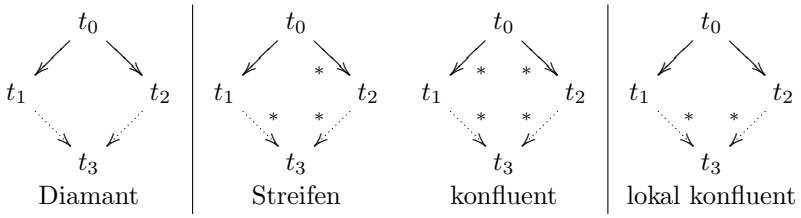


Abbildung 3.1: Konfluenzeigenschaften

Lemma 3.8 *Hat* $\longrightarrow$ *die Diamanteigenschaft, dann auch die Streifeneigenschaft.*

Beweis. Sei $t_0 \longrightarrow^* t_2$. Durch Induktion über die Anzahl der Reduktionsschritte zeigen wir, dass es für alle t_1 mit $t_0 \longrightarrow t_1$ ein t_3 gibt mit $t_1 \longrightarrow^* t_3$ und $t_2 \longrightarrow t_3$. □

Lemma 3.9 *Hat $\longrightarrow$ die Streifeneigenschaft, dann ist $\longrightarrow$ konfluent.*

Beweis. Sei $t_0 \longrightarrow^* t_1$. Durch Induktion über die Anzahl der Reduktionsschritte zeigen wir, dass es für alle t_2 mit $t_0 \longrightarrow^* t_2$ ein t_3 gibt mit $t_1 \longrightarrow^* t_3$ und $t_2 \longrightarrow^* t_3$. □

Korollar 3.10 *Hat $\longrightarrow$ die Diamanteigenschaft, dann ist $\longrightarrow$ konfluent.*

Beweis. (Siehe Abbildung 3.2) Durch Zusammensetzen der beiden Lemmata. □

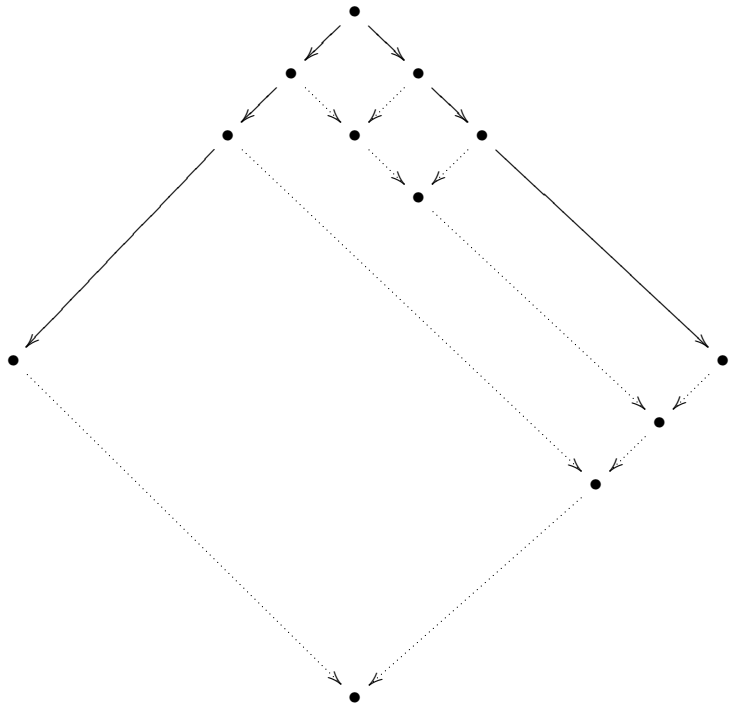


Abbildung 3.2: Beweis der Konfluenz aus der Diamanteigenschaft

Bemerkung 3.11 *Nicht jede lokal konfluente Relation ist auch konfluent! Standardbeispiel ist die folgende Relation:*

Aber in Abwesenheit von unendlichen Reduktionsfolgen ist lokale Konfluenz ausreichend:

Lemma 3.12 (Newman) *Ist $\longrightarrow$ eine lokal konfluente Relation und wohlfundiert, d.h. es gibt keine unendlichen Ketten $t_0 \longrightarrow t_1 \longrightarrow \ldots$, so ist $\longrightarrow$ schon konfluent.*

Beweis. Später. □

Beispiel 3.13 (Nicht-Konfluenz Klop (1980)) *Das folgende Termersetzungssystem ist nicht konfluent:*

$$\begin{array}{lcl} \delta\, t\, t & \longrightarrow & \epsilon \\ \mathsf{c}\, t & \longrightarrow & \delta\, t\, (\mathsf{c}\, t) \\ \mathsf{a} & \longrightarrow & \mathsf{c}\, \mathsf{a} \end{array}$$

Dazu betrachte man folgende Reduktionsfolgen ausgehend von $\mathsf{c}\,\mathsf{a}$*:*

$$\begin{array}{l} \mathsf{c}\,\mathsf{a} \longrightarrow^+ \delta\, \mathsf{a}\, (\mathsf{c}\,\mathsf{a}) \longrightarrow^+ \delta\, (\mathsf{c}\,\mathsf{a})\, (\mathsf{c}\,\mathsf{a}) \longrightarrow^+ \epsilon \\ \mathsf{c}\,\mathsf{a} \longrightarrow^+ \mathsf{c}\, (\mathsf{c}\,\mathsf{a}) \longrightarrow^+ \mathsf{c}\, \epsilon \end{array}$$

Diese beiden Redukte können nicht mehr zusammengeführt werden.

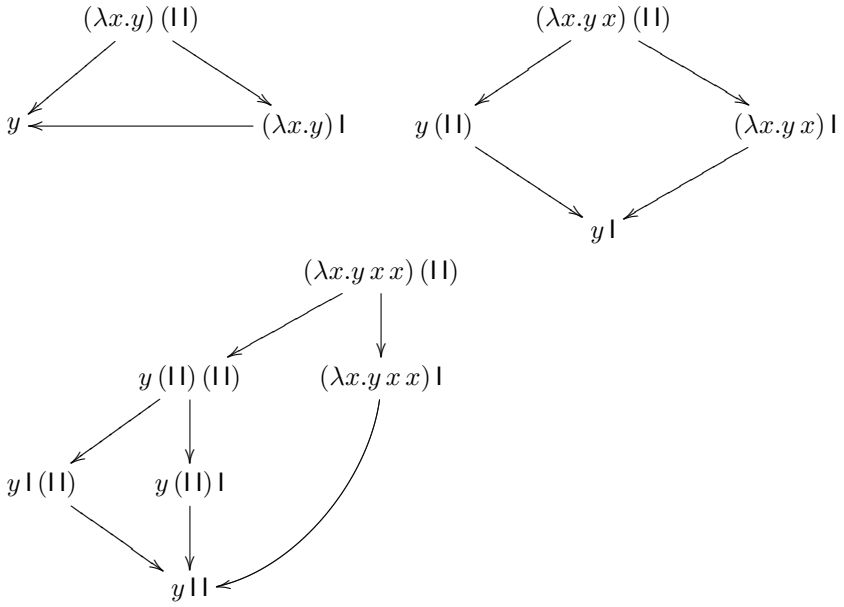
Die Symbole c *und* a *können auch mittels des Turing'schen Fixpunkt-Kombinators $\Phi = (\lambda x \lambda f.\, f\, (x\, x\, f))\, (\lambda x \lambda f.\, f\, (x\, x\, f))$ und β-Reduktion implementiert werden. "Schuld" an der Nicht-Konfluenz ist die sog.* nicht-lineare *linke Seite der Regel $\delta\, t\, t \longrightarrow \epsilon$ (zwei Vorkommen desselben Terms t).*

3.2.2 Lokale Konfluenz der β-Reduktion

Satz 3.14 *$\longrightarrow_\beta$ hat nicht die Diamanteigenschaft.*

Beweis. Betrachten wir das Reduktionspaar $t[s/x] \longleftarrow (\lambda x t)\, s \longrightarrow (\lambda x t)\, s'$ wobei $s \longrightarrow s'$. Falls die Substitution für x den eingesetzen Term vervielfacht oder verliert, kann $t[s/x]$ nicht in genau einem Schritt auf $t[s'/x]$ reduziert wer-

den, sondern nur in mehreren Schritten oder keinem. Beispiele:



□

Satz 3.15 ($\longrightarrow_\beta$ ist lokal konfluent) *Wenn $t_0 \longrightarrow_\beta t_1$ und $t_0 \longrightarrow_\beta t_2$ dann gibt es ein t_3 mit $t_1 \longrightarrow_\beta^* t_3$ und $t_2 \longrightarrow_\beta^* t_3$.*

Beweis. Durch simultane Induktion über $t_0 \longrightarrow_\beta t_1$ und $t_0 \longrightarrow_\beta t_2$. Wir betrachten folgende Fälle:

Fall Reduktion unter λ:

$$\frac{t_1 \;{}_\beta\!\longleftarrow t_0}{\lambda x t_1 \;{}_\beta\!\longleftarrow \lambda x t_0} \qquad \frac{t_0 \longrightarrow_\beta t_2}{\lambda x t_0 \longrightarrow_\beta \lambda x t_2}$$

Nach Induktionsvoraussetzung gibt es ein t_3 mit $t_1 \longrightarrow_\beta^* t_3$ und $t_2 \longrightarrow_\beta^* t_3$. Mit der Kompatibilität von $\longrightarrow_\beta^*$ auch $\lambda x t_1 \longrightarrow_\beta^* \lambda x t_3$ und $\lambda x t_2 \longrightarrow_\beta^* \lambda x t_3$.

Fall Unabhängige Reduktionen in Applikation:

$$\frac{r' \;{}_\beta\!\longleftarrow r}{r'\,s \;{}_\beta\!\longleftarrow r\,s} \qquad \frac{s \longrightarrow_\beta s'}{r\,s \longrightarrow_\beta r\,s'}$$

Beide Redukte reduzieren zu $r'\,s'$ in einem Schritt.

Fall Reduktion im Argument vs. β:

$$\frac{s' \;{}_\beta\!\longleftarrow s}{(\lambda x t)\,s' \;{}_\beta\!\longleftarrow (\lambda x t)\,s} \qquad \frac{}{(\lambda x t)\,s \longrightarrow_\beta t[s/x]}$$

Nach Korollar 3.23, $t[s/x] \longrightarrow_\beta^* t[s'/x]$. Ausserdem, $(\lambda x t)\, s' \longrightarrow_\beta t[s'/x]$. An diesem Fall scheitert die Diamanteigenschaft von $\longrightarrow_\beta$.

Fall Reduktion in der Funktion vs. β:

$$\frac{t' \;{}_\beta\!\longleftarrow t}{(\lambda x t')\, s \;{}_\beta\!\longleftarrow (\lambda x t)\, s} \qquad \frac{}{(\lambda x t)\, s \longrightarrow_\beta t[s/x]}$$

Aus der Substitutivität von $\longrightarrow_\beta$ erhalten wir $t[s/x] \longrightarrow_\beta t'[s/x]$. Ausserdem $(\lambda x t')\, s \longrightarrow_\beta t'[s/x]$.

□

3.2.3 Konfluenz der β-Reduktion

Sektion 3.2.1 deutet eine Strategie an, um die Konfluenz einer Relation zu beweisen: Man beweise zuerst die Diamanteigenschaft, die dann Streifeneigenschaft und Konfluenz impliziert. In Sektion 3.2.2 haben wir gesehen, dass die β-Reduktion wegen der Duplikation von Redexen die Diamanteigenschaft nicht hat. Wir müssen daher die Ein-Schritt-Reduktion so verstärken, dass mehrere, von einander unabhängige Reduktionen gleichzeitig durchgeführt werden können. Wenn z.B. s in einem Schritt auf s' reduziert, so wollen wir in eine Schritt auch $t[s/x]$ auf $t[s'/x]$ reduzieren können. Eine Verstärkung der Ein-Schritt-Reduktion, die diese Eigenschaft hat, ist die *parallele Reduktion* nach Tait und Martin-Löf. Sie erlaubt die gleichzeitige Reduktion einer beliebigen Menge von Redexen, die im Moment sichtbar sind. Z.B. sind im Term $(\mathsf{I}\,(\mathsf{K}\,x))\,(\mathsf{I}\,y)$ die 3 Redexe $\mathsf{I}\,(\mathsf{K}\,x)$, $\mathsf{K}\,x$ und $\mathsf{I}\,y$ sichtbar. Werden diese gleichzeitig entfernt, ensteht der Term $(\lambda z.\, x)\, y$. Dieser Redex ist neu, er war vorher nicht sichtbar, und kann nur in einem zweiten Reduktionsschritt entfernt werden.

Die parallele Reduktion hat die Diamanteigenschaft, was aus der folgenden einfachen Überlegung hervorgeht: Betrachten wir einen Term t_0, und zwei parallele Redukte t_1 und t_2. Der erste Term t_1 geht aus t_0 durch die Entfernung einer gewissen Menge von sichtbaren Redexen hervor, und ebenso entsteht t_2 aus t_0 durch die Entfernung einer möglicherweise verschiedenen Menge von Redexen. Sowohl t_1 als auch t_2 lassen sich aber in einem parallelen Schritt auf den Term t_3 reduzieren, der durch die Entfernung *aller* sichtbaren Redexe von t_0 entsteht. Wir nennen t_3 die vollständige Entwicklung von t_0.

Im Folgenden formalisieren wir den nun informell vorgestellten Konfluenzbeweis. Dazu definieren wir $\Longrightarrow$, eine Relation $\longrightarrow_\beta \subseteq \Longrightarrow \subseteq \longrightarrow_\beta^*$, die zwischen $\longrightarrow_\beta$ und $\longrightarrow_\beta^*$ liegt und die Diamanteigenschaft hat. Daraus folgt schon die Konfluenz von $\longrightarrow_\beta^*$. Die Definition der parallelen Reduktion und der Konfluenz-Beweis geht auf Tait und Martin-Löf zurück und ist z.B. bei Takahashi Takahashi (1995) aufgeschrieben.

Lemma 3.16 *Sei $\longrightarrow \subseteq \Longrightarrow \subseteq \longrightarrow^*$ und $\Longrightarrow$ konfluent. Dann ist $\longrightarrow^*$ auch konfluent.*

Beweis. Trivial, da $\longrightarrow^* \subseteq \Longrightarrow^* \subseteq \longrightarrow^*$. □

Definition 3.17 (Parallele Reduktion) *Die Relation $\Longrightarrow$ ist induktiv definiert durch:*

$$\text{PAR-VAR}\ \frac{}{x \Longrightarrow x} \qquad \text{PAR-}\xi\ \frac{t \Longrightarrow t'}{\lambda x t \Longrightarrow \lambda x t'} \qquad \text{PAR-APP}\ \frac{r \Longrightarrow r' \qquad s \Longrightarrow s'}{r\, s \Longrightarrow r'\, s'}$$

$$\text{PAR-}\beta\ \frac{t \Longrightarrow t' \qquad s \Longrightarrow s'}{(\lambda x t)\, s \Longrightarrow t'[s'/x]}$$

Hinweis: Die Relation ist nicht-deterministisch.

Definition 3.18 (Parallel-kompatibel) *Eine Relation $R \subseteq \Lambda \times \Lambda$ heisse* parallel-kompatibel*, falls*

1. *$x\ R\ x$ für alle $x \in \mathsf{V}$,*
2. *$t\ R\ t'$ impliziert $\lambda x t\ R\ \lambda x t'$, und*
3. *$r\ R\ r'$ und $s\ R\ s'$ implizieren $r\, s\ R\ r'\, s'$.*

Die parallele Reduktion is parallel-kompatibel dank PAR-VAR, PAR-ξ, und PAR-APP.

Lemma 3.19 (Reflexivität) *Ist R parallel-kompatibel, so gilt $t\ R\ t$ für alle $t \in \Lambda$.*

Beweis. Durch strukturelle Induktion über t. □

Lemma 3.20 (Einbettung von $\longrightarrow_\beta$) *Wenn $t \longrightarrow_\beta t'$, dann $t \Longrightarrow t'$.*

Beweis. Durch Induktion über $t \longrightarrow_\beta t'$ unter Benutzung der Reflexivität. □

Lemma 3.21 (Einbettung in $\longrightarrow_\beta^*$) *Wenn $t \Longrightarrow t'$, dann $t \longrightarrow_\beta^* t'$.*

Beweis. Durch Induktion über $t \Longrightarrow t'$ unter Benutzung der Abschlusseigenschaften von $\longrightarrow_\beta^*$.

Fall

$$\text{PAR-}\beta\ \frac{t \Longrightarrow t' \qquad s \Longrightarrow s'}{(\lambda x t)\, s \Longrightarrow t'[s'/x]}$$

Nach Induktionsvoraussetzung $t \longrightarrow_\beta^* t'$ und $s \longrightarrow_\beta^* s'$, zusammen also $(\lambda x t)\, s \longrightarrow_\beta^* (\lambda x t')\, s' \longrightarrow_\beta t'[s'/x]$.

□

Lemma 3.22 (Abschluss unter Substitution) *Falls $s \Longrightarrow s'$ und $r \Longrightarrow r'$, dann $r[s/x] \Longrightarrow r'[s'/x]$.*

Beweis. Durch Induktion über $r \Longrightarrow r'$. De facto ersetzen wir jedes Blatt $x \Longrightarrow x$ des Herleitungsbaums $r \Longrightarrow r'$ durch die Herleitung von $s \Longrightarrow s'$. □

Korollar 3.23 (Abschluss von $\longrightarrow_\beta^*$ unter Substitution)

1. *Falls $s \longrightarrow_\beta s'$, then $t[s/x] \longrightarrow_\beta^* t[s'/x]$.*
2. *Falls $s \longrightarrow_\beta^* s'$ and $t \longrightarrow_\beta^* t'$, then $t[s/x] \longrightarrow_\beta^* t'[s'/x]$.*

Beweis. 1. folgt aus dem vorherigen Lemma, da $s \Longrightarrow s'$ und $t \Longrightarrow t$. 2. folgt aus 1. und der Substitutivität von $\longrightarrow_\beta^*$: Betrachte das Zwischenprodukt $t[s'/x]$. □

Definition 3.24 (Vollständige Entwicklung) *Für $t \in \Lambda$ definieren wir die vollständige Entwicklung t^* durch Rekursion über t.*

$$\begin{array}{lcl} x^* & = & x \\ (\lambda x t)^* & = & \lambda x.\, t^* \\ (r\, s)^* & = & \begin{cases} t^*[s^*/x] & \text{falls } r = \lambda x t \\ r^*\, s^* & \text{sonst} \end{cases} \end{array}$$

Lemma 3.25 *Wenn $r' \Longrightarrow r^*$ und $s' \Longrightarrow s^*$, dann $r'\, s' \Longrightarrow (r\, s)^*$.*

Beweis. Durch Unterscheidung der Fälle, ob r' eine λ-Abstraktion ist oder nicht. □

Lemma 3.26 (Maximale Verlängerung einer parallelen Reduktion) *Falls $t \Longrightarrow t'$, dann $t' \Longrightarrow t^*$.*

Beweis. Durch Induktion über $t \Longrightarrow t'$, unter Benutzung des vorherigen Lemmas im Fall PAR-APP, und des Substitutions-Lemmas in Fall PAR-β.

Fall

$$\text{PAR-APP}\ \frac{r \Longrightarrow r' \qquad s \Longrightarrow s'}{r\, s \Longrightarrow r'\, s'}$$

Nach Induktionsvoraussetzung $r' \Longrightarrow r^*$ und $s' \Longrightarrow s^*$. Nach dem Lemma also $r'\, s' \Longrightarrow (r\, s)^*$.

Fall

$$\text{PAR-}\beta\ \frac{t \Longrightarrow t' \qquad s \Longrightarrow s'}{(\lambda x t)\, s \Longrightarrow t'[s'/x]}$$

Nach Induktionsvoraussetzung $s' \Longrightarrow s^*$ und $t' \Longrightarrow t^*$. Mit dem Substitutionslemma, $t'[s'/x] \Longrightarrow t^*[s^*/x] = ((\lambda x t)\, s)^*$.

□

Korollar 3.27 $\Longrightarrow$ *hat die Diamanteigenschaft.*

Beweis. Falls $t_0 \Longrightarrow t_1$ und $t_0 \Longrightarrow t_2$, so gilt nach dem vorigen Lemma, $t_1 \Longrightarrow t_0^*$ und $t_2 \Longrightarrow t_0^*$. □

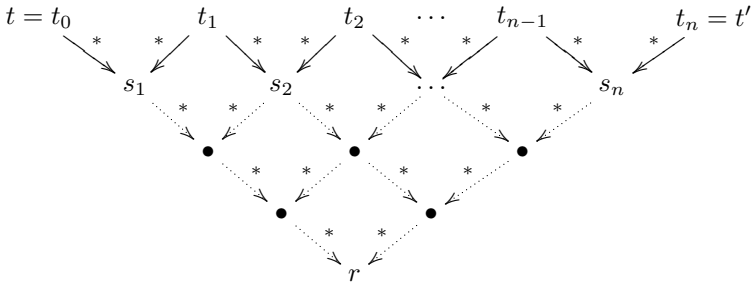
Satz 3.28 $\longrightarrow_\beta$ *ist konfluent.*

Beweis. Durch Kombination der Teilaussagen. □

Damit können wir der β-Gleichheit eine Richtung geben.

Korollar 3.29 (Richtung der β-Gleichheit) *Wenn $t =_\beta t'$, dann gibt es einen Term r mit $t \longrightarrow_\beta^* r$ und $t' \longrightarrow_\beta^* r$.*

Beweis. $=_\beta$ is die transitive Hülle der symmetrischen Hülle von $\longrightarrow_\beta^*$. Also gibt es Terme $t_0 = t$, t_1, t_2, …, $t_n = t'$ und Terme $s_1, \ldots, s_n$, so dass $t_i \longrightarrow_\beta^* s_{i+1}$ ${}_\beta^*\!\longleftarrow t_{i+1}$. Durch $n(n-1)/2$-maliges Anwenden der Konfluenz erhält man daraus den gesuchten Term r.



□

3.3 Normalformen

Eine wichtige Konsequenz aus der Konfluenz ist die Eindeutigkeit von Normalformen.

Satz 3.30 (Eindeutigkeit von Normalformen) *Ist $\longrightarrow$ konfluent, dann hat jedes $t \in \Lambda$ höchstens eine Normalform.*

Beweis. Angenommen $t \longrightarrow^* t_1 \not\longrightarrow$ und $t \longrightarrow^* t_2 \not\longrightarrow$. Dann gibt es nach Konfluenz ein t_3 mit $t_1 \longrightarrow^* t_3$ und $t_2 \longrightarrow^* t_3$. Da t_1 und t_2 normal sind, bestehen die letztgenannten Reduktionssequenzen aus genau 0 Schritten, also $t_1 \equiv t_3 \equiv t_2$. □

Lemma 3.31 (Charakterisierung von β-Normalformen)

1. *$x \not\longrightarrow_\beta$.*
2. *$\lambda x t \not\longrightarrow_\beta$ gdw. $t \not\longrightarrow_\beta$.*
3. *$r\,s \not\longrightarrow_\beta$ gdw. $r \not\longrightarrow_\beta$, $s \not\longrightarrow_\beta$ und r keine λ-Abstraktion.*

Mit Hilfe dieser Charakterisierung können wir Normalformen induktiv definieren.

Definition 3.32 (Induktive Charakterisierung von Normalformen) *Wir definieren die Menge $\mathsf{NF} \subseteq \Lambda$ induktiv wie folgt:*

$$\frac{}{x \in \mathsf{NF}} \qquad \frac{r \in \mathsf{NF} \qquad s \in \mathsf{NF}}{r\,s \in \mathsf{NF}}\ r \neq \lambda x t \qquad \frac{t \in \mathsf{NF}}{\lambda x t \in \mathsf{NF}}$$

Lemma 3.33 (Korrektheit und Vollständigkeit von NF) *$t \in \mathsf{NF}$ g.d.w. $t \not\longrightarrow_\beta$.*

Beweis. Durch Induktion nach t. □

Lemma 3.34 *Falls $t \in \mathsf{NF}$ und t ist keine Abstraktion, dann $t = x\,\vec{s}$ für ein $x \in \mathsf{V}$ und einen (möglicherweise leeren) Argumentvektor $\vec{s} \in \mathsf{NF}$.*

Mit Hilfe der Vektornotation lässt sich NF wie folgt definieren:

Lemma 3.35 (Alternative induktive Charakterisierung von Normalformen) *Wir definieren die Menge $\mathsf{NF}' \subseteq \Lambda$ induktiv wie folgt:*

$$\frac{s_1, \ldots, s_n \in \mathsf{NF}'}{x\,\vec{s} \in \mathsf{NF}'}\ n \geq 0 \qquad \frac{t \in \mathsf{NF}'}{\lambda x t \in \mathsf{NF}'}$$

Damit gilt $\mathsf{NF} = \mathsf{NF}'$.

Definition 3.36 (Divergenz) *Ein Term t* divergiert*, falls er keine Normalform besitzt.*

Beispiel 3.37 *Der Term $\Omega := (\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$ divergiert, denn er reduziert nur zu sich selbst. Ebenso divergieren die Terme $\lambda y \Omega$ und $\lambda y.\, y\, \Omega$. Sei $\mathsf{K} := \lambda x \lambda y.\, x$. Der Term $\mathsf{K}\, \mathsf{K}\, \Omega$ reduziert zwar zu sich selbst, allerdings nicht ausschliesslich; er hat die Normalform K.*

3.4 Standardisierung

Hat ein Term eine (β-)Normalform, so ist sie durch Reduktion erreichbar, das sagt uns die Konfluenz. Allerdings gibt es viele Möglichkeiten, einen Term zu reduzieren, und nicht alle bringen uns weiter auf dem Weg zur Normalform. Z.B. der Term $(\lambda xy)\, \Omega$ erlaubt zwei Reduktionen, eine bringt uns zur Normalform y, die andere lässt uns "auf der Stelle treten". Im Folgenden zeigen wir, dass jeder Term eine Standardreduktionsfolge hat, die zur Normalform führt, falls der Term eine besitzt.

Lemma 3.38 (Charakterisierung von Termen I) *Jeder Term hat eine der folgenden Gestalten:*

1. $x\, \vec{s}$*,*
2. λxt*, oder*
3. $(\lambda xt)\, s\, \vec{s}$*.*

Terme der Gestalt $x\, \vec{s}$ und λxt sind in schwacher Kopf-Normalform *(*WHNF*), Terme der Gestalt $(\lambda xt)\, s\, \vec{s}$* haben einen schwachen Kopf-Redex*.*

Definition 3.39 (Schwache Kopfreduktion) *Die schwache Kopfreduktion ist gegeben durch das Axiomenschema $(\lambda xt)\, s\, \vec{s} \longrightarrow_{\mathsf{wh}} t[s/x]\vec{s}$. (Keine Kompatibilitätsregeln!)*

Lemma 3.40 (Charakterisierung von Termen II) *Jeder Term hat eine der folgenden Gestalten:*

1. $\lambda \vec{y}.\, x\, \vec{s}$*, oder*
2. $\lambda \vec{y}.\, (\lambda xt)\, s\, \vec{s}$*.*

Im ersten Fall ist der Term in Kopf-Normalform *(*HNF*), im zweiten Fall* hat er einen Kopf-Redex*.*

Definition 3.41 (Kopfreduktion) *Die Kopfreduktion ist gegeben durch das Axiomenschema $\lambda \vec{y}.\, (\lambda xt)\, s\, \vec{s} \longrightarrow_{\mathsf{h}} \lambda \vec{y}.\, t[s/x]\vec{s}$. Alternative Sicht: Abschluss von $\longrightarrow_{\mathsf{wh}}$ unter der ξ-Regel.*

Kopf-Redexe müssen auf jeden Fall entfernt werden, um zu einer Normalform zu gelangen. Andere Redexe können auch durch leere Substitution verschwinden, wie z.B. im Term $(\lambda xy)\,\Omega$ der Redex in Ω.

Betrachen wir eine Reduktionsfolge $t \longrightarrow_\beta^* t'$. Hat t einen schwachen Kopf-Redex und wir entfernen ihn im ersten Schritt nicht, haben wir im zweiten Schritt wieder die Wahl, ob wir ihn entfernen wollen oder nicht. In der *Standard-Reduktionsfolge* entfernen wir einen schwachen Kopf-Redex entweder im ersten Schritt, oder er bleibt bis zum Ende stehen und wir reduzieren nur noch in den Subtermen. Dies ist in folgender induktiver Definition ausgedrückt (Plotkin, 1975, S.146).

Definition 3.42 (Standard-Reduktion, induktiv) *Im Folgenden definieren wir die Relation $t \longrightarrow_{\mathsf{s}} t'$ induktiv.*

$$\textsc{sred-whr}\ \frac{t \longrightarrow_{\mathsf{wh}} t' \qquad t' \longrightarrow_{\mathsf{s}} t''}{t \longrightarrow_{\mathsf{s}} t''} \qquad \textsc{sred-var}\ \frac{}{x \longrightarrow_{\mathsf{s}} x}$$

$$\textsc{sred-xi}\ \frac{t \longrightarrow_{\mathsf{s}} t'}{\lambda x t \longrightarrow_{\mathsf{s}} \lambda x t'} \qquad \textsc{sred-app}\ \frac{r \longrightarrow_{\mathsf{s}} r' \qquad s \longrightarrow_{\mathsf{s}} s'}{r\,s \longrightarrow_{\mathsf{s}} r'\,s'}$$

Bemerkung 3.43 (Alternative Definition) *Durch Integration von* SRED-APP *in* SRED-VAR *und* SRED-XI *und durch Explikation der schwachen Kopfreduktion erhält man folgende, äquivalente Definition (Loader, 1998, S.13):*

$$\textsc{sred-whr}\ \frac{r[s/x]\,\vec{s} \longrightarrow_{\mathsf{s}} t}{(\lambda x r)\,s\,\vec{s} \longrightarrow_{\mathsf{s}} t}$$

$$\textsc{sred-clos-var}\ \frac{\vec{s} \longrightarrow_{\mathsf{s}} \vec{s}'}{x\,\vec{s} \longrightarrow_{\mathsf{s}} x\,\vec{s}'} \qquad \textsc{sred-clos-abs}\ \frac{r \longrightarrow_{\mathsf{s}} r' \qquad \vec{s} \longrightarrow_{\mathsf{s}} \vec{s}'}{(\lambda x r)\,\vec{s} \longrightarrow_{\mathsf{s}} (\lambda x r')\,\vec{s}'}$$

Die Standardreduktion determiniert nicht die Reihenfolge von Reduktionen, die absolut nichts miteinander zu tun haben und deswegen parallel ausgeführt werden können. Z.B. haben im Term $x\,s\,s'$ die Subterme s und s' keine Möglichkeit mehr zur Interaktion, deswegen ist es irrelevant, ob man zuerst s oder s' reduziert. Bleibt ein Kopf-Redex $(\lambda x r)\,s\,\vec{s}$ stehen, interagieren die Subterme $r, s, \vec{s}$ in den folgenden Reduktionen auch nicht mehr.

Lemma 3.44 (Reflexivität) *$t \longrightarrow_{\mathsf{s}} t$ für alle t.*

Beweis. $\longrightarrow_{\mathsf{s}}$ ist parallel-kompatibel. □

Lemma 3.45 (Sequentialisierung) *Wenn $t \longrightarrow_{\mathsf{s}} t'$, dann $t \longrightarrow_\beta^* t'$.*

Beweis. Durch Induktion über $t \longrightarrow_{\mathsf{s}} t'$. □

Nun wollen wir zeigen, dass sich jede Reduktionsfolge $t \longrightarrow_\beta^* t'$ zu $t \longrightarrow_{\mathsf{s}} t'$ standardisieren lässt Loader (1998). Dazu brauchen wir zwei neue Konstruktionsprinzipien für Standardreduktionsfolgen.

Lemma 3.46 (Abschluss unter Substitution) *Wenn* $t \longrightarrow_{\mathsf{s}} t'$ *und* $s \longrightarrow_{\mathsf{s}} s'$*, dann* $t[s/x] \longrightarrow_{\mathsf{s}} t'[s'/x]$.

Beweis. Durch Induktion über $t \longrightarrow_{\mathsf{s}} t'$.

Fall

$$\textsc{sred-whr}\ \frac{t \longrightarrow_{\mathsf{wh}} t' \qquad t' \longrightarrow_{\mathsf{s}} t''}{t \longrightarrow_{\mathsf{s}} t''}$$

Zunächst gilt $t[s/x] \longrightarrow_{\mathsf{wh}} t'[s/x]$, d.h. die schwache Kopfreduktion ist substitutiv, und nach I.H. $t'[s/x] \longrightarrow_{\mathsf{s}} t''[s'/x]$, also folgt die Behauptung $t[s/x] \longrightarrow_{\mathsf{s}} t''[s'/x]$ mit SRED-WHR. □

Nun haben wir die Mittel zusammen, um das zentrale Lemma zu beweisen:

Lemma 3.47 ("Anstückeln" der Standardreduktionsfolge) *Wenn* $t \longrightarrow_{\mathsf{s}} t'$ *und* $t' \longrightarrow_{\beta} t''$*, dann* $t \longrightarrow_{\mathsf{s}} t''$.

Beweis. Durch Induktion über $t \longrightarrow_{\mathsf{s}} t'$ unter Betrachtung der möglichen Fälle der Ein-Schritt-Reduktion $t' \longrightarrow_{\beta} t''$. Widmen wir uns zuerst dem schwierigsten Fall:

Fall

$$\frac{\dfrac{r \longrightarrow_{\mathsf{s}} r'}{\lambda x r \longrightarrow_{\mathsf{s}} \lambda x r}\,\textsc{sred-xi} \qquad s \longrightarrow_{\mathsf{s}} s'}{(\lambda x r)\, s \longrightarrow_{\mathsf{s}} (\lambda x r')\, s'}\,\textsc{sred-app} \quad \text{und } (\lambda x r')\, s' \longrightarrow_{\beta} r'[s'/x]$$

Die Standardreduktion hat den schwachen Kopf-Redex stehengelassen, durch die Ein-Schritt-Reduktion wird er nachträglich entfernt. Um wieder eine Standardreduktion zu erhalten müssen wir diesen Schritt natürlich zu Beginn ausführen. Dazu ist es notwendig, die bisherige Standardreduktionfolge umzubauen: Nach dem Substitutionslemma gilt $r[s/x] \longrightarrow_{\mathsf{s}} r'[s'/x]$, und durch Anwendung von SRED-WHR erhalten wir $(\lambda x r)\, s \longrightarrow_{\mathsf{s}} r'[s'/x]$.

Fall

$$\textsc{sred-xi}\ \frac{r \longrightarrow_{\mathsf{s}} r'}{\lambda x r \longrightarrow_{\mathsf{s}} \lambda x r'} \text{ und } \frac{r' \longrightarrow_{\beta} r''}{\lambda x r' \longrightarrow_{\beta} \lambda x r''}$$

Nach I.V., $r \longrightarrow_{\mathsf{s}} r''$, mit SRED-XI deshalb $\lambda x r \longrightarrow_{\mathsf{s}} \lambda x r''$. Fast alle anderen Fälle gehen analog, bis auf einen letzten, der noch einfacher ist:

Fall

$$\textsc{sred-whr}\ \frac{t \longrightarrow_{\mathsf{wh}} t_0 \qquad t_0 \longrightarrow_{\mathsf{s}} t'}{t \longrightarrow_{\mathsf{s}} t'} \text{ und } t' \longrightarrow_{\beta} t''$$

Hier haben wir nach I.V., $t_0 \longrightarrow_{\mathsf{s}} t''$, und mit SRED-WHR beenden wir den Beweis. □

Satz 3.48 (Standardisierung) *Wenn* $t \longrightarrow_{\beta}^{*} t'$*, dann* $t \longrightarrow_{\mathsf{s}} t'$.

Beweis. Wir starten mit der leeren Standardreduktionsfolge $t \longrightarrow_{\mathsf{s}} t$ und stückeln dann die einzelnen Reduktionsschritte an. □

Korollar 3.49 $\longrightarrow_\beta^* = \longrightarrow_{\mathsf{s}}$.

Damit haben wir ein neues Induktionsprinzip für $\longrightarrow_\beta^*$ gewonnen.

Eine Konsequenz der Standardisierung ist die Vollständigkeit von (schwacher) Kopf-Reduktion: Existiert eine (schwache) Kopf-Normalform, dann findet die (schwache) Kopf-Normalisierung eine.

Lemma 3.50

1. *Wenn* $t \longrightarrow_{\mathsf{s}} x\,\vec{s}'$, *dann* $t \longrightarrow_{\mathsf{wh}}^* x\,\vec{s}$ *und* $\vec{s} \longrightarrow_{\mathsf{s}} \vec{s}'$.
2. *Wenn* $t \longrightarrow_{\mathsf{s}} \lambda x r'$, *dann* $t \longrightarrow_{\mathsf{wh}}^* \lambda x r$ *und* $r \longrightarrow_{\mathsf{s}} r'$.
3. *Wenn* $t \longrightarrow_{\mathsf{s}} \lambda\vec{y}.\,x\,\vec{s}'$, *dann* $t \longrightarrow_{\mathsf{h}}^* \lambda\vec{y}.\,x\,\vec{s}$ *und* $\vec{s} \longrightarrow_{\mathsf{s}} \vec{s}'$.

Beweis. Durch Induktion nach $t \longrightarrow_{\mathsf{s}} \ldots$. □

Satz 3.51 (Korrektheit und Vollständigkeit von (schwacher) Kopf-Reduktion)

1. *Wenn* $t =_\beta w \in \mathsf{WHNF}$, *dann* $t \longrightarrow_{\mathsf{wh}}^* w' \in \mathsf{WHNF}$ *und* $w =_\beta w'$.
2. *Wenn* $t =_\beta h \in \mathsf{HNF}$, *dann* $t \longrightarrow_{\mathsf{h}}^* h' \in \mathsf{HNF}$ *und* $h =_\beta h'$.

Beweis. Aus dem Lemma mittels Konfluenz und Standardisierung. □

3.4.1 Schwache Normalisierung

Definition 3.52 *Ein Term* t *heißt* schwach normalisierend bzgl. einer Reduktionsrelation $\longrightarrow$ *(*$t \in \mathsf{wn}_{\longrightarrow}$*), falls er eine Normalform hat, d.h., es gibt ein* v *mit* $t \longrightarrow v \not\longrightarrow$.

Im folgenden geben wir eine induktive Definition von schwach β-normalisierenden Termen an. Wir wissen bereits:

$$\begin{array}{lcll} t \in \mathsf{wn} & \iff & \exists v \in \mathsf{NF}_\beta.\, t =_\beta v & \\ & \iff & \exists v \in \mathsf{NF}_\beta.\, t \longrightarrow_\beta^* v & \text{Konfluenz} \\ & \iff & \exists v \in \mathsf{NF}_\beta.\, t \longrightarrow_{\mathsf{s}} v & \text{Standardisierung} \end{array}$$

Dabei sind $\longrightarrow_{\mathsf{s}}$ und NF_β bereits induktiv definiert. Wir erhalten die induktive Definition von wn mechanisch durch Einschränkung der rechten Seite von $\longrightarrow_{\mathsf{s}}$ auf Normalformen.

Definition 3.53 (Schwach β-normalisierende Terme, induktiv)

$$\text{WN-NE}\ \frac{\vec{s} \in \mathsf{WN}_\beta}{x\,\vec{s} \in \mathsf{WN}_\beta} \qquad \text{WN-XI}\ \frac{t \in \mathsf{WN}_\beta}{\lambda x t \in \mathsf{WN}_\beta} \qquad \text{WN-WHR}\ \frac{t[s/x]\,\vec{s} \in \mathsf{WN}_\beta}{(\lambda x t)\,s\,\vec{s} \in \mathsf{WN}_\beta}$$

Bemerkung 3.54 *Die dritte Regel lässt sich auch schreiben als*

$$\text{WN-WHR}\ \frac{t \longrightarrow_{\mathsf{wh}} t' \qquad t' \in \mathsf{WN}_\beta}{t \in \mathsf{WN}_\beta}.$$

Nun lassen wir den Index β weg.

Lemma 3.55 *Wenn $t \in \mathsf{WN}$, dann $t \longrightarrow_{\mathsf{s}} v \in \mathsf{NF}$.*

Beweis. Trivial, durch Induktion über $t \in \mathsf{WN}$. □

Lemma 3.56 *Wenn $t \longrightarrow_{\mathsf{s}} v \in \mathsf{NF}$, dann $t \in \mathsf{WN}$.*

Beweis. Trivial, durch Induktion über $t \longrightarrow_{\mathsf{s}} v$. □

Satz 3.57 $\mathsf{wn} = \mathsf{WN}$

Der bisherige Begriff von Standardreduktion enthält Parallelität. Hat ein Term eine Normalform, dann greift sogar eine deterministische Strategie: Entferne immer den am weitesten links stehenden Redex.

Definition 3.58 (Äußere Reduktion (engl. leftmost-outermost reduction))

$$\text{L-WHR}\ \frac{t \longrightarrow_{\mathsf{wh}} t'}{t \longrightarrow_{\mathsf{l}} t'} \qquad \text{L-XI}\ \frac{t \longrightarrow_{\mathsf{l}} t'}{\lambda x t \longrightarrow_{\mathsf{l}} \lambda x t'} \qquad \text{L-RIGHT}\ \frac{\vec{s} \in \mathsf{NF} \qquad s \longrightarrow_{\mathsf{l}} s'}{x\,\vec{s}\,s\,\vec{t} \longrightarrow_{\mathsf{l}} x\,\vec{s}\,s'\,\vec{t}}$$

Lemma 3.59 (Kompatibilitätsregeln für $\longrightarrow_{\mathsf{l}}^*$)

1. *Wenn $t \longrightarrow_{\mathsf{l}}^* t'$ then $\lambda x t \longrightarrow_{\mathsf{l}}^* \lambda x t'$.*
2. *Wenn $x\,\vec{s} \in \mathsf{NF}$ und $s \longrightarrow_{\mathsf{l}}^* s'$, dann $x\,\vec{s}\,s \longrightarrow_{\mathsf{l}}^* x\,\vec{s}\,s'$.*
3. *Wenn $\vec{s} \longrightarrow_{\mathsf{l}}^* \vec{v} \in \mathsf{NF}$, dann $x\,\vec{s} \longrightarrow_{\mathsf{l}}^* x\,\vec{v} \in \mathsf{NF}$.*

Satz 3.60 *Wenn $t \in \mathsf{WN}$, dann $t \longrightarrow_{\mathsf{l}}^* v \in \mathsf{NF}$.*

Beweis. Durch Induktion über $t \in \mathsf{WN}$, unter Verwendung des vorherigen Lemmas. □

Kapitel 4

Eta-Reduktion

Wenn $x \notin \mathsf{FV}(t)$, dann $(\lambda x.\, t\, x)\, s \longrightarrow_\beta t\, s$. Dies motiviert die *Programmoptimierung* $\lambda x.\, t\, x \longrightarrow_\eta t$.

Definition 4.1 (η-Reduktion) *Die Ein-Schritt-η-Reduktion $\longrightarrow_\eta$ ist die kongruente Hülle des Axiomenschemas $\lambda x.\, t\, x \longrightarrow_\eta t$ ($x \notin \mathsf{FV}(t)$). Die $\beta\eta$-Reduktion ist definiert durch $\longrightarrow_{\beta\eta} := \longrightarrow_\beta \cup \longrightarrow_\eta$. η-Gleichheit und $\beta\eta$-Gleichheit sind wie üblich definiert, als kleinste Äquivalenzrelation über $\longrightarrow_\eta$ bzw. $\longrightarrow_{\beta\eta}$.*

Definition 4.2 (Extensionalität) *Sei M eine Menge mit einer Applikationsoperation, geschrieben als Juxtaposition: $r, s \in M$ implziert $r\, s \in M$. Eine Relation R auf M heisst* extensional *falls $r\ R\ r'$ gdw. $r\, s\ R\ r'\, s$ für alle $s \in M$.*

Bemerkung 4.3 *Die Gleichheit in der Mengenlehre ist extensional. In der (Schul-)Mathematik verwenden wir oft (implizit) Extensionalität, denn um zu zeigen, dass zwei Funktionen übereinstimmen, zeigen wir gewöhnlicherweise die punktweise Übereinstimmung. Die β-Gleichheit ist jedoch nicht extensional, da $\lambda x.\, t\, x \neq t$.*

Satz 4.4 (Curry) *Folgende Gleichheitsrelationen sind äquivalent:*

1. *$=_\beta$ + Extensionalität,*
2. *$=_\beta$ + Axiomenschema $t\, x =_\beta t'\, x$ $(x \notin \mathsf{FV}(t, t')) \implies t =_\beta t'$.*
3. *$=_{\beta\eta}$.*

Beweis. Es gilt:

$$
\begin{array}{lll}
 & t\, s =_\beta t'\, s & \text{für alle } s \in \Lambda \\
\iff & t\, x =_\beta t'\, x & \text{für ein } x \notin \mathsf{FV}(t, t') \\
\iff & \lambda x.\, t\, x =_\beta \lambda x.\, t'\, x & \text{für ein } x \notin \mathsf{FV}(t, t') \\
\iff & t =_{\beta\eta} t' &
\end{array}
$$

□

4.1 Eigenschaften der η-Reduktion

β- und η-Reduktion überlappen: Wenn $x \notin \mathsf{FV}(t)$ dann

$$\begin{array}{lll} (\lambda x.\, t\, x)\, s & \longrightarrow_{\beta\cap\eta} & t\, s \\ \lambda x.\, (\lambda y t)\, x & \longrightarrow_{\beta\cap\eta} & \lambda y t \equiv \lambda x.\, t[x/y] \end{array}$$

Lemma 4.5 (Erhaltung freier Variablen) *$t \longrightarrow_\eta t'$ impliziert $\mathsf{FV}(t) = \mathsf{FV}(t')$.*

Lemma 4.6 *η-Reduktion ist substitutiv.*

Beweis. Sei $x \notin \mathsf{FV}(t)$. Da die Substitution $(\lambda x.\, t\, x)[s/y] \equiv \lambda x.\, t[s/y]\, x$ keine Variablen gefangennimmt, ist auch $x \notin \mathsf{FV}(t[s/y])$. □

Lemma 4.7 *$\longrightarrow_\eta^=$ hat die Diamanteigenschaft.*

Satz 4.8 *η-Reduktion ist konfluent.*

Definition 4.9 (Starke Normalisierung, klassisch) *Wir sagen, eine Relation $\longrightarrow$ ist stark normalisierend, wenn es keine unendlichen Reduktionsfolgen $t_0 \longrightarrow t_1 \longrightarrow \ldots$ gibt.*

Die β-Reduktion ist nicht stark normalisierend wegen $\Omega \longrightarrow_\beta \Omega$.

Satz 4.10 *η-Reduktion ist stark normalisierend.*

Beweis. Jeder Reduktionsschritt entfernt ein λ. Davon gibt es aber nur endlich viele. □

Satz 4.11 (η-Reduktion erhält β-Normalformen) *Wenn $t \in \mathsf{NF}_\beta$ und $t \longrightarrow_\eta t'$, dann $t' \in \mathsf{NF}_\beta$.*

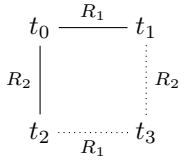
Beweis. Durch Induktion über $t \in \mathsf{NF}_\beta$. Der prinzipielle Fall ist $t \equiv \lambda x.\, t'\, x \longrightarrow_\eta t'$. Es gilt aber $t \in \mathsf{NF}_\beta \iff t'\, x \in \mathsf{NF}_\beta \implies t' \in \mathsf{NF}_\beta$. □

4.2 Konfluenz der $\beta\eta$-Reduktion

Konfluenz der $\beta\eta$-Reduktion kann z.B. durch eine Adaption der parallelen Reduktion gezeigt werden Selinger (2006). Wir verwenden hier eine andere Technik (Barendregt, 1984, S.64ff).

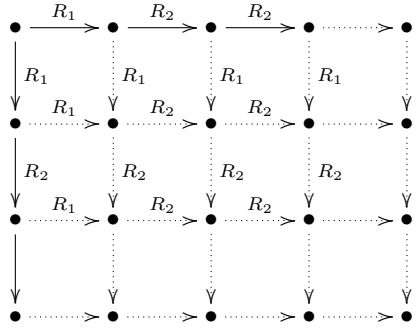
Wir scheiben $R\,\diamond$ wenn R die Diamanteigenschaft hat.

Definition 4.12 *Zwei Relationen R_1, R_2 über einer Menge M* kommuntieren, *in Zeichen $R_1 \square R_2$, falls es für alle $t_0, t_1, t_2 \in M$ mit $t_2\ R_2^{-1}\ t_0\ R_1\ t_1$ ein $t_3 \in M$ gibt mit $t_2\ R_1\ t_3\ R_2^{-1}\ t_1$.*



Lemma 4.13 (Hindley und Rosen) *Falls $R_1 \diamond$, $R_2 \diamond$ und $R_1 \square R_2$, dann $(R_1 \cup R_2)^* \diamond$.*

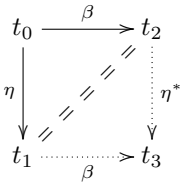
Beweis.



□

Bemerkung 4.14 *Falls $R_1 \diamond$, $R_2 \diamond$ und $R_1 \square R_2$, dann $(R_1 \cup R_2) \diamond$. Daraus folgt dann mit Lemma* **??** *$(R_1 \cup R_2)^* \diamond$.*

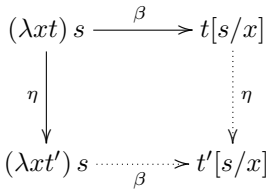
Lemma 4.15 *Wenn $t_0 \longrightarrow_\eta t_1$ und $t_0 \longrightarrow_\beta t_2$, dann ist entweder $t_1 \equiv t_2$ oder es gibt ein t_3 mit $t_1 \longrightarrow_\beta t_3$ und $t_2 \longrightarrow_\eta^* t_3$.*



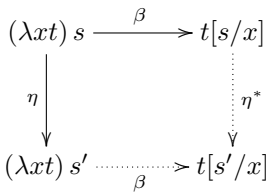
Beweis. Durch simultane Induktion über $t_0 \longrightarrow_\eta t_1$ und $t_0 \longrightarrow_\beta t_2$. Wir unterscheiden folgende Fälle:

1. Beide Reduktionen in Teiltermen: klar, mit Hilfe der I.V.
2. Reduktionen überlappend: $t_0 \equiv (\lambda x.\, t\, x)\, s$ oder $t_0 \equiv \lambda x.\, (\lambda y t)\, x$ mit $x \notin \mathsf{FV}(t)$. Dann $t_1 \equiv t_2$.

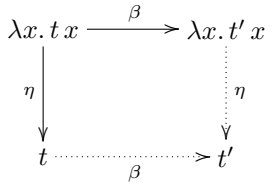
3.



4.



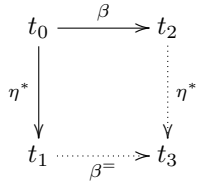
5. $x \notin \mathsf{FV}(t)$ und



Hier verwenden wir $t \longrightarrow_\beta t' \implies \mathsf{FV}(t) \supseteq \mathsf{FV}(t')$.

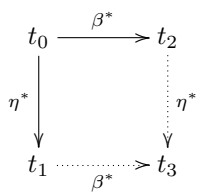
□

Lemma 4.16 $\longrightarrow_\eta^* \square \longrightarrow_\beta^=$.



Beweis. Durch Induktion über $t_0 \longrightarrow_\eta^* t_1$. □

Satz 4.17 (η-Reduktion kommuntiert mit β-Reduktion) $\longrightarrow_\eta^* \square \longrightarrow_\beta^*$.



Beweis. Durch Induktion über $t_0 \longrightarrow_\beta^* t_2$. □

Satz 4.18 ($\beta\eta$-Reduktion ist konfluent) $\longrightarrow_{\beta\eta}^* \diamond$.

Beweis. Da $\longrightarrow_\beta \diamond$, $\longrightarrow_\eta \diamond$ und $\longrightarrow_\beta^* \square \longrightarrow_\eta^*$, gilt nach dem Lemma von Hindley und Rosen $(\longrightarrow_\beta^* \cup \longrightarrow_\eta^*)^* \diamond$, also $\longrightarrow_{\beta\eta}^* \diamond$. □

Übung 4.19 (Alternativer Konfluenzbeweis) *Die vollständige η-Superentwicklung t^η, die in einen Term t alle η-Redexe entfernt, ist rekursiv definiert wie folgt:*

$$\begin{array}{lcl} x^\eta & \equiv & x \\ (r\,s)^\eta & \equiv & r^\eta\, s^\eta \\ (\lambda x t)^\eta & \equiv & \begin{cases} t' & \text{falls } t^\eta \equiv t'\,x \text{ und } x \notin \mathsf{FV}(t') \\ \lambda x.\, t^\eta & \text{sonst} \end{cases} \end{array}$$

Zeigen Sie:

1. $t \longrightarrow_\eta^* t^\eta \not\longrightarrow_\eta$.
2. *Wenn* $t_1 \longrightarrow_\eta t_2$, *dann* $t_1^\eta \equiv t_2^\eta$.
3. $(t[s/x])^\eta \equiv t^\eta[s^\eta/x]$.
4. *Wenn* $t_1 \longrightarrow_\beta t_2$, *dann* $t_1^\eta \longrightarrow_\beta^= t_2^\eta$.
5. *Wenn* $t_1 \longrightarrow_{\beta\eta}^* t_2$, *dann* $t_1 \longrightarrow_\beta^* t_2^\eta$.
6. $\longrightarrow_{\beta\eta}$ *ist konfluent.*

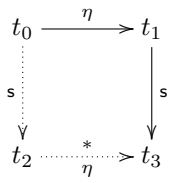
4.3 η-Aufschiebung (η-postponement)

Bemerkung 4.20 *Die Aussage* $\longrightarrow_\eta \longrightarrow_\beta \subseteq \longrightarrow_\beta \longrightarrow_\eta^*$ *scheitert am Fall* $(\lambda y.(\lambda x t)\, y)\, s \longrightarrow_\eta (\lambda x t)\, s \longrightarrow_\beta t[s/x]$. *Hier sind zwei* β*-Reduktionen nötig.*

Takahashi Takahashi (1995) zeigt $\Longrightarrow_\eta \Longrightarrow_\beta \subseteq \Longrightarrow_\beta \Longrightarrow_\eta$ für parallele β- und η-Reduktion. Mit $t \Longrightarrow_\eta v \in \mathsf{nf}_\beta$ impliziert $t \in \mathsf{wn}_\beta$ folgt daraus, dass die schwach β-normalisierenden Terme unter η-Expansion abgeschlossen sind, also das wichtige Resultat $\mathsf{wn}_{\beta\eta} \subseteq \mathsf{wn}_\beta$. Hier das Argument (hat Simon Huber mir nahegebracht): Sei $t \Longrightarrow_\eta t' \in \mathsf{wn}_\beta$, also $t' \Longrightarrow_\beta^* v \in \mathsf{nf}_\beta$. Dann gilt mit η-Aufschiebung, $t \Longrightarrow_\beta^* t'' \Longrightarrow_\eta v$, also $t'' \in \mathsf{wn}_\beta$ und damit auch $t \in \mathsf{wn}_\beta$. Hierfür schein essentiell, dass die eine parallele η-Reduktion nach der Aufschiebung immer noch *eine* ist.

Wir zeigen eine stärkere Aussage als Takahashi:

Satz 4.21 (Abschluss der Standardreduktion unter η-Expansion) *Wenn* $t_0 \longrightarrow_\eta t_1 \longrightarrow_\mathsf{s} t_3$, *dann gibt es ein* t_2 *mit* $t_0 \longrightarrow_\mathsf{s} t_2 \longrightarrow_\eta^* t_3$.



Wenn $t_3 \in \mathsf{nf}_\beta$, *dann auch* $t_2 \in \mathsf{nf}_\beta$.

Der direkte Beweis des Satzes scheitert am Fall $(\lambda x t)\, s \longrightarrow_\eta (\lambda x t)\, s' \longrightarrow_\mathsf{wh} t[s'/x] \longrightarrow_\mathsf{s} r$: Dann haben wir nur $(\lambda x t)\, s \longrightarrow_\eta t[s/x] \longrightarrow_\eta^* t[s'/x] \longrightarrow_\mathsf{s} r$, darauf ist die I.V. aber nicht anwendbar.

Wir verstärken die Aussage auf parallele η-Reduktion, die wir induktiv wie folgt definieren:

$$\frac{}{x \Longrightarrow_\eta x} \qquad \frac{t \Longrightarrow_\eta t'}{\lambda x t \Longrightarrow_\eta \lambda x t'} \qquad \frac{r \Longrightarrow_\eta r' \qquad s \Longrightarrow_\eta s'}{r\, s \Longrightarrow_\eta r'\, s'} \qquad \frac{t \Longrightarrow_\eta t'}{\lambda x.\, t\, x \Longrightarrow_\eta t'}\ x \notin \mathsf{FV}(t)$$

Es gilt $\longrightarrow_\eta \subseteq \Longrightarrow_\eta \subseteq \longrightarrow_\eta^*$. Ausserdem ist $\Longrightarrow_\eta$ unter Substitution abgeschlossen:

Lemma 4.22 *Wenn* $t \Longrightarrow_\eta t'$ *und* $s \Longrightarrow_\eta s'$, *dann* $t[s/x] \Longrightarrow_\eta t'[s'/x]$.

Beweis. Durch Induktion über $t \Longrightarrow_\eta t'$. □

Wenn $r \Longrightarrow_\eta \lambda x t'$, dann ist r eine mehrfache η-Expansion eines Termes $\lambda x t$ mit $t \Longrightarrow_\eta t'$ Takahashi (1995). Uns genügt es die folgende Konsequenz dieser Tatsache zu zeigen:

Lemma 4.23 *Wenn* $r \Longrightarrow_\eta \lambda x t'$, *dann* $r\, s \longrightarrow_\mathsf{wh}^+ t[s/x]$ *für ein* $t \Longrightarrow_\eta t'$.

Beweis. Durch Induktion über $r \Longrightarrow_\eta \lambda x t'$. □

Dies verallgemeinern wir noch auf:

Lemma 4.24 *Wenn* $r \Longrightarrow_\eta (\lambda x t')\, s'_1 \dots s'_n$, *dann* $r\, s_{n+1} \longrightarrow_{\mathsf{wh}}^+ t[s_1/x]\, s_2 \dots s_{n+1}$ *für ein* $t \Longrightarrow_\eta t'$ *und Terme* $\vec{s} \Longrightarrow_\eta \vec{s}'$.

Beweis. Durch Induktion über $r \Longrightarrow_\eta (\lambda x t')\, \vec{s}'$.

Fall $n = 0$, $r \equiv \lambda x t$ und $t \Longrightarrow_\eta t'$. Dann $r\, s_{n+1} \equiv r\, s_1 \longrightarrow_{\mathsf{wh}} t[s_1/x]$.

Fall $n > 0$ und

$$\frac{r \Longrightarrow_\eta (\lambda x t')\, s'_1 \dots s'_{n-1} \qquad s_n \Longrightarrow_\eta s'_n}{r\, s_n \Longrightarrow_\eta (\lambda x t')\, s'_1 \dots s'_n}$$

Nach I.V., $r\, s_n \longrightarrow_{\mathsf{wh}}^+ t[s_1/x]\, s_2 \dots s_n$ mit $t \Longrightarrow_\eta t'$ und $s_1, \dots, s_{n-1} \Longrightarrow_\eta s'_1, \dots, s'_{n-1}$. Damit $r\, s_n\, s_{n+1} \longrightarrow_{\mathsf{wh}}^+ t[s_1/x]\, s_2 \dots s_{n+1}$ und $\vec{s} \Longrightarrow_\eta \vec{s}'$.

Fall

$$\frac{r \Longrightarrow_\eta (\lambda x t')\, \vec{s}'}{\lambda y.\, r\, y \Longrightarrow_\eta (\lambda x t')\, \vec{s}'}\ y \notin \mathsf{FV}(r)$$

$(\lambda y.\, r\, y)\, s_{n+1} \longrightarrow_{\mathsf{wh}} r\, s_{n+1} \longrightarrow_{\mathsf{wh}}^+ t[s_1/x]\, s_2 \dots s_{n+1}$ und $t, \vec{s} \Longrightarrow_\eta t', \vec{s}'$ nach I.H.

□

Lemma 4.25 *Wenn* $r \Longrightarrow_\eta t \neq \lambda x t'$, *dann* $r\, s \longrightarrow_{\mathsf{wh}}^+ r'\, s$ *mit* $r' \Longrightarrow_\eta t$ *und* r' *ist keine Abstraktion.*

Beweis. Durch Induktion über $r \Longrightarrow_\eta t$.

Fall

$$\frac{r \Longrightarrow_\eta t}{\lambda x.\, r\, x \Longrightarrow_\eta t}$$

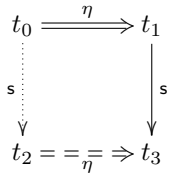
Es gilt $(\lambda x.\, r\, x)\, s \longrightarrow_{\mathsf{wh}} r\, s \longrightarrow_{\mathsf{wh}}^+ r'\, s$ mit r' wie gewünscht nach I.V.

Dies ist die einzige Möglichkeit für r Abstraktion. Falls r keine Abstraktion, sind wir sofort fertig (wähle $r' = r$). □

Lemma 4.26 *Wenn* $t \longrightarrow_{\mathsf{s}} \lambda x t'$, *dann* $t\, y \longrightarrow_{\mathsf{s}} t'[y/x]$.

Beweis. Trivial, durch Induktion über $t \longrightarrow_{\mathsf{s}} \lambda x t'$. □

Lemma 4.27 (Aufschiebung einer parallelen η-Reduktion) *Wenn $t_0 \Longrightarrow_\eta t_1 \longrightarrow_{\mathsf{s}} t_3$, dann gibt es ein t_2 mit $t_0 \longrightarrow_{\mathsf{s}} t_2 \Longrightarrow_\eta t_3$.*



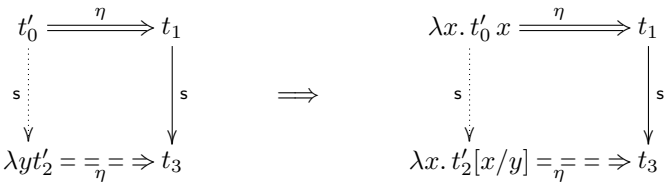
Wenn $t_3 \in \mathsf{nf}_\beta$, dann auch $t_2 \in \mathsf{nf}_\beta$.

Beweis. Durch lexikographische Induktion über $(t_1 \longrightarrow_{\mathsf{s}} t_3, t_0 \Longrightarrow_\eta t_1)$. Wir behandeln zuerst η-Kontraktionen an der Wurzel von t_0.

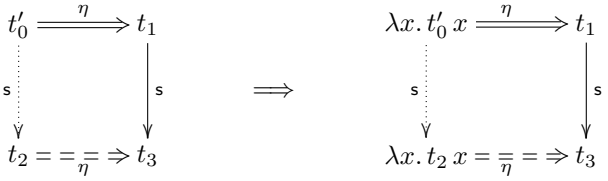
$$\frac{t_0' \Longrightarrow_\eta t_1}{\lambda x.\, t_0'\, x \Longrightarrow_\eta t_1}\ x \notin \mathsf{FV}(t_0')$$

Nach I.V. $t_0' \longrightarrow_{\mathsf{s}} t_2 \Longrightarrow_\eta t_3$. Wir unterscheiden zwei Fälle:

Unterfall $t_2 \equiv \lambda y t_2'$. Nach Lemma 4.26 gilt $t_0'\, x \longrightarrow_{\mathsf{s}} t_2'[x/y]$, also $t_0 \equiv \lambda x.\, t_0'\, x \longrightarrow_{\mathsf{s}} \lambda x.\, t_2'[x/y] \equiv t_2 \Longrightarrow_\eta t_3$. Wenn $t_3 \in \mathsf{nf}_\beta$, dann nach I.V. auch $t_2 \in \mathsf{nf}_\beta$.



Unterfall t_2 ist keine λ-Abstraktion. Dann $t_0 \equiv \lambda x.\, t_0'\, x \longrightarrow_{\mathsf{s}} \lambda x.\, t_2\, x \Longrightarrow_\eta t_3$. Wenn $t_3 \in \mathsf{nf}_\beta$, dann nach I.V. auch t_2, und da t_2 keine Abstraktion ist, $\lambda x.\, t_2\, x \in \mathsf{nf}_\beta$.



Es verbleiben nur noch Fälle, in denen t_0 selbst kein η-Redex ist. Wir betrachten nun die möglichen Herleitungen von $t_1 \longrightarrow_{\mathsf{s}} t_3$.

Fall $t_0 \equiv r\, s_n$ und $t_1 \equiv (\lambda x t')\, s\, \vec{s}$ mit $n = |\vec{s}|$, und

$$\frac{r \Longrightarrow_\eta (\lambda x t')\, s'\, s_1' \ldots s_{n-1}' \qquad s_n \Longrightarrow_\eta s_n'}{r\, s_n \Longrightarrow_\eta (\lambda x t')\, s'\, \vec{s}'} \qquad \text{und} \qquad \frac{t'[s'/x]\vec{s}' \longrightarrow_{\mathsf{s}} t_3}{(\lambda x t')\, s'\, \vec{s}' \longrightarrow_{\mathsf{s}} t_3}$$

Nach Lemma 4.24 $t_0 \equiv r\, s_n \longrightarrow_{\mathsf{wh}}^+ t[s/x]\, \vec{s}$ für Terme $t, s, \vec{s} \Longrightarrow_\eta t', s', \vec{s}'$. Mit dem Substitutionslemma für parallele η-Reduktion gilt $t_1' :\equiv t[s/x]\, \vec{s} \Longrightarrow_\eta$

$t'[s'/x]\,\vec{s}' \longrightarrow_{\mathsf{s}} t_3$, worauf wir die Induktionshypothese anwenden können. Wir erhalten t_2 mit $t_1' \longrightarrow_{\mathsf{s}} t_2 \Longrightarrow_\eta t_3$, und wegen $t_0 \longrightarrow_{\mathsf{wh}}^+ t_1'$ schliesslich $t_0 \longrightarrow_{\mathsf{s}} t_2 \Longrightarrow_\eta t_3$. Wenn $t_3 \in \mathsf{nf}_\beta$, dann ist nach I.V. auch $t_2 \in \mathsf{nf}_\beta$.

Fall

$$\frac{r_0 \Longrightarrow_\eta r_1 \longrightarrow_{\mathsf{s}} r_3 \qquad s_0 \Longrightarrow_\eta s_1 \longrightarrow_{\mathsf{s}} s_3}{r_0\, s_0 \Longrightarrow_\eta r_1\, s_2 \longrightarrow_{\mathsf{s}} r_3\, s_3}$$

Nach I.V., $r_0 \longrightarrow_{\mathsf{s}} r_2 \Longrightarrow_\eta r_3$ und $s_0 \longrightarrow_{\mathsf{s}} s_2 \Longrightarrow_\eta s_3$, also $r_0\, s_0 \longrightarrow_{\mathsf{s}} r_2\, s_2 \Longrightarrow_\eta r_3\, s_3$. Falls $r_3\, s_3 \in \mathsf{nf}_\beta$, dann gilt nach I.V, $r_2, s_2 \in \mathsf{nf}_\beta$. Ausserdem ist r_3 keine Abstraktion, also gibt es nach Lemma 4.25 ein $r_2' \Longrightarrow_\eta r_3$ mit $r_2\, s_2 \longrightarrow_{\mathsf{wh}}^+ r_2'\, s_2$, mit Standardisierung somit $r_0\, s_0 \longrightarrow_{\mathsf{s}} r_2'\, s_2$. Da r_2' keine Abstraktion ist, aber β-normal, ist $r_2'\, s_2 \in \mathsf{nf}_\beta$. Insgesamt $r_0\, s_0 \longrightarrow_{\mathsf{s}} r_2'\, s_2 \Longrightarrow_\eta r_3\, s_3$.

Fall

$$\frac{t_0' \Longrightarrow_\eta t_1' \longrightarrow_{\mathsf{s}} t_3'}{\lambda x t_0' \Longrightarrow_\eta \lambda x t_1' \longrightarrow_{\mathsf{s}} \lambda x t_3'}$$

Klar nach I.V.

Der verbleibende Fall $x \Longrightarrow_\eta x \longrightarrow_{\mathsf{s}} x$ ist banal. □

Nun folgt direkt Satz 4.21. Außerdem:

Korollar 4.28 WN_β *ist unter η-Expansion abgeschlossen.*

Satz 4.29 (η-Aufschiebung)

1. $(\longrightarrow_\beta \cup \longrightarrow_\eta)^* \subseteq \longrightarrow_\beta^* \longrightarrow_\eta^*$.
2. $\mathsf{wn}_{\beta\eta} \subseteq \mathsf{wn}_\beta$.

Beweis. 1. Aus $t_1 \longrightarrow_{\beta\eta}^* t_3 \longrightarrow_{\mathsf{s}} t_3 \longrightarrow_\eta^* t_3$ zeigen wir $t_1 \longrightarrow_{\mathsf{s}} t_2 \longrightarrow_\eta^* t_3$ durch Induktion über $t_1 \longrightarrow_{\beta\eta}^* t_3$. 2. "$\subseteq$" folgt aus der starken Normalisierung der η-Reduktion, für "$\supseteq$" betrachte $t \longrightarrow_{\beta\eta}^* v \in \mathsf{nf}_{\beta\eta} \subseteq \mathsf{wn}_\beta$. Durch Induktion über $t \longrightarrow_{\beta\eta}^* v$ folgt mit dem Korollar $t \in \mathsf{wn}_\beta$. □

4.4 Übungen

Übung 4.30 *Betrachten Sie die Erweiterung des λ-Kalküls um drei Konstanten* Pair, fst *und* snd *und die zusätzlichen β-Axiome*

$$\mathsf{fst}\,(\mathsf{Pair}\, r\, s) \longrightarrow_\beta r \qquad \mathsf{snd}\,(\mathsf{Pair}\, r\, s) \longrightarrow_\beta s.$$

Zeigen Sie, dass η-Aufschiebung nun nicht mehr gilt. [Hinweis: "Schuld" daran sind "sinnlose" Terme wie $\mathsf{fst}\,(\lambda x t)$*.]*

Übung 4.31 *Betrachten Sie die Erweiterung des λ-Kalküls um zwei Konstanten* L *und* R*, Paare* $\langle r, s\rangle$ *und die zusätzlichen β-Axiome*

$$\langle r, s\rangle\,\mathsf{L} \longrightarrow_\beta r \qquad \langle r, s\rangle\,\mathsf{R} \longrightarrow_\beta s$$

1. *Zeigen Sie, dass η-Aufschiebung gilt.*
2. *Zeigen Sie, dass η-Aufschiebung nicht mehr gilt, wenn wir das Axiom* $\langle t\,\mathsf{L},\ t\,\mathsf{R}\rangle \longrightarrow_\eta t$ *hinzunehmen.*

Kapitel 5

Starke Normalisierung

Definition 5.1 (Maximale Reduktionslänge) *Sei M eine Menge und $\longrightarrow$ eine nicht-reflexive (Reduktions-)Relation auf M.*

$$\ell(t) := \sup\{n \in \mathbb{N} \mid t \longrightarrow^n t' \text{ for some } t'\} \in \mathbb{N} \cup \{\infty\}$$

heißt die maximale Reduktionslänge *von t.*

Im Falle von $\longrightarrow_\beta$ schreiben wir auch ℓ_β.

Beispiel 5.2 *Falls t normal, dann ist $\ell(t) = 0$. Ein Objekt t, dass zu sich selbst reduziert (wie Ω im Lambda-Kalkül), hat maximale Reduktionslänge ∞, denn für alle $n \in \mathbb{N}$ gibt es die Reduktionsfolge $t \longrightarrow^n t$, und* $\sup \mathbb{N} = \infty$.

Ein Term t heisst *stark normalisierend*, wenn es keine unendliche Reduktionsfolge gibt, die mit t beginnt. Positiv formuliert ist jede Reduktionsfolge beginnend mit t endlich. Dies findet in folgender induktiver Definition Ausdruck:

Definition 5.3 (Starke Normalisierung) *Die Menge $\mathsf{sn}_{\longleftarrow} \subseteq M$ ist die kleinste Menge, die unter der Regel*

$$\frac{t' \in \mathsf{sn}_{\longleftarrow} \text{ für alle } t' \text{ mit } t \longrightarrow t'}{t \in \mathsf{sn}_{\longleftarrow}}.$$

abgeschlossen ist.

Wenn klar ist, welche Relation $\longrightarrow$ gemeint ist, schreiben wir nur sn. Wir schreiben sn_β im Falle von $\longrightarrow_\beta$ und $\mathsf{sn}_{\beta\eta}$ im Falle von $\longrightarrow_{\beta\eta}$.

Eine Normalform t ist stark normalisierend (sn), weil es kein t' gibt mit $t \longrightarrow t'$, also ist die Prämisse der Regel trivial erfüllt.

Satz 5.4 *Sei $\longrightarrow$ endlich verzweigend, d.h. $\{t' \mid t \longrightarrow t'\}$ endlich für jedes t. Wenn $t \in \mathsf{sn}$, dann ist $\ell(t) = \max\{1 + \ell(t)' \mid t \longrightarrow t'\}$ endlich (also $\in \mathbb{N}$).*

Beweis. Durch Induktion über $t \in \mathsf{sn}$. Wenn $t \not\longrightarrow$ dann ist $\ell(t) = 0 = \max \emptyset$. Andernfalls seien $t_1, \ldots, t_m$ mit $m > 0$ die Ein-Schritt-Redukte von t.

$$\begin{array}{rcl} \ell(t) & = & \sup\{1 + n \mid t \longrightarrow t_i \longrightarrow^n t' \text{ for some } i, t'\} \\ & = & 1 + \sup_i \sup\{n \mid t_i \longrightarrow t' \text{ for some } t'\} \\ & = & 1 + \sup_i \ell(t_i) \\ & = & \max\{1 + \ell(t_i) \mid t \longrightarrow t_i\} \end{array}$$

Der letzte Schritt benutzt die Endlichkeit von $\ell(t_i)$ nach I.V. □

Beispiel 5.5 *Sei $t \longrightarrow t$. Wir zeigen $t \notin \mathsf{sn}$, was gleichbedeutend ist mit $t \in \mathsf{sn} \implies \bot$, durch Induktion nach $t \in \mathsf{sn}$. Es gibt nur den Schrittfall $t \in \mathsf{sn}$, in dem wir $\bot$ zeigen müssen. Nach I.V. ist gilt $\bot$ für alle $t \longrightarrow t'$. Da aber $t \longrightarrow t$, gilt nach I.V. $\bot$.*

Derselbe Beweis in gebräuchlicher mathematischer Sprache:

Beispiel 5.6 *Sei $t \longrightarrow t$. Dann ist $t \notin \mathsf{sn}$. Wir zeigen einen Widerspruch durch Induktion nach $t \in \mathsf{sn}$. Es gibt nur den Schrittfall, in dem wir zeigen müssen, dass $t \in \mathsf{sn}$ widersprüchlich. Nach I.V. ist $t' \in \mathsf{sn}$ widersprüchlich für alle $t \longrightarrow t'$. Da aber $t \longrightarrow t$, ist $t \in \mathsf{sn}$ widersprüchlich.*

5.1 Wohlfundiertheit und Noethersche Induktion

Definition 5.7 (Erreichbarer Teil) *Sei M eine Menge und R eine Relation auf M. Der erreichbare Teil $\mathsf{Acc}_R \subseteq M$ ist induktiv definiert durch:*

$$\frac{t' \in \mathsf{Acc}_R \text{ für alle } t' \neq t \text{ mit } t' \mathrel{R} t}{t \in \mathsf{Acc}_R}$$

Eine Relation R heisst wohlfundiert *oder* noethersch*, falls $\mathsf{Acc}_R = M$.*

Daraus ergibt sich folgendes Induktionsprinzip: um eine Aussage $A(t)$ für $t \in \mathsf{Acc}_<$ zu zeigen, darf man jeweis $A(t')$ für alle $t' < t$ voraussetzen.

Beispiel 5.8 *Wohlfundierte Relationen auf $\mathbb{N}$ sind $<$ (Ordnungstyp ω) und die Teilbarkeitsrelation (Ordnungstyp $\omega + 1$, z.B. Kette $2^0 \mid 2^1 \mid 2^2 \mid \cdots \mid 0$). Sind zwei Relationen R_1 und R_2 wohlfundiert, so auch ihr lexikographisches Produkt $R_1 \otimes R_2$.*

Lemma 5.9 (Kontravarianz) *Wenn $R \subseteq R'$, dann $\mathsf{Acc}_{R'} \subseteq \mathsf{Acc}_R$.*

Beweis. Wir zeigen $t \in \mathsf{Acc}_R$ durch Induktion über $t \in \mathsf{Acc}_{R'}$. Sei $t' \mathrel{R} t$ beliebig, zu zeigen ist $t' \in \mathsf{Acc}_R$. Dies gilt aber nach I.V., da $t' \mathrel{R'} t$. □

Lemma 5.10 (Abstieg) *Wenn $t \in \mathsf{Acc}_R$ und $t'\ R^*\ t$, dann $t' \in \mathsf{Acc}_R$.*

Beweis. Durch Induktion über $t'\ R^*\ t$. □

Satz 5.11 (Erreichbarer Teil invariant unter Transitivität) $\mathsf{Acc}_{R^+} = \mathsf{Acc}_R$.

Beweis. Richtung "$\subseteq$" folgt aus Kontravarianz. Für die Rückrichtung zeigen wir $t \in \mathsf{Acc}_{R^+}$ durch Induktion über $t \in \mathsf{Acc}_R$. Seien t'', t' beliebig mit $t''\ R^*\ t'\ R\ t$; zu zeigen ist $t'' \in \mathsf{Acc}_{R^+}$. Nach I.V. ist $t' \in \mathsf{Acc}_{R^+}$ also nach dem Abstiegslemma auch $t'' \in \mathsf{Acc}_{R^+}$. □
Damit können wir das Induktionsprinzip verbessern auf: um eine Aussage $A(t)$ für $t \in \mathsf{Acc}_R$ zu zeigen, darf man jeweis $A(t')$ für alle $t'\ R^+\ t$ voraussetzen. Dies ist die *noethersche* oder *wohlfundierte* Induktion.

5.2 Allgemeine Eigenschaften starker Normalisierung

Lemma 5.12 (Newman) *Sei $\longrightarrow$ eine lokal konfluente Relation. Wenn $t_0 \in \mathsf{sn}$ und $t_0 \longrightarrow^* t_1$ und $t_0 \longrightarrow^* t_2$, dann gibt es ein t_3, so dass $t_1 \longrightarrow^* t_3$ und $t_2 \longrightarrow^* t_3$.*

Beweis. Durch noethersche Induktion über $t_0 \in \mathsf{sn}$, und Fallunterscheidung über die Reduktionsfolgen $t_0 \longrightarrow^* t_1$ und $t_0 \longrightarrow^* t_2$.

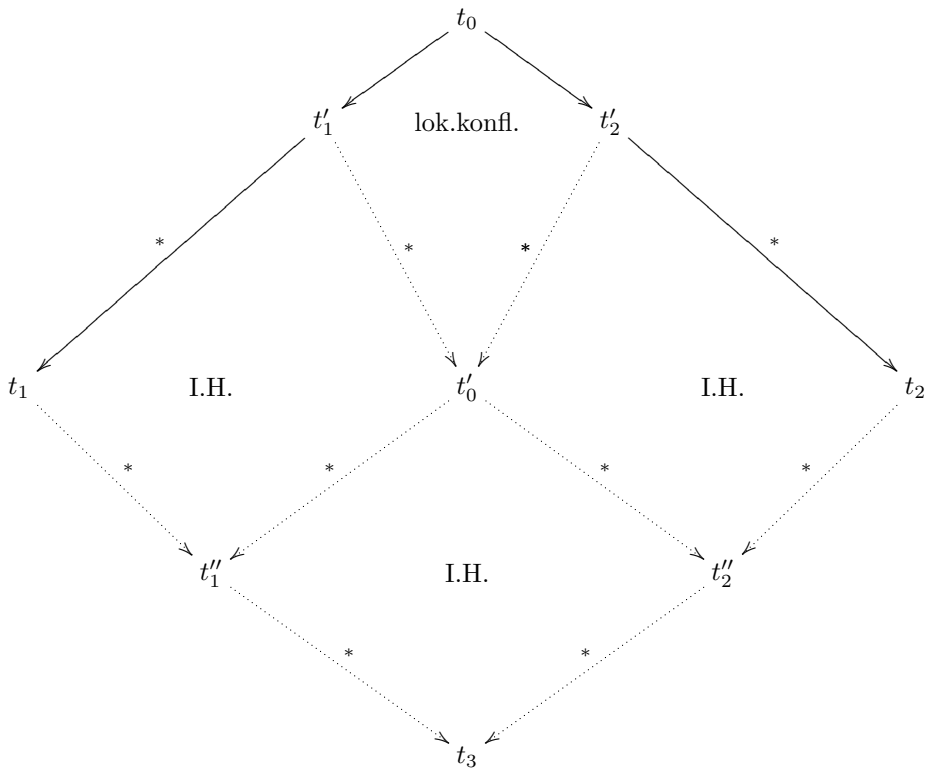
Fall $t_0 \equiv t_1 \equiv t_2$. Dann $t_3 := t_0$.

Fall $t_0 \equiv t_1$ und $t_0 \longrightarrow^+ t_2$. Dann $t_3 := t_2$.

Fall $t_0 \longrightarrow^+ t_1$ und $t_0 \equiv t_2$. Dann $t_3 := t_1$.

Fall $t_0 \longrightarrow t_1' \longrightarrow^* t_1$ und $t_0 \longrightarrow t_2' \longrightarrow^* t_2$. Dann gibt es aufgrund der lokalen Konfluenz von $\longrightarrow$ einen Term t_0' mit $t_1' \longrightarrow^* t_0'$ und $t_2' \longrightarrow^* t_0'$. Nach Induktionsvoraussetzung für t_1' gibt es einen Term t_1'' so dass $t_1 \longrightarrow^* t_1''$ und $t_0' \longrightarrow^* t_1''$, ebenso gibt es einen Term t_2'' so dass $t_0' \longrightarrow^* t_2''$ und $t_2 \longrightarrow^* t_2''$. Schließlich können wir die Induktionsvoraussetzung auf t_0' anwenden

und erhalten den gesuchten Term t_3.



□

Lemma 5.13 (Subterme von sn-Termen) *Sei* $\longrightarrow$ *eine kongruente (nicht-reflexive) Reduktions-Relation auf* Λ *und* sn *die Menge der sn Terme bzgl. dieser Relation.*

1. *Wenn* $r\,s \in \mathsf{sn}$*, dann* $r \in \mathsf{sn}$*.*
2. *Wenn* $r\,s \in \mathsf{sn}$*, dann* $s \in \mathsf{sn}$*.*
3. *Wenn* $\lambda x t \in \mathsf{sn}$*, dann* $t \in \mathsf{sn}$*.*
4. *Wenn* $t[s/x] \in \mathsf{sn}$*, dann* $t \in \mathsf{sn}$*.*

Beweis. Wir zeigen 1. durch Induktion über $r\,s \in \mathsf{sn}$. Dazu nehmen wir an dass $r \longrightarrow r'$. Da auch $r\,s \longrightarrow r'\,s$, gilt nach Induktionshypothese $r' \in \mathsf{sn}$. Da r' beliebig war, gilt also auch $r \in \mathsf{sn}$. Die anderen Aussagen werden analog bewiesen. □

Lemma 5.14 (Maximale Reduktionslängen von Teiltermen)

1. $\ell(r\,s) \geq \ell(r)$
2. $\ell(r\,s) \geq \ell(s)$
3. $\ell(\lambda xt) \geq \ell(t)$

5.3 Starke β- und $\beta\eta$-Normalisierung

Lemma 5.15 (Abschlusseigenschaften von $\mathsf{sn}_{\beta\eta}$)

1. *Wenn* $\vec{s} \in \mathsf{sn}_{\beta\eta}$, *dann* $x\,\vec{s} \in \mathsf{sn}_{\beta\eta}$.
2. *Wenn* $t \in \mathsf{sn}_{\beta\eta}$, *dann* $\lambda xt \in \mathsf{sn}_{\beta\eta}$.
3. *Wenn* $t, s, \vec{s} \in \mathsf{sn}_{\beta\eta}$ *und* $t[s/x]\,\vec{s} \in \mathsf{sn}_{\beta\eta}$, *dann* $(\lambda xt)\,s\,\vec{s} \in \mathsf{sn}_{\beta\eta}$.

Beweis.

1. Wir zeigen durch Induktion über $\vec{s} \in \mathsf{sn}_{\beta\eta}$, dass jedes Redukt von $x\,\vec{s}$ in $\mathsf{sn}_{\beta\eta}$ ist. Sei $n := |\vec{s}|$. Der Fall $n = 0$ ist trivial. Andernfalls, wenn $x\,\vec{s} \longrightarrow_{\beta\eta} r$, dann $r = x\,s_1 \ldots s_k\,s'\,s_{k+2} \ldots s_n$ mit $s_{k+1} \longrightarrow_{\beta\eta} s'$. Nun ist $r \in \mathsf{sn}_{\beta\eta}$ nach I.V. für s_{k+1}.

2. Analog, durch Induktion über $t \in \mathsf{sn}_{\beta\eta}$. Falls $\lambda xt \longrightarrow_{\beta\eta} \lambda xt'$ mit $t \longrightarrow_{\beta\eta} t'$, dann $\lambda xt' \in \mathsf{sn}_{\beta\eta}$ nach I.V. Andernfalls, $t \equiv t'\,x$ und $\lambda x.\,t'\,x \longrightarrow_{\eta} t'$. Da $t \in \mathsf{sn}_{\beta\eta}$, ist auch der Subterm $t' \in \mathsf{sn}_{\beta\eta}$.

3. Hier geht die Induktion über $t, s, \vec{s} \in \mathsf{sn}_{\beta\eta}$. Falls $(\lambda xt)\,s\,\vec{s} \longrightarrow_{\beta\eta} r$, dann wurde entweder einer Subterme $t, s, \vec{s}$ reduziert, worauf eine I.V. greift, oder $r \equiv t[s/x]\,\vec{s} \in \mathsf{sn}_{\beta\eta}$ nach Voraussetzung, oder $t = t'\,x$ mit $x \notin \mathsf{FV}(t')$ und $r \equiv t'\,s\,\vec{s}$. Dann ist aber wieder $r \equiv t[s/x]\,\vec{s} \in \mathsf{sn}_{\beta\eta}$ nach Voraussetzung.

□

Bemerkung 5.16 sn_{β} *hat diesselben Abschlusseigenschaften.*

Definition 5.17 (Induktive Charakterisierung starker Normalisierung)

$$\text{SN-NE}\ \frac{\vec{s} \in \mathsf{SN}}{x\,\vec{s} \in \mathsf{SN}} \qquad \text{SN-XI}\ \frac{t \in \mathsf{SN}}{\lambda xt \in \mathsf{SN}} \qquad \text{SN-WHR}\ \frac{t[s/x]\,\vec{s} \in \mathsf{SN} \qquad s \in \mathsf{SN}}{(\lambda xt)\,s\,\vec{s} \in \mathsf{SN}}$$

Lemma 5.18 (SN ist korrekt für $\beta\eta$) *Wenn* $t \in \mathsf{SN}$, *dann* $t \in \mathsf{sn}_{\beta\eta}$.

Beweis. Durch Induktion über $t \in \mathsf{SN}$, unter Benutzung der Abschlusseigenschaften von $\mathsf{sn}_{\beta\eta}$. □

Da $\longrightarrow_\beta \subseteq \longrightarrow_{\beta\eta}$, gilt $\mathsf{sn}_{\beta\eta} \subseteq \mathsf{sn}_\beta$. Also ist SN auch korrekt für β alleine.

Lemma 5.19 (SN ist vollständig) *Wenn $r \in \mathsf{sn}_\beta$, dann gilt $r \in \mathsf{SN}$.*

Beweis. Durch lexikographische Induktion über $(\ell_\beta(r), \mathsf{h}(r))$.

Fall $r = \lambda x t$. Dann gilt nach Induktionsvoraussetzung $t \in \mathsf{SN}$, weil ja $\ell_\beta(t) \leq \ell_\beta(\lambda x t)$ und $\mathsf{h}(t) < \mathsf{h}(\lambda x t)$. Damit $\lambda x t \in \mathsf{SN}$ nach Regel SN-XI.

Fall $r = x\,\vec{s}$. Analog mit nach Regel SN-NE.

Fall $r = (\lambda x t)\,s\,\vec{s} \longrightarrow_\beta r' := t[s/x]\,\vec{s}$. Zuerst ist nach Induktionsvoraussetzung $r' \in \mathsf{SN}$, weil $\ell_\beta(r') < \ell_\beta(r)$. Wieder nach Induktionsvoraussetzung ist $s \in \mathsf{SN}$. Zusammen, mit Regel SN-WHR, $r \in \mathsf{SN}$.

□

Satz 5.20 $\mathsf{sn}_{\beta\eta} = \mathsf{sn}_\beta = \mathsf{SN}$.

Beweis. $\mathsf{SN} \subseteq \mathsf{sn}_{\beta\eta} \subseteq \mathsf{sn}_\beta \subseteq \mathsf{SN}$. □

5.4 Übungen

Übung 5.21 *Zeigen Sie: Eine stark normalisierende Relation ist irreflexiv.*

Übung 5.22 *Die Relation $\prec$ auf $\mathbb{N}$ sei induktiv definiert durch:*

$$\frac{}{2n-1 \prec 2n+1} \qquad \frac{}{3n \prec 2n}$$

Beweisen Sie oder widerlegen Sie die Aussage $\prec$ ist wohlfundiert.

Übung 5.23 (Fundiertheit des lexikographischen Produkts) *Seien R_1, R_2 zwei (nicht-reflexive) wohlfundierte Relationen. Zeigen Sie, dass das lexikographische Produkt R von R_1 und R_2 wohlfundiert ist. [Hinweis: $(a,b)\ R\ (a',b')$ gdw. $a\ R_1\ a'$ oder $a = a'$ und $b\ R_2\ b'$.]*

Übung 5.24 (Fundiertheit der lexikographischen Ordnung auf Listen) *Sei R eine strikte Ordnung auf einer Menge M. Zeigen Sie, dass die lexikographische Ordnung R^* auf M^* wohlfundiert ist. Dabei gilt $a_1 a_2 \ldots a_m\ R^*\ b_1 b_2 \ldots b_n$ wenn $m < n$, oder wenn $m = n$ und es ein $k < m$ gibt mit $a_i = b_i$ für $i < k$ und $a_k\ R\ b_k$.*

Übung 5.25 (Affiner λ-Kalkül ist s.n.) *Der* affine λ-Kalkül *ist der λ-Kalkül mit folgender Einschränkung: Für jedem Abstraktionsterm $\lambda x t$ darf die gebundene Variable x höchstens einmal in t vorkommen.*

Zeigen Sie, dass die β-Reduktion im affinen λ-Kalkül stark normalisierend ist.

Übung 5.26 (Starke und schwache Normalisierung identisch in λI) *Der λI-Kalkül ist folgende Einschränkung des λ-Kalküls: Für jedem Abstraktionsterm $\lambda x t$ gilt $x \in \mathsf{FV}(t)$. Leere Abstraktionen, wie sie z.B. im Term K vorkommen, sind im λI-Kalkül verboten. Der λI-Kalkül geht auf Church (1941) zurück, wurde aber durch die heutige Form des λ-Kalküls, der zur Unterscheidung auch λK-Kalkül genannnt wird, verdrängt.*

Zeigen Sie:

1. *Sei $\longrightarrow$ eine kongruente Reduktionsrelation. Wenn $t[s/x] \in \mathsf{sn}_{\longleftarrow}$ und $x \in \mathsf{FV}(t)$, dann $s \in \mathsf{sn}_{\longleftarrow}$.*

2. *Im λI-Kalkül fallen starke und schwache Normalisierung zusammen. [Hinweis: Verwenden Sie die induktiven Charakterisierungen WN und SN.]*

Übung 5.27 (SN für Paare) *Betrachten Sie die Erweiterung des λ-Kalküls um Paare wie in Übung 4.30, mit dem zusätzlichen Axiomenschema $\mathsf{Pair}\,(\mathsf{fst}\,t)\,(\mathsf{snd}\,t) \longrightarrow_\eta t$.*

Ein Evaluationskontext *ist ein Term mit einem Loch $[\,]$ in schwacher Kopfposition, gegeben durch die Grammatik $E ::= [\,] \mid E\,s \mid \mathsf{fst}\,E \mid \mathsf{snd}\,E$. Die Einsetzung von r in das Loch von E bezeichnen wir mit $E[r]$. Beispiele für Evaluationskontexte sind $[\,]\,\vec{s}$ und $(\mathsf{fst}\,([\,]\,s))\,t$, keine Evaluationskontexte sind z.B. $\lambda x.\,[\,]$ und $r\,[\,]$.*

Für den erweiterten Kalkül definieren wir SN wie folgt:

$$\frac{\vec{s} \in \mathsf{SN}}{x\,\vec{s} \in \mathsf{SN}} \qquad \frac{t \in \mathsf{SN}}{\lambda x t \in \mathsf{SN}} \qquad \frac{r, s \in \mathsf{SN}}{\mathsf{Pair}\,r\,s \in \mathsf{SN}}$$

$$\frac{E[t[s/x]] \in \mathsf{SN} \qquad s \in \mathsf{SN}}{E[(\lambda x t)\,s] \in \mathsf{SN}}$$

$$\frac{E[r] \in \mathsf{SN} \qquad s \in \mathsf{SN}}{E[\mathsf{fst}\,(\mathsf{Pair}\,r\,s)] \in \mathsf{SN}} \qquad \frac{E[s] \in \mathsf{SN} \qquad r \in \mathsf{SN}}{E[\mathsf{snd}\,(\mathsf{Pair}\,r\,s)] \in \mathsf{SN}}$$

1. *Finden Sie einen stark $\beta\eta$-normalisierenden Term t mit $t \notin \mathsf{SN}$. [Hinweis: Ist notwendigerweise ein "unsinniger" Term.]*

2. *Zeigen Sie: $\mathsf{SN} \subseteq \mathsf{sn}_{\beta\eta}$.*

Kapitel 6

Lambda-Theorien

Definition 6.1 (Gleichung) *Eine* (geschlossene) Gleichung *ist ein Paar* $r = s$ *von (geschlossenen) Termen.*

Definition 6.2 (λ-Theorie) *Ist* T *eine Menge von geschlossenen Termen, so bezeichne* T^+ *den Abschluss von* T *unter* β *und den Kompatibilitäts- und Äquivalenzregeln.* T *ist* konsistent*, falls* $T \neq \Lambda^0 \times \Lambda^0$ *(es also eine geschlossene Gleichung* $\notin T$ *gibt).* T *ist eine* λ-Theorie *falls* T *konsistent und* $T = T^+$*. Wir schreiben* $T \vdash r = s$ *falls* $r = s$ *in der Theorie* T *herleitbar ist.*

Die (leere) β-Theorie $\emptyset^+$ bezeichnen wir mit λ, die $\beta\eta$-Theorie mit η.

Definition 6.3 (Erweiterung) *Mit* $T + (r = s)$ *bezeichnen wir die Theorie* $(T \cup (r = s))^+$*. Mit* $T\eta$ *ist* $(T \cup \{\lambda x.\, t\, x = t \mid t \in \Lambda, x \notin \mathsf{FV}(t)\})^+$ *gemeint.*

Wir schreiben $T \vdash r \# s$, falls $T + (r = s)$ inkonsistent.

Lemma 6.4 *Ist* T *eine* λ*-Theorie, so gilt* $T\eta = T + (\mathbf{1} = \mathsf{I})$.

Beweis. Für die Hinrichtung genügt es zu sehen, dass $\mathbf{1} =_{\beta\eta} \mathsf{I}$. Die Rückrichtung folgt für $x \notin \mathsf{FV}(t)$ aus $\lambda x.\, t\, x = \mathbf{1}\, t = \mathsf{I}\, t = t$. □

6.1 Böhm's Theorem

Definition 6.5 (Kontext) *Ein Kontext ist ein Term mit beliebig vielen Vorkommen eines Lochs* $[\,]$.

$$\begin{array}{rcll} C & ::= & [\,] & \textit{Loch} \\ & \mid & x \mid C\, C' \mid \lambda x.\, C & \end{array}$$

Auf Kontexte wird die α*-Gleichheit* nicht *angewendet. Die Einsetzung von* t *in das Loch* $[\,]$*, die Variablen von* t *gefangennehmen kann, wir mit* $C[t]$ *bezeichnet.*

Satz 6.6 (Böhm, 1968) *Sind v, v' zwei verschiedene $\beta\eta$-Normalformen, dann ist $\lambda\eta \vdash v \# v'$.*

Der Satz folgt aus der Aussage: Sind v, v' zwei verschiedene $\beta\eta$-Normalformen, dann gibt es einen Kontext C mit $C[v] =_\beta \mathsf{T}$ und $C[v'] =_\beta \mathsf{F}$.

Ein Herleitungssystem für hypothetische Gleichungen.

$$\textsc{app}\ \frac{r = s}{r\,t = s\,t} \qquad \textsc{subst}\ \frac{r = s}{r[t/x] = s[t/x]} \qquad \textsc{eq}\ \frac{r = s}{r' = s'}\ r \longrightarrow_\beta^* r', s \longrightarrow_\beta^* s'$$

Wir schreiben $r = s \vdash r' = s'$, falls es eine Herleitung

$$\begin{array}{c} r = s \\ \hline \\ \vdots \\ \\ \hline r' = s' \end{array}$$

gibt. In diesem Fall können wir einen Kontext $C(r, s)$ durch Rekursion über die Herleitung gewinnen, so dass $C[r] =_\beta r'$ und $C[s] =_\beta s'$:

$$\begin{array}{lrcl} \textsc{app} & C(r,s) & = & C(r\,t,\ s\,t)[[\,]\,t] \\ \textsc{subst} & C(r,s) & = & C(r[t/x],\ s[t/x])[(\lambda x[\,])\,t] \\ \textsc{eq} & C(r,s) & = & C(r',s') \end{array}$$

Nun erstellen wir schrittweise Herleitungen $r = s \vdash \mathsf{T} = \mathsf{F}$, für verschiedene Fälle von Termen r, s mit unterschiedlicher $\beta\eta$-Normalform.

Leicht unterscheidet man Terme mit unterschiedlichen Variablen im Kopf:

$$\cfrac{\cfrac{\cfrac{x\,\vec{r} = y\,\vec{s}}{(x\,\vec{r})[\lambda\vec{x}.\mathsf{T}/x] = y\,\vec{s}}\ \textsc{subst}, |\vec{x}| = |\vec{r}|}{\mathsf{T} = y\,\vec{s}}\ \textsc{eq}}{\mathsf{T} = \mathsf{F}}\ \textsc{subst}, \textsc{eq} \tag{6.1}$$

Tupel und Selektoren können im λ-Kalkül representiert werden. Wir schreiben $\langle t_1, \ldots, t_n\rangle$ für $\lambda x.\, x\, t_1 \ldots t_n$.

$$\begin{array}{lcl} \mathsf{tup}^n & = & \lambda x_1 \ldots x_n.\, \langle x_1, \ldots, x_n\rangle \\ \mathsf{proj}_i^n & = & \lambda x_1 \ldots x_n.\, x_i \end{array}$$

Es gilt $\mathsf{tup}^n\, t_1 \ldots t_n\, \mathsf{proj}_i^n =_\beta t_i$.

Die Terme $\lambda\vec{x}.\,\langle\vec{r}\rangle$ und $\lambda\vec{y}.\,\langle\vec{s}\rangle$ können auch leicht unterschieden werden, wenn $|\vec{x}| \neq |\vec{y}|$. Betrachten wir den Fall $<$:

$$\cfrac{\cfrac{\lambda\vec{x}x.\, x\,\vec{r} = \lambda\vec{x}x\vec{y}y.\, y\,\vec{s}}{(\lambda\vec{x}x.\, x\,\vec{r})\,\vec{x}\,x\,\vec{y}\,y = (\lambda\vec{x}x\vec{y}y.\, y\,\vec{s})\,\vec{x}\,x\,\vec{y}\,y}\ \textsc{app}}{x\,\vec{r}\,\vec{y}\,y = y\,\vec{s}}\ \textsc{eq} \tag{6.2}$$

Diese Herleitung können wir mit (6.1) zu $\mathsf{T} = \mathsf{F}$ ergänzen.

Ist die gleiche Variable im Kopf, aber auf unterschiedlich viele Argumente angewendet, können wir auch leicht eine Inkonsistenz herleiten. Sei im Folgenden $|\vec{r}| = |\vec{s}|$, $|\vec{t}| = |\vec{y}|$ und $n = |\vec{s}, s, \vec{t}|$.

$$\dfrac{\dfrac{x\,\vec{r} = x\,\vec{s}\,s\,\vec{t}}{(x\,\vec{r})[\mathsf{tup}^n/x] = (x\,\vec{s}\,s\,\vec{t})[\mathsf{tup}^n/x]}\,\text{SUBST}}{\lambda y\vec{y}.\,\langle \vec{r}', y, \vec{y}\rangle = \langle \vec{s}', s', \vec{t}'\rangle}\,\text{EQ} \tag{6.3}$$

Hierin ist $s' = s[\mathsf{tup}^n/x]$ und analog für $\vec{r}', \vec{s}', \vec{t}'$. Zur Inkonsistenz kommen wir mit (6.2).

Diese Herleitung lässt sich direkt erweitern, um die Inkonsistenz von $\lambda\vec{x}.\,\langle\vec{r}\rangle = \lambda\vec{x}.\,\langle\vec{s}, s, \vec{t}\rangle$ zu zeigen:

$$\dfrac{\lambda\vec{x}x.\,x\,\vec{r} = \lambda\vec{x}x.\,x\,\vec{s}\,s\,\vec{t}}{x\,\vec{r} = x\,\vec{s}\,s\,\vec{t}}\,\text{APP, EQ} \tag{6.4}$$

Wir schreiben $r \# s$ für $r = s \vdash \mathsf{T} = \mathsf{F}$.

Lemma 6.7 *Zusammenfassung.*

1. $x\,\vec{r} \# y\,\vec{s}$, *falls* $x \neq y$.
2. $\lambda\vec{x}.\langle\vec{r}\rangle \# \lambda\vec{y}.\langle\vec{s}\rangle$, *falls* $|\vec{x}| \neq |\vec{y}|$.
3. $x\,\vec{r} \# x\,\vec{s}$, *falls* $|\vec{r}| \neq |\vec{s}|$.
4. $\lambda\vec{x}.\langle\vec{r}\rangle \# \lambda\vec{x}.\langle\vec{s}\rangle$, *falls* $|\vec{r}| \neq |\vec{s}|$.

Simultane Substitution: Ist X eine Menge von Variablen und $\sigma \in X \to \Lambda$, so schreiben wir $t\sigma$ für die simultane Ersetzung von jedem $x \in X$ durch $\sigma(x)$ in t. Mit $\sigma[x \mapsto t]$ bezeichnen wir die Substitution $\sigma' \in (X \cup \{x\}) \to \Lambda$ mit $\sigma'(x) = t$ und $\sigma'(y) = \sigma(y)$ für $x \neq y$.

Wir schreiben $\vec{s}\sigma$ für $s_1\sigma, \ldots, s_n\sigma$, falls $|\vec{s}| = n$. Der Term $r\,\vec{s}\sigma$ bedeutet dann $r\,(s_1\sigma)\ldots(s_n\sigma)$.

Lemma 6.8 (Verallgemeinerung des Böhmschen Satzes) *Sind v, v' zwei verschiedene $\beta\eta$-Normalformen und $X = \{x_1, \ldots, x_k\}$ eine Menge paarweise verschiedener Variablen. Dann gibt es für beliebige, jedoch genügend große und paarweise verschiedene $n_1, \ldots, n_k \in \mathbb{N}$, eine Herleitung von $v\sigma \# v'\sigma$, wobei $\sigma \in X \to \Lambda$ mit $\sigma(x_i) = \mathsf{tup}^{n_i}$ für $i = 1, \ldots, k$.*

Beweis. Durch Induktion über $|v| + |v'|$, wobei $|t|$ die Anzahl der Variablen und λs in t bezeichnet. Also $|x| = 1$, $|\lambda x t| = 2 + |t|$, $|t\,t'| = |t| + |t'|$.

Fall Beide Terme sind Abstraktionen. Dann o.B.d.A. $v = \lambda x r$ und $v' = \lambda x s$ für $x \notin X$. Trivialerweise gilt auch $x \notin \mathsf{FV}(\sigma(x_i))$ für alle x_i, da jedes $\sigma(x_i)$ geschlossen .

$$\dfrac{\dfrac{\dfrac{(\lambda x r)\sigma = (\lambda x s)\sigma}{(\lambda x.\, r\sigma)\, x = (\lambda x.\, s\sigma)\, x}\,\text{APP}}{r\sigma = s\sigma}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,I.H.$$

Die I.H. ist anwendbar, da r und s notwendigerweise verschiedene $\beta\eta$-Normalformen sind und $|r| + |s| < |v| + |v'|$.

Fall Nur ein Term ist eine Abstraktion. O.B.d.A. $v' = \lambda x s$ wobei $x \notin \mathsf{FV}(v)$ und $x \notin X$ gewählt wird.

$$\dfrac{\dfrac{\dfrac{v\sigma = (\lambda x s)\sigma}{v\sigma\, x = (\lambda x.\, s\sigma)\, x}\,\text{APP}}{(v\, x)\sigma = s\sigma}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,I.H.$$

Der Term s ist normal, und da v normal und keine Abstraktion, ist auch $v\, x$ normal. Wäre $v\, x \equiv s$, dann kann $\lambda x s$ nicht η-normal gewesen sein. Also sind $v\, x$ und s zwei verschiedene $\beta\eta$-Normalformen mit $|v\, x| + |s| = 1 + |v| + |s| < |v| + |\lambda x s|$, die I.H. ist also anwendbar.

In den verbleibenden Fällen sind v und v' neutral.

- $v = x\, \vec{r}$ und $v' = y\, \vec{s}$ mit $x, y \notin X$. Falls $x \neq y$ oder $|\vec{r}| \neq |\vec{s}|$ sind wir nach Lemma 6.7, angewendet auf $v\sigma$ und $v'\sigma$, sofort fertig. Andernfalls gilt $r_i \not\equiv s_i$ für ein $1 \leq i \leq p := |\vec{r}| = |\vec{s}|$. Definiere $X' = X \cup \{x\}$ und $\sigma' \in X' \to \Lambda$ durch $\sigma' = \sigma[x \mapsto \mathsf{tup}^n]$, wobei $n \geq p$.

$$\dfrac{\dfrac{\dfrac{v\sigma = v'\sigma}{v\sigma' = v'\sigma'}\,\text{SUBST}}{\mathsf{tup}^n\, (r_1\sigma') \ldots (r_p\sigma')\, y_{p+1} \ldots y_n\, \mathsf{proj}_i^n = \mathsf{tup}^n\, (s_1\sigma') \ldots (s_p\sigma')\, y_{p+1} \ldots y_n\, \mathsf{proj}_i^n}\,\text{APP}}{r_i\sigma' = s_i\sigma'}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,I.H.$$

- $v = x_i\, \vec{r}$ und $v' = y\, \vec{s}$ mit $y \notin X$. Sei $p = |r|$ und $n_i \geq p$. Seien z und $\vec{z} = z_p + 1, \ldots z_{n_i}$ neue Variablen.

$$\dfrac{\dfrac{\dfrac{v\sigma = v'\sigma}{v\sigma\, z_{p+1} \ldots z_{n_i}\, z = v\sigma\, z_{p+1} \ldots z_{n_i}\, z}\,\text{APP}}{z\, \vec{r}\sigma\, \vec{z} = y\, \vec{s}\sigma\, \vec{z}}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,\text{Lemma 6.7}$$

- $v = x_i\,\vec{r}$ und $v' = x_j\,\vec{s}$ mit $i \neq j$. Sei $n_i \geq p = |\vec{r}|$ und $n_j \geq q = |\vec{s}|$ mit $n_i \neq n_j$. Seien $\vec{y} = y_{p+1}, \ldots, y_{n_i}$ und $\vec{z} = z_{q+1}, \ldots, z_{n_j}$ neue Variablen.

$$\dfrac{\dfrac{\mathsf{tup}^{n_i}\vec{r}\sigma = \mathsf{tup}^{n_j}\vec{s}\sigma}{\lambda\vec{y}.\langle\vec{r}\sigma, \vec{y}\rangle = \lambda\vec{z}.\langle\vec{s}\sigma, \vec{z}\rangle}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,\text{Lemma 6.7}$$

  Das Lemma ist anwendbar, da wegen $n_i \neq n_j$ entweder $|\vec{y}| \neq |\vec{z}|$ or $|\vec{r}| \neq |\vec{s}|$.

- $v = x_i\,\vec{r}$ und $v' = x_i\,\vec{s}$. Sei o.B.d.A. $n_i \geq p := |\vec{r}| \geq q := |\vec{s}|$. Wir wählen neue Variablen $\vec{z} = z_{q+1}, \ldots, z_p$ und $\vec{z'} = z_{p+1}, \ldots, z_{n_i}$. Falls $p = q$, so gibt es ein $j \leq p$ mit $r_j \not\equiv s_j$. Ansonsten sei $j = q + 1$.

$$\dfrac{\dfrac{\dfrac{\mathsf{tup}^{n_i}\,\vec{r}\sigma = \mathsf{tup}^{n_i}\,\vec{s}\sigma}{\mathsf{tup}^{n_i}\,\vec{r}\sigma\,\vec{z'}\,\mathsf{proj}_j^{n_i} = \mathsf{tup}^{n_i}\,\vec{s}\sigma\,\vec{z}\,\vec{z'}\,\mathsf{proj}_j^{n_i}}\,\text{APP}}{r_j\sigma = s_j\sigma \;\text{ oder }\; r_j\sigma = z_j}\,\text{EQ}}{\mathsf{T} = \mathsf{F}}\,\text{I.H. oder Lemma 6.7}$$

  Falls $r_j\sigma$ keine λ-Abstraktion ist, können wir im Fall $r_j\sigma = z_j$ direkt mit Lemma 6.7 abschließen, andernfalls mit der I.H.

□

Korollar 6.9 *Seien v, v' zwei verschiedene, geschlossene $\beta\eta$-Normalformen, dann gibt es Terme $\vec{s}$ mit $v\,\vec{s} = \mathsf{T}$ und $v\,\vec{s} = \mathsf{F}$.*

Beweis. Nach dem Lemma gibt es einen Kontext $C = C(v, v')$ mit $C[v] = \mathsf{T}$ und $C[v'] = \mathsf{F}$. Durch Induktion über die Entstehung von C beweisen wir $C[r] = r\,\vec{s}$ für einen geschlossenen Term r.

- APP $C = C'\,t$. Klar nach I.H.
- SUBST $C = (\lambda x C')\,t$. Nach I.H. ist $C'[r] = r\,\vec{s}$, also $C[r] = (\lambda x.\,C'[r])\,t = (\lambda x.\,r\,\vec{s})\,t = r\,\vec{[t/x]s}$.
- EQ $C = C'$. Klar nach I.H.

□

Eine Konsequenz aus Böhms Theorem ist, dass man $\lambda\eta$ höchstens um Gleichungen zwischen Termen erweitern kann, von denen mindestens einer keine Normalform hat.

Setzt man in λ alle Terme gleich, die keine *Kopf-Normalform* haben, so erhält man die Theorie $\mathcal{H}$. Eben so kann man $\lambda\eta$ zu $\mathcal{H}\eta$ erweitern. Diese Theorien motivieren sich folgendermaßen: Terme, die keine Kopfnormalform haben,

divergieren in jedem Kontext C. Daher sind sie durch "Experimente" oder *Beobachtungen* nicht unterscheidbar, und man kann sie identifizieren. Dies lässt sich jedoch nicht auf Terme übertragen, die nur keine Normalform habe. Sei $t_1 = \mathsf{Y}\,\mathsf{tup}^1$ und $t_2 = \mathsf{Y}\,\mathsf{succ}$, dann gilt:

$$\begin{array}{rcl} t_1\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) & = & (\lambda x.\, x\, t_1)\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) \\ & = & (\lambda x.\, x\, t_1)\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) \\ & = & \mathsf{K}\ (\mathsf{K}\ \mathsf{T})\ t_1\ (\mathsf{K}\ \mathsf{F}) \\ & = & \mathsf{K}\ \mathsf{T}\ (\mathsf{K}\ \mathsf{F}) \\ & = & \mathsf{T} \end{array}$$

$$\begin{array}{rcl} t_2\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) & = & (\lambda f x.\, f\ (t_2\ f\ x))\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) \\ & = & \mathsf{tup}^1\ (t_2\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T}))\ (\mathsf{K}\ \mathsf{F}) \\ & = & \mathsf{K}\ \mathsf{F}\ (t_2\ \mathsf{tup}^1\ (\mathsf{K}\ (\mathsf{K}\ \mathsf{T})) \\ & = & \mathsf{F} \end{array}$$

An diesem Beispiel sieht man, dass sich Böhm's Theorem erweitern lässt auf Terme, die unterschiedliche Böhm-Bäume besitzen, d.h., unterschiedliche iterierte Kopf-Normalformen, wobei die jeweiligen Subterme, die keine Kopf-Normalform besitzen, durch $\bot$ ersetzt werden.

Teil II

Getypte Lambda-Kalküle

Rohfassung vom 17. November 2010

Kapitel 7

Einfach-getypter Lambda-Kalkül

Engl. simply-typed lambda-calculus (STL).

Definition 7.1 (Einfacher Typ) *Sei* Ty^0 *eine (höchstens abzählbare) Menge von* Grundtypen*, die wir mit den Metavariablen a, b und o bezeichnen. Die Menge* Ty *der* einfachen Typen *ist gegeben durch die Grammatik.*

$$\begin{array}{rcll} \mathsf{Ty} \ni A, B, C & ::= & o & \textit{Grundtyp} \\ & | & A \to B & \textit{Funktionstyp} \end{array}$$

Wir sprechen von "einfachen" Typen, um uns z.B. gegen polymorphe Typen abzugrenzen.

Der Pfeil assoziiert nach rechts, also $A \to B \to C = A \to (B \to C) \neq (A \to B) \to C$. Wir schreiben $\vec{A} \to B$ für $A_1 \to (A_2 \to \cdots \to (A_n \to B) \ldots))$.

Definition 7.2 (Ordnung) *Die Ordnung* $\mathsf{ord}(A)$ *eines Typen ist rekursiv definiert durch:*

$$\begin{array}{lcl} \mathsf{ord}(o) & = & 0 \\ \mathsf{ord}(A \to B) & = & \max(\mathsf{ord}(A) + 1, \mathsf{ord}(B)) \end{array}$$

Der Typ einer "normalen" Funktion, z.B. $o \to o \to o$, hat Ordnung 1, Funktionen *höherer Ordnung* haben Typen mit Ordung ≥ 2, z.B., $(o \to o) \to o \to o$.

Definition 7.3 (Typisierungskontext) *Ein Typisierungskontext* Γ *ist eine endliche Abbildung aus der Menge der Variablen* V *in die Menge der Typen* Ty*. Der endliche Definitionsbereich von* Γ *wird mit* $\mathsf{dom}(\Gamma)$ *bezeichnet. Wir schreiben* $\Gamma(x) = A$*, wenn* Γ *der Variable x den Typ A zuweist.*

Definition 7.4 (Kontexterweiterung) *Mit $\Gamma, x:A$ bezeichnen wir die Abbildung Γ' mit*

$$\begin{array}{lcll} \Gamma'(x) & = & A & \\ \Gamma'(y) & = & \Gamma(y) & \text{falls } y \neq x. \end{array}$$

Definition 7.5 (Typisierung/Typzuweisung (engl. type assignment)) *Die Typzuweisung ist eine dreistellige Relation $\Gamma \vdash t : A$ (in mixfix-Schreibweise) zwischen einem Kontext Γ, einem Term t und einem Typ A, induktiv gegeben durch folgende Regeln:*

$$\text{TY-VAR}\ \frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \text{TY-ABS}\ \frac{\Gamma, x:A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B} \qquad \text{TY-APP}\ \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

Beispiel 7.6

- $\vdash \lambda x x : A \to A$
- *Typherleitung für Church-Ziffer 1:*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f:A \to B, x:A \vdash f : A \to B}\,\text{TY-VAR} \qquad \dfrac{}{f:A \to B, x:A \vdash x : A}\,\text{TY-VAR}}{f:A \to B, x:A \vdash f\, x : B}\,\text{TY-APP}}{f:A \to B \vdash \lambda x.\, f\, x : A \to B}\,\text{TY-ABS}}{\vdash \lambda f x.\, f\, x : (A \to B) \to (A \to B)}\,\text{TY-ABS}$$

- $\vdash \lambda f x.\, f\,(f\, x) : (A \to A) \to (A \to A)$
- $\vdash \lambda x z.\, z\, x : A \to (A \to B) \to B$
- $\vdash \lambda x y z.\, z\, x\, y : A \to B \to (A \to B \to C) \to C$
- $\not\vdash \lambda x.\, x\, x : A$ *for any* A.

7.1 Typerhaltung unter Reduktion

Lemma 7.7 (Abschwächung (engl. weakening)) *Wenn $\Gamma \vdash t : C$ und $x \notin \mathsf{dom}(\Gamma)$, dann $\Gamma, x:A \vdash t : C$.*

Beweis. Durch Induktion über die Herleitung von $\Gamma \vdash t : C$. □

Lemma 7.8 (Verstärkung (engl. strengthening)) *Wenn $\Gamma, x:A \vdash t : C$ und $x \notin \mathsf{FV}(t)$, dann $\Gamma \vdash t : C$.*

Beweis. Durch Induktion über die Typisierungherleitung. □

Lemma 7.9 (Substitution) *Wenn $\Gamma, x:A \vdash t : C$ und $\Gamma \vdash s : A$, dann $\Gamma \vdash t[s/x] : C$.*

Beweis. Durch Induktion über $\Gamma, x:A \vdash t : C$. □

Lemma 7.10 (Inversion der Typisierung)

1. *Wenn* $\Gamma \vdash x : C$, *dann* $\Gamma(x) = C$.
2. *Wenn* $\Gamma \vdash \lambda x t : C$, *dann gibt es* A, B *mit* $C \equiv A \to B$ *und* $\Gamma, x:A \vdash t : B$.
3. *Wenn* $\Gamma \vdash r\, s : C$, *dann gibt es ein* A *mit* $\Gamma \vdash r : A \to C$ *und* $\Gamma \vdash s : A$.

Beweis. Trivial, jeweils durch Fallunterscheidung über die Typherleitung. □

Satz 7.11 (Typerhaltung unter Reduktion (subject reduction)) *Wenn* $\Gamma \vdash t : C$ *und* $t \longrightarrow_{\beta\eta} t'$, *dann* $\Gamma \vdash t' : C$.

Beweis. Durch Induktion über $t \longrightarrow_{\beta\eta} t'$.

Fall $t \equiv (\lambda x r)\, s \longrightarrow_\beta t[s/x]$. Die Typisierungherleitung von t hat notwendigerweise die Gestalt:

$$\dfrac{\dfrac{\Gamma, x:A \vdash r : C}{\Gamma \vdash \lambda x r : A \to C}\,\text{TY-ABS} \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x r)\, s : C}\,\text{TY-ABS}$$

Mit dem Substitutionslemma ist $\Gamma \vdash r[s/x] : C$.

Fall $t \equiv \lambda x.\, r\, x \longrightarrow_\eta r$ mit $x \notin \mathsf{FV}(r)$. Hier hat die Typisierungsherleitung von t die Gestalt:

$$\dfrac{\dfrac{\Gamma, x:A \vdash r : A \to B \qquad \dfrac{}{\Gamma, x:A \vdash x : A}\,\text{TY-VAR}}{\Gamma, x:A \vdash r\, x : B}\,\text{TY-APP}}{\Gamma \vdash \lambda x.\, r\, x : A \to B}\,\text{TY-ABS}$$

Mit dem Verstärkungslemma folgt $\Gamma \vdash r : A \to B$.

Fall $t \equiv \lambda x r \longrightarrow_{\beta\eta} \lambda x r'$. Nach dem Inversionlemma gilt $C \equiv A \to B$ und $\Gamma, x : A \vdash r : B$. Nach I.H. ist auch $\Gamma, x : A \vdash r' : B$, also mit TY-ABS schliesslich $\Gamma \vdash \lambda x r' : A \to B$.

Fall Kongruenzregel für Applikation analog.

□

7.2 Starke Normalisierung

Ein *neutraler* Term ist von der Form $x\,\vec{s}$.

Für den einfach getypten λ-Kalkül zeigt man schwache Normalisierung sehr leicht (Girard et al., 1989, Kap. 4): Sei der Variable x der Typ $C = \vec{A} \to B$ zugewiesen. Der Grad eines Redexes $(\lambda x t)\,s$ sei $\mathsf{ord}(C)$. Bei der Reduzierung von $(\lambda x t)\,s$ können neue Redexe entstehen, falls s nicht neutral ist. Diese Redexe entstehen in den Subtermen von t der Gestalt $x\,\vec{r}$. Da der Typ A_i jedes Terms r_i eine kleinere Ordung hat als der Typ $\vec{A} \to B$ von s haben diese neuen Redexe einen kleineren Grad. Man kann nun das Gewicht eines Terms definieren als das lexikographische Produkt aus dem höchst vorkommenden Redex-Grad und der Anzahl jener Redexe. Reduziert man immer einen Redex vom höchsten Grad, nimmt das Gewicht des Terms bei jedem Schritt ab.

Diesen Beweis erweitern wir nun formal auf starke Normalisierung.

Definition 7.12 (Getypte s.n. terme) *Die 3-stelligen Relationen $\Gamma \vdash t \downarrow C$ und $\Gamma \vdash t \uparrow C$ sind induktiv gegeben durch die folgenden Regeln:*

$$\text{TY-SN-VAR}\ \frac{}{\Gamma \vdash x \downarrow \Gamma(x)} \qquad \text{TY-SN-APP}\ \frac{\Gamma \vdash r \downarrow A \to B \qquad \Gamma \vdash s \uparrow A}{\Gamma \vdash r\,s \downarrow B}$$

$$\text{TY-SN-NE}\ \frac{\Gamma \vdash r \downarrow A}{\Gamma \vdash r \uparrow A} \qquad \text{TY-SN-ABS}\ \frac{\Gamma, x\!:\!A \vdash t \uparrow B}{\Gamma \vdash \lambda x t \uparrow A \to B}$$

$$\text{TY-SN-EXP}\ \frac{\Gamma \vdash s \uparrow A \qquad \Gamma \vdash r[s/x]\,\vec{s} \uparrow C}{\Gamma \vdash (\lambda x r)\,s\,\vec{s} \uparrow C}$$

Lemma 7.13 (Korrektheit)

1. *Wenn $\Gamma \vdash t \downarrow A$, dann ist t neutral und $t \in \mathsf{SN}$.*
2. *Wenn $\Gamma \vdash t \uparrow A$, dann $t \in \mathsf{SN}$.*

Beweis. Simultan durch Induktion über die Herleitung. □

Lemma 7.14 (Substitution und Applikation) *Sei $\Gamma \vdash s \uparrow A$.*

1. *Wenn $\mathcal{D} :: \Gamma, x\!:\!A \vdash t \downarrow C$, dann entweder $\Gamma \vdash t[s/x] \downarrow C$ oder $\mathsf{ord}(C) \leq \mathsf{ord}(A)$ und $\Gamma \vdash t[s/x] \uparrow C$.*
2. *Wenn $\mathcal{D} :: \Gamma, x\!:\!A \vdash t \uparrow C$, dann $\Gamma \vdash t[s/x] \uparrow C$.*
3. *Wenn $\mathcal{D} :: \Gamma \vdash t \uparrow A \to C$, dann $\Gamma \vdash t\,s \uparrow C$.*

Beweis. Simultan, durch Hauptinduktion über $\mathsf{ord}(A)$ und Nebeninduktion über die Herleitung $\mathcal{D}$. □

Satz 7.15 (STL ist s.n.) *Wenn* $\Gamma \vdash t : C$*, dann* $\Gamma \vdash t \uparrow C$*.*

Beweis. Durch Induktion über $\Gamma \vdash t : C$, unter Benutzung des Applikationslemmas. □

Korollar 7.16 *Jeder einfach getypte Term ist stark* $\beta\eta$*-normalisierend.*

7.3 Church-Stil

Sind die Typen aller Variablen in einem Term bekannt, so ist der Typ des Terms dadurch determiniert. Versieht man also die gebundenen Variablen mit ihrem Typ, so lässt sich der Typ eines geschlossenen Terms direkt berechnen.

Definition 7.17 (Church-Terme) *Der Terme des einfach getypten λ-Kalküls im Church-Stil sind gegeben durch die Grammatik.*

$$r, s, t ::= x \mid \lambda x{:}A.\, t \mid r\, s$$

Definition 7.18 (Typisierung für Church-Terme) *Induktiv gegeben durch folgende Regeln:*

$$\text{TY-VAR}\ \frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \text{TY-ABS}\ \frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x{:}A.\, t : A \to B}$$

$$\text{TY-APP}\ \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

Im Gegensatz zur Typzuweisung (type assignment) für ungetypte Terme ist die Typisierung für Church-Terme eindeutig, d.h., der Typ eines Church-Terms t ist durch die Typen seiner freien Variablen eindeutig bestimmt.

Satz 7.19 (Eindeutigkeit der Typisierung) *Wenn $\Gamma \vdash t : A$ und $\Gamma \vdash t : B$, dann $A = B$.*

Beweis. Durch Induktion über $\Gamma \vdash t : A$ und Fallunterscheidung über $\Gamma \vdash t : B$. □

Es ist hilfreich, sich zu überlegen, wo der Beweis für Terme im Curry-Stil fehlschlägt.

Wegen der Eindeutigkeit können wir eine Funktion angeben, die den Typen eines Terms in einem Kontext berechnet, und einen Fehlerwert $\emptyset$ zurückgibt, falls der Term nicht wohlgetypt ist.

Definition 7.20 (Typberechnung) *Gegeben ein Typisierungskontext Γ und ein Church-Term t berechnet die Funktion $\mathsf{typeOf}(\Gamma \vdash t)$ den Typ A von t durch Rekursion über t. Falls t nicht wohlgetypt ist in Γ, wird $\emptyset$ zurückgegeben.*

$$\mathsf{typeOf}(\Gamma \vdash x) = \begin{cases} A & \text{wenn } \Gamma(x) = A \\ \emptyset & \text{sonst} \end{cases}$$

$$\mathsf{typeOf}(\Gamma \vdash \lambda x{:}A.\, t) = \begin{cases} \emptyset & \text{wenn } \mathsf{typeOf}(\Gamma, x{:}A \vdash t) = \emptyset \\ A \to \mathsf{typeOf}(\Gamma, x{:}A \vdash t) & \text{sonst} \end{cases}$$

$$\mathsf{typeOf}(\Gamma \vdash r\, s) = \begin{cases} B & \text{wenn } \mathsf{typeOf}(\Gamma \vdash r) = A \to B \\ & \text{und } \mathsf{typeOf}(\Gamma \vdash s) = A \\ \emptyset & \text{sonst} \end{cases}$$

7.4 Getypte Gleichheit

$\beta\eta$-Gleichheit für einfach getypte Term definieren wir als Urteil $\Gamma \vdash t = t' : C$, gesprochen "in Kontext Γ sind t und t' gleich vom Typ C". Diese 4stellige Relation ist induktiv gegeben durch die folgenden Regeln. Im Prinzip handelt es sich um die kleinste Kongruenz über den Axiomen für β- und η-Kontraktion. Allerdings betrachten wir nur wohlgetypte Terme, und die Prämissen der Regeln stellen sicher, dass wir den Bereich der wohlgetypten Terme nicht verlassen.

$$\text{EQ-}\beta\ \frac{\Gamma, x{:}A \vdash t : B \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x{:}A.\, t)\, s = t[s/x] : B} \qquad \text{EQ-}\eta\ \frac{\Gamma \vdash t : A \to B}{\Gamma \vdash \lambda x{:}A.\, (t\, x) = t : A \to B}\ x \notin \mathsf{FV}(t)$$

$$\text{EQ-}\xi\ \frac{\Gamma, x{:}A \vdash t = t' : B}{\Gamma \vdash \lambda x{:}A.\, t = \lambda x{:}A.\, t' : A \to B}$$

$$\text{EQ-APP}\ \frac{\Gamma \vdash r = r' : A \to B \qquad \Gamma \vdash s = s' : A}{\Gamma \vdash r\, s = r'\, s' : B}$$

$$\text{EQ-REFL}\ \frac{\Gamma \vdash t : C}{\Gamma \vdash t = t : C} \qquad \text{EQ-SYM}\ \frac{\Gamma \vdash t = t' : C}{\Gamma \vdash t' = t : C}$$

$$\text{EQ-TRANS}\ \frac{\Gamma \vdash t_1 = t_2 : C \qquad \Gamma \vdash t_2 = t_3 : C}{\Gamma \vdash t_1 = t_3 : C}$$

Lemma 7.21 (Wohlgetyptheit) *Wenn $\Gamma \vdash t = t' : C$, dann $\Gamma \vdash t : C$ und $\Gamma \vdash t' : C$.*

Beweis. Durch Induktion über das Gleichheitsurteil. □

Lemma 7.22 (Strukturelle Regeln)

1. *Exchange: Wenn $\Gamma, x : A, y : B, \Gamma' \vdash t = t' : C$, dann $\Gamma, y : B, x : A, \Gamma' \vdash t = t' : C$.*

2. *Weakening: Wenn $\Gamma \vdash t = t' : C$, dann $\Gamma, x{:}A \vdash t = t' : C$.*

Beweis. Jeweils durch Induktion über das Gleichheitsurteil. □

Strengthening ist nicht direkt durch Induktion beweisbar. Die Aussage "Wenn $\Gamma, x : A \vdash t_1 = t_3 : C$ und $x \notin \mathsf{FV}(t, t')$ dann $\Gamma \vdash t_1 = t_3 : C$" scheitert an der Regel für Transitivität. Die Induktionshypothese ist nicht anwendbar auf $\Gamma, x : A \vdash t_1 = t_2 : C$ und $\Gamma, x : A \vdash t_2 = t_3 : C$, da x womöglich frei in t_2 vorkommt. Dies ist der Fall, wenn t_2 eine β-Expansion von t_1 und t_3 ist. Wie können wir *strengthening* beweisen?

7.5 Denotationelle Semantik

Wir interpretieren Typen als Mengen und Terme als Bewohner dieser Mengen. λ-Abstraktion wird als mengentheoretische Funktion interpretiert.

Definition 7.23 (Mengentheoretische Funktion) *Eine Funktion $f : X \to Y$ is eine Menge von Paaren $(x, y) \in X \times Y$ so dass es für jedes $x \in X$ genau ein $y \in Y$ gibt mit $(x, y) \in f$). Wir schreiben $f(x)$ für dieses y. Die Menge aller Funktionen von X nach Y bezeichnen wir schlicht mit $X \to Y$.*

Definition 7.24 (Typinterpretation) *Für jeden Grundtyp $o \in \mathsf{Ty}^0$ sei eine Menge M_o gegeben. Die Interpretation $[\![C]\!]$ eines Typen C definieren wir durch Rekursion über C.*

$$\begin{array}{lcl} [\![o]\!] & = & M_o \\ [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \quad \textit{(mengentheoretischer Funktionsraum)} \end{array}$$

Definition 7.25 (Umgebung) *Sei Γ ein Typisierungskontext. Eine Γ-Umgebung ρ ist eine Funktion, die jedes $x \in \mathsf{dom}(\Gamma)$ auf ein Element $d \in [\![\Gamma(x)]\!]$ abbildet. Wir schreiben $\rho \in [\![\Gamma]\!]$.*

Definition 7.26 (Umgebungserweiterung) *Wenn $\rho \in [\![\Gamma]\!]$ und $d \in [\![A]\!]$, dann sei $\rho[x \mapsto d] \in [\![\Gamma, x\!:\!A]\!]$ definiert durch*

$$(\rho[x \mapsto d])(y) = \begin{cases} d & \textit{falls } x = y \\ \rho(y) & \textit{andernfalls.} \end{cases}$$

Definition 7.27 (Denotation) *Sei $\Gamma \vdash t : C$ und $\rho \in [\![\Gamma]\!]$. Wir definieren die Bedeutung des Terms t vom Typ C im Kontext Γ, $[\![\Gamma \vdash t : C]\!]_\rho \in [\![C]\!]$, durch Rekursion über t.*

$$\begin{array}{lcl} [\![\Gamma \vdash x : C]\!]_\rho & = & \rho(x) \\ [\![\Gamma \vdash \lambda x\!:\!A.\, t : B]\!]_\rho & = & f : [\![A]\!] \to [\![B]\!] \\ \quad \textit{wobei } f(a) & = & [\![\Gamma, x\!:\!A \vdash t : B]\!]_{\rho[x \mapsto a]} \\ [\![\Gamma \vdash r\, s : C]\!]_\rho & = & [\![\Gamma \vdash r : A \to C]\!]_\rho([\![\Gamma \vdash s : A]\!]_\rho) \\ \quad \textit{wobei } A \textit{ der eindeutig bestimmte Typ von } s \textit{ in } \Gamma \textit{ ist.} \end{array}$$

Diese Definition ist wohlgeformt.

Bemerkung 7.28 *Kann man auch $[\![t]\!]_\rho \in [\![C]\!]$ definieren? Wie müssen wir die Definition abändern?*

Wenn wir von der Denotation $[\![\Gamma \vdash t : C]\!]$ sprechen setzen wir die Wohlgetyptheit $\Gamma \vdash t : C$ im Folgenden immer voraus.

Lemma 7.29 (Unabhängigkeit der Denotation von der Kontextstruktur)

1. *(exchange)* $[\![\Gamma, x{:}A, y{:}B, \Gamma' \vdash t : C]\!]_\rho = [\![\Gamma, y{:}B, x{:}A, \Gamma' \vdash t : C]\!]_\rho$.

2. *(weakening und strengthening) Wenn* $x \notin \mathsf{FV}(t)$ *und* $a \in [\![A]\!]$*, dann ist* $[\![\Gamma, x{:}A \vdash t : C]\!]_{\rho[x \mapsto a]} = [\![\Gamma \vdash t : C]\!]_\rho$.

Lemma 7.30 (Substitution) *Sei* $\Gamma \vdash s : A$ *und* $\Gamma, x : A \vdash t : C$*. Dann ist* $[\![\Gamma \vdash t[s/x] : C]\!]_\rho = [\![\Gamma, x{:}A \vdash t : C]\!]_{\rho[x \mapsto [\![\Gamma \vdash s:A]\!]_\rho]}$.

Beweis. Durch Induktion über $\Gamma, x{:}A \vdash t : C$. □

Satz 7.31 (Korrektheit der getypten Gleichheit) *Wenn* $\Gamma \vdash t = t' : C$*, dann* $[\![\Gamma \vdash t : C]\!]_\rho = [\![\Gamma \vdash t' : C]\!]_\rho$ *für alle* $\rho \in [\![\Gamma]\!]$.

7.6 Allgemeiner Modellbegriff

Definition 7.32 (Applikative Struktur) *Eine* (getypte) applikative Struktur $(\mathcal{A}, \mathsf{App})$ *ist ein Paar aus einer Typ-Interpretations-Funktion* $\mathcal{A}[\![-]\!]$*, die jedem Typ* C *eine Menge* $\mathcal{A}[\![C]\!]$ *zuordnet, und einer Familie von Funktionen* $\mathsf{App}^{A,B} \in \mathcal{A}[\![A \to B]\!] \times \mathcal{A}[\![A]\!] \to \mathcal{A}[\![B]\!]$ *für* $A, B \in \mathsf{Ty}$.

Eine applikative Struktur ist extensional*, wenn für alle Typen* A, B *und für alle* $f, g \in \mathcal{A}[\![A \to B]\!]$ *gilt:* $\forall a \in \mathcal{A}[\![A]\!]$, $\mathsf{App}^{A,B}(f, a) = \mathsf{App}^{A,B}(g, a)$ *impliziert* $f = g$.

Gunter (**?**, Ch. 2) bezeichnet eine getypte extensionale applikative Struktur als *pre-frame*. Falls aus dem Zusammenhang ersichtlich ist, um welche applikative Struktur es sich handelt, schreiben wir schlicht $[\![C]\!]$ anstatt $\mathcal{A}[\![C]\!]$. Wir sparen uns auch manchmal die Typannotation an der Anwendungsfunktion App.

Definition 7.33 (Term-Interpretation und Umgebungsmodell) *Sei* $(\mathcal{A}, \mathsf{App})$ *eine applikative Struktur. Eine* Γ*-Umgebung* ρ *ist eine Funktion mit Definitionsbereich* $\mathsf{dom}(\Gamma)$*, so dass* $\rho(x) \in \mathcal{A}[\![\Gamma(x)]\!]$ *für alle* $x \in \mathsf{dom}(\Gamma)$*. Eine* Term-Interpretation *ist eine Familie von Funktionen* $(\!|\Gamma \vdash t : C|\!)_- \in \mathcal{A}[\![\Gamma]\!] \to \mathcal{A}[\![C]\!]$*, die folgenden Gleichungen genügt:*

1. $(\!|\Gamma \vdash x : A|\!)_\rho = \rho(x)$,

2. $(\!|\Gamma \vdash r\, s : B|\!)_\rho = \mathsf{App}^{A,B}((\!|\Gamma \vdash r : A \to B|\!)_\rho, (\!|\Gamma \vdash s : A|\!)_\rho)$*, und*

3. $\mathsf{App}^{A,B}((\!|\Gamma \vdash \lambda x{:}A.\, t : A \to B|\!)_\rho, a) = (\!|\Gamma, x{:}A \vdash t : B|\!)_{\rho[x \mapsto a]}$.

Eine applikative Struktur zusammen mit einer Term-Interpretation bezeichnen wir als Umgebungsmodell. Oft werden wir Typ- und Term-Interpretation mit denselben semantischen Klammern $[\![-]\!]$ *bezeichnen.*

Bei Gunter heisst ein extensionales Umgebungsmodell ein *frame*.

Lemma 7.34 (Eindeutigkeit der extensionalen Interpretation) *Über einer extensionalen applikativen Struktur* $(\mathcal{A}, \mathsf{App})$ *ist die Term-Interpretation eindeutig bestimmt.*

Beweis. Seien $[\![-]\!]_-$ und $(\!|-|\!)_-$ zwei Term-Interpretationen. Wir zeigen $[\![\Gamma \vdash t : C]\!]_\rho = (\!|\Gamma \vdash t : C|\!)_\rho$ durch Induktion über $\Gamma \vdash t : C$. □

Es ist sinnvoll, auch nicht-extensionale Modelle zu betrachten. Dort werden Funktionen mit unterschiedlicher Implementierung, jedoch gleichen Verhaltens, nicht notwendigerweise identifiziert. Damit können z.B. Komplexitätsklassen mitmodelliert werden.

Übung 7.35 *Zeigen Sie: Die Mengentheoretische Interpretation aus dem vorigen Abschnitt ist ein extensionales Umgebungsmodell.*

Definition 7.36 (Gültigkeit) *Sei $\mathcal{A}$ ein Umgebungsmodell. Wir sagen, die Gleichung $\Gamma \vdash t = t' : C$ ist* gültig *in $\mathcal{A}$ falls $\mathcal{A}[\![\Gamma \vdash t : C]\!]_\rho = \mathcal{A}[\![\Gamma \vdash t' : C]\!]_\rho$ für alle $\rho \in \mathcal{A}[\![\Gamma]\!]$. Dies schreiben wir als $\Gamma \models_{\mathcal{A}} t = t' : C$ oder auch schlicht als $\Gamma \models t = t' : C$.*

Satz 7.37 (Korrektheit) *Gegeben ein extensionales Umgebungsmodell $\mathcal{A}$. Falls $\Gamma \vdash t = t' : C$, dann $\Gamma \models_{\mathcal{A}} t = t' : C$.*

Beweis. Wir zeigen zuerst, dass Vertauschung, Abschwächung/Verstärkung, und Substitution für ein extensionales Umgebungsmodell genauso gelten wie fur das mengentheoretische Modell. Dann zeigen wir die Aussage des Satzes durch Induktion über die Herleitung von $\Gamma \vdash t = t' : C$. In den Fällen η und ξ benutzen wir Extensionalität. □

Gilt der Satz auch für nicht-extensionale Umgebungsmodelle? Nein, nicht im Allgemeinen. Schon der Beweis der Lemmata für Vertauschung, Abschwächung und Substitution scheitert im Abstraktions-Fall.

Definition 7.38 (Semi-Extensionalität) *Ein Umgebungsmodell $(\mathcal{A}, \mathsf{App}, [\![-]\!])$ heisst* semi-extensional*, falls für alle $\Gamma \vdash \lambda x\!:\!A.t, \lambda x\!:\!A.t' : A \to B$ und $\rho \in [\![\Gamma]\!]$ gilt: $[\![\Gamma \vdash \lambda x : A.\, t : A \to B]\!]_\rho = [\![\Gamma \vdash \lambda x : A.\, t' : A \to B]\!]_\rho$ genau dann wenn $[\![\Gamma, x\!:\!A \vdash t : B]\!]_{\rho[x \mapsto a]} = [\![\Gamma, x\!:\!A \vdash t' : B]\!]_{\rho[x \mapsto a]}$ für alle $a \in [\![A]\!]$.*

Wir verlangen Extensionalität also nicht für alle Werte von Funktionstyp, sondern nur für die Werte von λ-Abstraktionen. Dies ist die ξ-Regel in semantischer Form.

Lemma 7.39 *Vertauschung, Abschwächung und Substitution gelten in semi-extensionalen Modellen.*

Satz 7.40 *Sei $\Gamma \vdash t = t' : C$ hergeleitet ohne Verwendung der η-Regel. Dann $\Gamma \models_{\mathcal{A}} t = t' : C$ in jedem semi-extensionalen Umgebungsmodell $\mathcal{A}$.*

7.7 Termmodelle

Das einfachste vollständige Modell ist das Termmodell: Ein Typ wird durch die Menge seiner Bewohner modulo $\beta\eta$ modelliert. Mit Termmodellen lässt sich elegant die Normalisierung beweisen, d.h. die Existenz einer $\beta\eta$-Normalform für jeden wohlgetypten Term.

7.7.1 Geschlossenes Termmodell

Als Vorbereitung betrachten wir zunächst den einfacheren Fall des geschlossenen Termmodells, in dem jeder Typ durch eine Menge von geschlossenen Termen modulo $\beta\eta$ modelliert wird.

Definition 7.41 (Substitution) *Eine Substitution σ ist eine endliche Abbildung von Variablen auf Terme. Wir schreiben $\sigma[x \mapsto t]$ für die Substitution σ' mit $\sigma'(x) = t$ und $\sigma'(y) = \sigma(y)$ wenn $x \neq y$.*

Da die Variablen eine Teilmenge der Terme sind, können wir σ auch als totale Funktion von den Variablen in die Terme betrachten, wobei $\mathsf{dom}(\sigma) := \{x \mid \sigma(x) \neq x\}$ endlich sein muss.

Definition 7.42 (Parallele Substitution) *Sei σ eine Substitution. Wir definieren die parallele Substitution $t\sigma$ von σ in t durch Rekursion über t:*

$$\begin{array}{lcll} x\sigma & = & \sigma(x) & \text{falls } x \in \mathsf{dom}(\sigma) \\ y\sigma & = & y & \text{falls } y \notin \mathsf{dom}(\sigma) \\ (r\,s)\sigma & = & (r\sigma)\,(s\sigma) & \\ (\lambda x\!:\!A.\,t)\sigma & = & \lambda x\!:\!A.\,t\sigma & \text{wobei o.B.d.A. } x \notin \mathsf{dom}(\sigma) \\ & & & \quad \text{und } x \notin \mathsf{FV}(\sigma(y)) \text{ für alle } y \in \mathsf{dom}(\sigma) \end{array}$$

Definition 7.43 (Typisierung und Gleichheit für Substitutionen) *Wir schreiben $\Delta \vdash \sigma : \Gamma$ falls $\Delta \vdash \sigma(x) : \Gamma(x)$ für alle $x \in \mathsf{dom}(\Gamma)$. Analog sei $\Delta \vdash \sigma = \sigma' : \Gamma$ definiert als $\Delta \vdash \sigma(x) = \sigma'(x) : \Gamma(x)$ für alle $x \in \mathsf{dom}(\Gamma)$.*

Definition 7.44 (Äquivalenzklassen) *Sei t ein geschlossener Term vom Typ A, also $\vdash t : A$. Dann bezeichne $\overline{t}^A = \{t' \mid \vdash t = t' : A\}$ die (getypte) $\beta\eta$-Äquivalenzklasse von t.*

Sei σ eine geschlossene Γ-Substitution, d.h., $\vdash \sigma : \Gamma$. Dann bezeichne $\overline{\sigma}^\Gamma = \{\sigma' \mid \vdash \sigma = \sigma' : \Gamma\}$ die $\beta\eta$-Äquivalenzklasse von σ.

Definition 7.45 (Geschlossenes Termmodell) *Das geschlossene Termmodell $\mathcal{T}_0$ für den einfach getypten Lambda-Kalkül ist gegeben durch:*

$$\begin{array}{lcl} \mathcal{T}_0[\![C]\!] & = & \{\overline{t}^C \mid \vdash t : C\} \\ \mathcal{T}_0[\![\Gamma]\!] & = & \{\overline{\sigma}^\Gamma \mid \vdash \sigma : \Gamma\} \\ \mathcal{T}_0\mathsf{App}^{A,B}(\overline{r}^{A\to B}, \overline{s}^A) & = & \overline{r\,s}^B \\ \mathcal{T}_0[\![\Gamma \vdash t : C]\!]_{\overline{\sigma}^\Gamma} & = & \overline{t\sigma}^C \end{array}$$

Übung 7.46 (Wohldefiniertheit von Applikation und Denotation) *Zeigen Sie, dass* $\mathcal{T}_0\mathsf{App}^{A,B} \in \mathcal{T}_0[\![A \to B]\!] \times \mathcal{T}_0[\![A]\!] \to \mathcal{T}_0[\![B]\!]$ *und* $\mathcal{T}_0[\![\Gamma \vdash t : C]\!] \in \mathcal{T}_0[\![\Gamma]\!] \to \mathcal{T}_0[\![C]\!]$.

Übung 7.47 (Naiver Versuch eines offenen Termmodells) *Sei* $\overline{t}^C = \{t' \mid \Gamma \vdash t = t' : C$ *für ein* $\Gamma\}$ *und* $\mathcal{T}[\![C]\!] = \{\overline{t}^C \mid \Gamma \vdash t : C$ *für ein* $\Gamma\}$. *Zeigen Sie, dass die Applikation* $\mathcal{T}\mathsf{App}^{A,B}(\overline{r}, \overline{s}) = \overline{r\,s}$ *nun nicht wohldefiniert ist.*

Satz 7.48 *Das geschlossene Termmodell ist ein Umgebungsmodell.*

Beweis. Es genügt zu zeigen, dass $\mathcal{T}_0[\![\Gamma \vdash t : C]\!]_{\overline{\sigma}^\Gamma}$ eine Term-Interpretation ist.

1. $[\![\Gamma \vdash x : A]\!]_{\overline{\sigma}} = \overline{x\sigma} = \overline{\sigma(x)} = \overline{\sigma}(x)$.
2. $[\![\Gamma \vdash r\,s : B]\!]_{\overline{\sigma}} = \overline{(r\,s)\sigma} = \overline{(r\sigma)\,(s\sigma)} = \mathsf{App}^{A,B}(\overline{r\sigma}, \overline{s\sigma}) = \mathsf{App}^{A,B}([\![\Gamma \vdash r : A \to B]\!]_{\overline{\sigma}}, [\![\Gamma \vdash s : A]\!]_{\overline{\sigma}})$.
3. $\mathsf{App}^{A,B}([\![\Gamma \vdash \lambda x : A.\, t : A \to B]\!]_{\overline{\sigma}}, \overline{s}) = \overline{(\lambda x\!:\!A.\, t)\sigma\;\overline{s}} = \overline{(\lambda x\!:\!A.\, t\sigma)\,s} = \overline{t\sigma[x \mapsto s]} = [\![\Gamma, x\!:\!A \vdash t : B]\!]_{\overline{\sigma[x\mapsto s]}} = [\![\Gamma, x\!:\!A \vdash t : B]\!]_{\overline{\sigma}[x\mapsto\overline{s}]}$.

□

Das geschlossene Termmodell ist nicht extensional. Da wir keine Konstanten haben, gibt es keinen Bewohner eines Grundtyps im leeren Kontext, also $\mathcal{T}_0[\![o]\!]_= \emptyset$. Damit gilt also trivialerweise $\mathcal{T}_0\mathsf{App}([\![\lambda x\!:\!o.\, \lambda y\!:\!o.\, x]\!]_\rho, a) = \mathcal{T}_0\mathsf{App}([\![\lambda x\!:\!o.\, \lambda y\!:\!o.\, y]\!]_\rho, a)$, aber $[\![\lambda x\!:\!o.\, \lambda y\!:\!o.\, x]\!]_\rho \neq [\![\lambda x\!:\!o.\, \lambda y\!:\!o.\, y]\!]_\rho$ weil ja $\not\vdash \lambda x\!:\!o.\, \lambda y\!:\!o.\, x = \lambda x\!:\!o.\, \lambda y\!:\!o.\, y : o \to o \to o$.

7.7.2 Offenes Termmodell

Sei Γ^∞ ein unendlicher Kontext $x_0\!:\!A_0, x_1\!:\!A_1, \ldots$, so dass jede Variable genau einmal darin vorkommt, aber jeder Typ unendlich vielen Variablen zugeordnet ist. Wir interpretieren nun Terme $\Gamma \vdash t : A$, wobei $\Gamma \subset \Gamma^\infty$ eine endliche Teilmenge von Γ^∞ ist. Damit sind die Typen der freien Variablen festgelegt, und damit ist die Applikation von Termen verträglich, d.h. $\Gamma_1 \vdash r : A \to B$ und $\Gamma_2 \vdash s : A$ implizieren die Existenz eines Kontexts Γ_3 mit $\Gamma_3 \vdash r\,s : B$. Da $\Gamma_1, \Gamma_2 \subset \Gamma^\infty$ können wir z.B. $\Gamma_3 = \Gamma_1 \cup \Gamma_2$ wählen. Dieser Trick erlaubt uns eine einfache Definition des offenen Termmodells.

Zuerst erweitern wir den Begriff der getypten $\beta\eta$-Äquivalenzklasse.

Definition 7.49 (Äquivalenzklassen) *Sei* $\Gamma \vdash t : A$ *für ein* $\Gamma \subset \Gamma^\infty$. *Dann bezeichne* $\overline{t}^A = \{t' \mid \vdash \Gamma' \vdash t = t' : A$ *für ein* $\Gamma' \subset \Gamma^\infty\}$ *die (getypte)* $\beta\eta$*-Äquivalenzklasse von* t.

Sei σ *eine geschlossene* Γ*-Substitution, d.h.,* $\mathsf{dom}(\sigma) = \mathsf{dom}(\Gamma)$ *und es gibt* $\Delta \subset \Gamma^\infty$ *mit* $\Delta \vdash \sigma(x) : \Gamma(x)$ *für alle* x *im Definitionsbereich. Dann bezeichne* $\overline{\sigma}^\Gamma = \{\sigma' \mid \vdash \Delta'\sigma(x) = \sigma'(x) : \Gamma(x)$ *für ein* $\Delta'\}$ *die* $\beta\eta$*-Äquivalenzklasse von* σ.

Definition 7.50 (Geschlossenes Termmodell) *Das geschlossene Termmodell $\mathcal{T}$ für den einfach getypten Lambda-Kalkül ist gegeben durch:*

$$\begin{array}{lcl} \mathcal{T}[\![C]\!] & = & \{\overline{t}^C \mid \Gamma \vdash t : C \textit{ für ein } \Gamma \subset \Gamma^\infty\} \\ \mathcal{T}\mathsf{App}^{A,B}(\overline{r}^{A\to B}, \overline{s}^A) & = & \overline{r\,s}^B \\ \mathcal{T}[\![\Gamma \vdash t : C]\!]_{\overline{\sigma}^\Gamma} & = & \overline{t\sigma}^C \end{array}$$

Lemma 7.51 *Das offene Termmodell ist ein Umgebungsmodell.*

Beweis. Wie beim geschlossenen Termmodell. □

Lemma 7.52 *Das offene Termmodell ist extensional.*

Beweis. Seien $\overline{t}, \overline{t'} \in \mathcal{T}[\![A \to B]\!]$. Dann gibt es $\Gamma \subset \Gamma^\infty$ mit $\Gamma \vdash t, t' : A \to B$. Sei $x \notin \mathsf{FV}(t, t')$ mit $x : A \in \Gamma^\infty$. So ein x existiert, da wir zu jedem Typ A unendlich viele Variablen in Γ^∞ haben. Es ist $\overline{x} \in \mathcal{T}[\![A]\!]$, und nach Voraussetzung sei $\mathcal{T}\mathsf{App}(\overline{t}, \overline{x}) = \mathcal{T}\mathsf{App}(\overline{t'}, \overline{x})$, also auch $\Gamma, x : A \vdash t\,x = t'\,x : B$. Mit ξ und η gilt nun $\Gamma \vdash t = t' : A \to B$, also $\overline{t} = \overline{t'}$. □

Satz 7.53 (Vollständigkeit des offenen Termmodells) *Sei $\Gamma \subset \Gamma^\infty$. Wenn $\Gamma \models_{\mathcal{T}} t = t' : C$, dann $\Gamma \vdash t = t' : C$.*

Beweis. Die Identitätssubstitution $\sigma_0(x) = x$ hat die Typisierung $\Gamma \vdash \sigma : \Gamma$ für jedes Γ. Ausserdem gilt $\mathcal{T}[\![\Gamma \vdash t : C]\!]_{\overline{\sigma_0}} = \overline{t\sigma_0} = \overline{t}$ für alle $\Gamma \vdash t : C$, insofern $\Gamma \subset \Gamma^\infty$. Da nach Annahme die Gleichung $t = t'$ in $\mathcal{T}$ gültig ist, also insbesondere $\mathcal{T}[\![\Gamma \vdash t : C]\!]_{\overline{\sigma_0}} = \mathcal{T}[\![\Gamma \vdash t' : C]\!]_{\overline{\sigma_0}}$, gilt $\overline{t}^C = \overline{t'}^C$ und damit $\Gamma \vdash t = t' : C$. □

7.7.3 Schwache Normalisierung

Eine interessante Anwendung von Termmodellen ist Normalisierung. Im folgenden zeigen wir, dass jeder wohlgetypte Term eine $\beta\eta$-Normalform hat.

Definition 7.54 *Ein einfach getypter β-normaler Term t ist η-lang, falls jede wohlgetypte η-Expansion von t die β-Normalität zerstört.*

Beispiel 7.55

1. *$x : o \to o$ ist nicht η-lang, kann zu $\lambda y : o.\, x\, y$ expandiert werden.*
2. *$\lambda y : o.\, x\, y : o \to o$ ist η-lang, da die einzig wohlgetypte η-Expansion $\lambda y : o.\, (\lambda z : o.\, x\, z)\, y$ einen β-Redex besitzt.*
3. *$\lambda y : o \to o.\, x\, y : (o \to o) \to o$ ist nicht η-lang, aber seine Expansion $\lambda y : o \to o.\, x\, (\lambda z : o.\, y\, z)$.*

Definition 7.56 (Induktive Definition der $\beta\eta$-Normalformen) *Wir definieren induktiv die beiden Urteile $\Gamma \vdash t \Uparrow A$, "im Kontext Γ ist t eine lange $\beta\eta$-Normalform vom Typ A", und $\Gamma \vdash t \Downarrow A$, "im Kontext Γ ist t eine neutrale lange $\beta\eta$-Normalform vom Typ A".*

$$\frac{}{\Gamma \vdash x \Downarrow \Gamma(x)} \qquad \frac{\Gamma \vdash r \Downarrow A \to B \qquad \Gamma \vdash s \Uparrow A}{\Gamma \vdash r\, s \Downarrow B}$$

$$\frac{\Gamma \vdash t \Downarrow o}{\Gamma \vdash t \Uparrow o} \qquad \frac{\Gamma, x\!:\!A \vdash t \Uparrow B}{\Gamma \vdash \lambda x\!:\!A.\, t \Uparrow A \to B}$$

Wir setzen $\mathsf{Nf}^A = \{t \mid \Gamma \vdash t \Uparrow A \text{ für ein } \Gamma \subset \Gamma^\infty\}$ und $\mathsf{Ne}^A = \{t \mid \Gamma \vdash t \Downarrow A \text{ für ein } \Gamma \subset \Gamma^\infty\}$.

Es gilt $\mathsf{Nf}^o = \mathsf{Ne}^o$ für Grundtypen o.

Definition 7.57 (Substruktur für Normalisierung) *Wir definieren*

$$\begin{array}{lcl} \mathcal{N}[\![o]\!] & = & \{\overline{t} \in \mathcal{T}[\![o]\!] \mid t \in \mathsf{Ne}^o\} \\ \mathcal{N}[\![A \to B]\!] & = & \{\overline{r} \in \mathcal{T}[\![A \to B]\!] \mid \mathcal{T}\mathsf{App}(\overline{r}, \overline{s}) \in \mathcal{N}[\![B]\!] \text{ für alle } \overline{s} \in \mathcal{N}[\![A]\!]\} \end{array}$$

Trivialerweise folgt aus der Definition die Wohldefiniertheit von $\mathcal{T}\mathsf{App} \in \mathcal{N}[\![A \to B]\!] \times \mathcal{N}[\![A]\!] \to \mathcal{N}[\![B]\!]$ auf der Substruktur $\mathcal{N}$.

Lemma 7.58 $\overline{\mathsf{Ne}^C} \subseteq \mathcal{N}[\![C]\!] \subseteq \overline{\mathsf{Nf}^C}$ *für alle Typen C.*

Beweis. Durch Induktion über C. Im Fall $C = o$ gilt das schon nach Definition. Sei $C = A \to B$ und $r \in \mathsf{Ne}^{A \to B}$. Um $\overline{r} \in \mathcal{N}[\![A \to B]\!]$ zu zeigen, nehmen wir ein beliebiges $\overline{s} \in \mathcal{N}[\![A]\!]$ und folgern $\mathcal{T}\mathsf{App}(\overline{r}, \overline{s}) \in \mathcal{N}[\![B]\!]$. Nach I.H. für A ist $s \in \mathsf{Nf}^A$, damit $r\, s \in \mathsf{Ne}^B$, und wir haben $\overline{r\, s} \in \mathcal{N}[\![B]\!]$ mach I.H. für B.

Sei nun $\overline{r} \in \mathcal{N}[\![A \to B]\!]$. Damit ist $\Gamma \vdash r : A \to B$ für ein $\Gamma \subset \Gamma^\infty$. Sei nun $x \notin \mathsf{FV}(r)$ mit $x\!:\!A \in \Gamma^\infty$, dann ist $\Gamma, x\!:\!A \vdash x \Downarrow A$ und $\overline{x} \in \mathcal{N}[\![A]\!]$ nach I.H. für A. Also ist $\mathcal{T}\mathsf{App}(\overline{r}, \overline{x}) \in \mathcal{N}[\![B]\!]$ nach I.H. für B, was $\Gamma, x\!:\!A \vdash t = r\, x : B$ impliziert für ein $\Gamma, x : A \vdash t \Uparrow B$. Weiter gilt mit ξ und η die Gleichung $\Gamma \vdash \lambda x\!:\!A.\, t = r : A \to B$ und wir haben $\Gamma \vdash \lambda x\!:\!A.\, t \Uparrow A \to B$. Damit ist schliesslich $\overline{r} = \overline{\lambda x\!:\!A.\, t} \in \overline{\mathsf{Nf}^{A \to B}}$. □

Lemma 7.59 (Wohldefiniertheit der Interpretation) $\mathcal{T}[\![\Gamma \vdash t : C]\!] \in \mathcal{N}[\![\Gamma]\!] \to \mathcal{N}[\![C]\!]$.

Beweis. Durch Induktion über $\Gamma \vdash t : C$.

Fall $\mathcal{T}[\![\Gamma \vdash x : \Gamma(x)]\!]_{\overline{\sigma}} = \overline{\sigma}(x) \in \mathcal{N}[\![\Gamma(x)]\!]$, da $\overline{\sigma} \in \mathcal{N}[\![\Gamma]\!]$.

Fall $\mathcal{T}[\![\Gamma \vdash r\, s : B]\!]_{\overline{\sigma}} = \mathcal{T}\mathsf{App}^{A,B}(\mathcal{T}[\![\Gamma \vdash r : A \to B]\!]_{\overline{\sigma}}, \mathcal{T}[\![\Gamma \vdash s : A]\!]_{\overline{\sigma}}) \in \mathcal{N}[\![B]\!]$ unter Verwendung der I.H. für r und s.

Fall Um $\mathcal{T}[\![\Gamma \vdash \lambda x\!:\!A.\, t : A \to B]\!]_{\overline{\sigma}} \in \mathcal{N}[\![A \to B]\!]$ zu zeigen, wähle $\overline{s} \in \mathcal{N}[\![A]\!]$ beliebig und zeige $\mathcal{T}\mathsf{App}^{A,B}(\mathcal{T}[\![\Gamma \vdash \lambda x\!:\!A.\, t : A \to B]\!]_{\overline{\sigma}}, \overline{s}) \in \mathcal{N}[\![B]\!]$. Weil $\overline{\sigma'} := \overline{\sigma}[x \mapsto \overline{s}] \in [\![\Gamma, x\!:\!A]\!]$, gilt nach I.H. $\mathcal{T}[\![\Gamma, x\!:\!A \vdash t : V]\!]_{\overline{\sigma'}} \in \mathcal{N}[\![B]\!]$. Wir sind fertig, da $\mathcal{T}$ ein Umgebungsmodell ist.

□

Im Beweis haben wir nur verwendet, dass $\mathcal{T}[\![\Gamma \vdash t : C]\!]$ die Gleichungen für ein Umgebungsmodell erfüllt. In Teil **??** werden wir das Lemma auf beliebige Umgebungsmodelle verallgemeinern.

Satz 7.60 *Jeder wohlgetypte Term hat eine lange $\beta\eta$-Normalform.*

Beweis. Für die Identitätssubstitution $\sigma_0(x) = x$ gilt $\mathcal{T}[\![\Gamma \vdash t : A]\!]_{\overline{\sigma_0}} = \overline{t}$, also gilt $\Gamma \vdash t = r : A$ für ein $\Gamma \vdash r \Uparrow A$ nach dem vorigen Lemma. □

7.8 Logische Relationen und Vollständigkeit des Standardmodells

7.8.1 Logische Prädikate und Relationen

Definition 7.61 (Logisches Prädikat) *Sei $\mathcal{A}$ eine applikative Struktur. Ein logisches Prädikat $\mathcal{P}$ ist eine Familie $\mathcal{P}[\![C]\!] \subseteq \mathcal{A}[\![C]\!]$, so dass*

$$f \in \mathcal{P}[\![A \to B]\!] \text{ gdw. } \mathcal{A}\mathsf{App}^{A,B}(f,a) \in \mathcal{P}[\![B]\!] \text{ für alle } a \in \mathcal{P}[\![A]\!].$$

Beispiel 7.62 *$\mathcal{N}$ aus Def. 7.57 ist ein logisches Prädikat.*

Lemma 7.63 (Basislemma für logische Prädikate) *Sei $\mathcal{A}$ ein Umgebungsmodell und $\mathcal{P}$ ein logisches Prädikat auf $\mathcal{A}$. Dann gilt $\mathcal{A}[\![\Gamma \vdash t : C]\!] \in \mathcal{P}[\![\Gamma]\!] \to \mathcal{P}[\![C]\!]$.*

Beweis. Dies ist die Verallgemeinerung von Lemma 7.59, Beweis analog durch Induktion über $\Gamma \vdash t : C$. □

Definition 7.64 (Produktstruktur) *Seien $\mathcal{A}, \mathcal{B}$ applikative Strukturen. Die Produktstruktur $\mathcal{A} \times \mathcal{B}$ ist punktweise definiert, also*

$$\begin{array}{lcl} (\mathcal{A} \times \mathcal{B})[\![C]\!] & = & \mathcal{A}[\![C]\!] \times \mathcal{B}[\![C]\!] \\ (\mathcal{A} \times \mathcal{B})\mathsf{App}((f,g),(a,b)) & = & (\mathcal{A}\mathsf{App}(f,a), \mathcal{B}\mathsf{App}(g,b)). \end{array}$$

Ebenso bilden wir das Produkt von Umgebungsmodellen $\mathcal{A}, \mathcal{B}$ durch die Setzungen

$$(\mathcal{A} \times \mathcal{B})[\![\Gamma \vdash t : C]\!]_\rho \quad = \quad (\mathcal{A}[\![\Gamma \vdash t : C]\!]_{\rho_1}, \mathcal{B}[\![\Gamma \vdash t : C]\!]_{\rho_2}),$$

wobei $\rho_1 \in \mathcal{A}[\![\Gamma]\!]$ und $\rho_2 \in \mathcal{B}[\![\Gamma]\!]$ eindeutig bestimmt sind durch $\rho(x) = (\rho_1(x), \rho_2(x))$ für alle $x \in \mathsf{dom}(\Gamma)$.

Leicht sieht man, dass das Produkt von Umgebungsmodellen wieder ein Umgebungsmodell ist. Dabei beachte man, dass $(\mathcal{A} \times \mathcal{B})[\![\Gamma]\!] = \{\rho \mid \rho(x) \in (\mathcal{A} \times \mathcal{B})[\![\Gamma(x)]\!] \text{ für alle } x \in \mathsf{dom}(\Gamma)\}$.

Definition 7.65 (Logische Relation) *Eine (binäre) logische Relation $\mathcal{R}$ auf applikativen Strukturen $\mathcal{A}$ und $\mathcal{B}$ ist eine logisches Prädikat auf der Produktstruktur $\mathcal{A} \times \mathcal{B}$.*

7.8.2 Vollständigkeit des Standardmodells

Ein Modell ist vollständig, wenn jede Gleichung, die im Modell gilt, auch herleitbar ist. Ein endliches Modell, also in dem die Basistypen durch endliche Mengen interpretiert werden, ist nicht vollständig. Setzen wir z.B. $[\![o]\!] = \{0,1\}$, dann gilt $[\![\lambda f : o \to o.\, \lambda x : o.\, f\,(f\,(f\,x))]\!] = [\![\lambda f : o \to o.\, \lambda x : o.\, f\,x]\!]$, d.h. die Sequenz der Church-Ziffern kollabiert.

Interpretieren wir jeden Grundtyp als unendliche Menge, dann haben wir (mehr als) genug Elemente, um $\beta\eta$-verschiedene λ-Terme auseinander zu halten. Mit dem vollen Funktionsraum des Standardmodells ist dann $[\![o \to o]\!]$ schon überabzählbar, es gibt aber nur abzählbar viele λ-Terme. Wir sondern die Elemente heraus, die Termen entsprechen, in dem wir eine partielle Surjektion von dem Standardmodell in das Termmodell konstruieren.

Definition 7.66 (Partielle Surjektion) *Eine Relation $R \subseteq S \times T$ heisst* partielle Surjektion von S nach T*, falls R eine surjektive partielle Funktion ist, d.h.*

1. *$R(a,b)$ und $R(a,b')$ implizieren $b = b'$, und*
2. *für jedes $b \in B$ gibt es ein $a \in A$ mit $R(a,b)$.*

Lemma 7.67 *Ist R eine partielle Surjektion von S nach T, so existiert ein Isomorphismus $\phi : S' \leftrightarrow T$ für eine Teilmenge $S' \subseteq S$.*

Beweis. Wir verwenden das Auswahlaxiom. Da R surjektiv ist, gibt es zu jedem $b \in T$ eine nichtleere Menge $\{a \in S \mid (a,b) \in R\}$. Es gibt nach dem Auswahlaxiom also eine Funktion $\phi^{-1} : T \to S$, so dass $(\phi^{-1}(b), b) \in R$ für alle $b \in T$. Sei S' das Bild von ϕ^{-1}. Da R eine partielle Funktion ist, ist ϕ^{-1} injektiv und besitzt eine Umkehrung $\phi : S' \to T$. □

Lemma 7.68 (Vererbung der Vollständigkeit) *Seien $\mathcal{A}, \mathcal{B}$ Umgebungsmodelle und $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$ eine logische Relation von partiellen Surjektionen $\mathcal{R}[\![C]\!] \subseteq \mathcal{A}[\![C]\!] \times \mathcal{B}[\![C]\!]$.*

1. *Falls $\Gamma \models_{\mathcal{A}} t = t' : C$, dann $\Gamma \models_{\mathcal{B}} t = t' : C$.*
2. *Ist $\mathcal{B}$ vollständig, so auch $\mathcal{A}$.*

Beweis. 1. Sei $\rho_{\mathcal{B}} \in \mathcal{B}[\![\Gamma]\!]$. Da $\mathcal{R}$ eine Surjektion ist, gibt es ein $\rho_{\mathcal{A}} \in \mathcal{A}[\![\Gamma]\!]$ mit $\mathcal{R}[\![\Gamma]\!](\rho_{\mathcal{A}}, \rho_{\mathcal{B}})$. Nach Annahme ist $\mathcal{A}[\![\Gamma \vdash t : C]\!]_{\rho_{\mathcal{A}}} = \mathcal{A}[\![\Gamma \vdash t' : C]\!]_{\rho_{\mathcal{A}}} =: a$. Sei $b := \mathcal{B}[\![\Gamma \vdash t : C]\!]_{\rho_{\mathcal{B}}}$ und $b' := \mathcal{B}[\![\Gamma \vdash t' : C]\!]_{\rho_{\mathcal{B}}}$. Nach dem Basislemma für logische Relationen gilt $\mathcal{R}[\![C]\!](a,b)$ und $\mathcal{R}[\![C]\!](a,b')$ und wegen der Funktionalität von $\mathcal{R}$ schliesslich $b = b'$.

2. folgt sofort aus 1. □

Satz 7.69 (Vollständigkeit des Standardmodells) *Falls $\mathcal{M}[\![o]\!]$ mindestens abzählbar unendlich für allen Grundtypen o und $\mathcal{M}[\![A \to B]\!] = \mathcal{M}[\![A]\!] \to \mathcal{M}[\![B]\!]$ der volle klassische Funktionsraum ist, dann ist $\mathcal{M}$ vollständig.*

Beweis. Sei $\Gamma \models_{\mathcal{M}} t = t' : C$ und $\Gamma^\infty \supset \Gamma$ ein unendlicher Kontext wie in Teil 7.7.2. Wir konstruieren durch Induktion über C eine logische Relation $\mathcal{R}$ zwischen dem Standardmodell $\mathcal{M}$ und dem offenen Termmodell $\mathcal{T}$ über Γ^∞.

Für jeden Grundtypen o sei $\mathcal{R}[\![o]\!]$ eine beliebige partielle Surjektion von $\mathcal{M}[\![o]\!]$ nach $\mathcal{T}[\![o]\!]$. Diese existiert aus Kardinalitätserwägungen, da $\mathcal{M}[\![o]\!]$ eine

isomorphe Kopie von $\mathbb{N}$ enthält und es eine Surjektion $\mathbb{N} \to \mathcal{T}[\![o]\!]$ gibt, z.B. eine Aufzählung aller Terme.

Im Fall $C = A \to B$ existieren nach Induktionsvoraussetzung partielle Surjektionen $\mathcal{R}[\![A]\!]$ und $\mathcal{R}[\![B]\!]$ und nach Lemma 7.67 die zugehörigen Isormorphismen ϕ_A und ϕ_B. Wir "konstruieren" nun zu jedem $r \in \mathcal{T}[\![A \to B]\!]$ ein $f_r \in \mathcal{M}[\![A \to B]\!]$ mittels

$$f_r(a) = \begin{cases} \phi_B^{-1}(\mathcal{T}\mathsf{App}^{A,B}(r, \phi_A(a))) & \text{falls } a \in \mathsf{dom}(\phi_A) \\ \text{irgendein } b \in \mathcal{M}[\![B]\!] & \text{andernfalls.} \end{cases}$$

Die Relation $\mathcal{R}[\![A \to B]\!] = \{(f_r, r) \mid r \in \mathcal{T}[\![A \to B]\!]\}$ ist surjektiv. Ist nun $f_r = f_{r'}$ also insbesondere $f_r(a) = f_{r'}(a)$ für alle $a \in \mathsf{dom}(\phi_A)$, so folgt auch $\mathcal{T}\mathsf{App}(r, \phi_A(a)) = \mathcal{T}\mathsf{App}(r', \phi_A(a))$, und weil das Bild von ϕ_A ganz $\mathcal{T}[\![A]\!]$ ist, mit Extensionalität auch $r = r'$. Die Relation $\mathcal{R}[\![A \to B]\!]$ ist damit auch eine partielle Funktion. □

Der Beweis verwendet keine besonderen Eigenschaften des Termmodells, ausser Extensionalität, und dass die Interpretationen Basistypen abzählbare Mengen sind.

Kapitel 8

Modelle des getypten Lambda-Kalküls

TODO: Mengentheoretische Interpretation einfacher mit getypten Konstanten $\mathsf{Pair}_{A,B}$ ....

Ein Modell besteht aus

1. einer Interpretation (Menge) $[\![A]\!]$ für jeden Typen $A \in \mathsf{Ty}$,
2. einer Interpretation $[\![t]\!]_\rho$ für jeden wohlgetypten Term t, wobei ρ eine Interpretation (Belegung) der freien Variablen von t liefert,
3. einem Gleichheitsbegriff auf $\mathsf{D} \supseteq \bigcup_{A \in \mathsf{Ty}} [\![A]\!]$.

Wünschenswerte Eigenschaften:

1. *Die Interpretation eines Terms ist durch die Interpretation seiner freien Variablen eindeutig bestimmt:* Die Interpretation $[\![t]\!]_\rho$ hängt nur von der Belegung $\rho(x)$ der in t freien Variablen x ab.
2. *Die Interpretation wohlgetypter Terme ist wohldefiniert:* Ist $\Gamma \vdash t : A$ und $\rho(x) \in [\![\Gamma(x)]\!]$ für alle $x \in \mathsf{FV}(t)$, dann $[\![t]\!]_\rho \in [\![A]\!]$.
3. *Gleiche Terme haben gleiche Interpretation:* Sind t und t' von gleichen Typ und ist $t =_{\beta\eta} t'$, dann gilt $[\![t]\!]_\rho = [\![t']\!]_\rho$.

Beispiele von Modellen:

1. Maschinenprogramme. Die Interpretation ist dann Compilation. Die Gleichheit ist dann *beobachtbare* Gleichheit.
2. Berechenbare Funktionen. Z.B. Turing-Programme oder partiell rekursive Funktionen auf den natürlichen Zahlen.
3. Geschlossene λ-Terme.

4. Stark normalisierende Terme.
5. Terme modulo $=_\beta$ oder $=_{\beta\eta}$.
6. Mengentheoretische Funktionen.
7. Domaintheoretische Funktionen.

8.1 Modellbegriff für STL

Einfach getypter λ-Kalkül mit Konstanten.

$$\begin{array}{lll} c & ::= & \mathsf{pair} \mid \mathsf{fst} \mid \mathsf{snd} \mid \mathsf{inl} \mid \mathsf{inr} \mid \mathsf{case} \\ r, s, t & ::= & c \mid x \mid \lambda x^A t \mid r\, s \end{array}$$

Die Konstanten haben folgenden Typ für alle A, B, C.

$$\begin{array}{lcl} \mathsf{pair} & : & A \to B \to A \times B \\ \mathsf{fst} & : & A \times B \to A \\ \mathsf{snd} & : & A \times B \to B \\ \mathsf{inl} & : & A \to A + B \\ \mathsf{inr} & : & B \to A + B \\ \mathsf{case} & : & A + B \to (A \to C) \to (B \to C) \to C \end{array}$$

Für jeden einfachen Typen $A \in \mathsf{Ty}$ nehmen wir eine Menge $[\![A]\!]$, außerdem eine Menge D. Wir nehmen folgende Funktionen an:

$$\mathsf{App} \quad \in \quad \mathsf{D} \times \mathsf{D} \to \mathsf{D}$$

Ausserdem sei $[\![c]\!] \in \mathsf{D}$ für jede Konstante c.

Wir schreiben $d \in M \mapsto f(d)$ für die Funktion $g \in M \to \mathsf{D}$ mit $g(d) = f(d)$. Für eine Umgebung $\rho \in \mathsf{V} \to \mathsf{D}$, eine Variable x und ein Element $d \in \mathsf{D}$ setzen wir $\rho[x \mapsto d]$ für die Umgebung ρ' mit $\rho'(x) = d$ und $\rho'(y) = \rho(y)$ für $y \neq x$.

Für einen Term t und eine Umgebung $\rho \in \mathsf{V} \to \mathsf{D}$ definieren wir die Interpretation $[\![t]\!]_\rho$ durch Rekursion über t.

$$\begin{array}{rcll} [\![c]\!]_\rho & = & [\![c]\!] & \\ [\![x]\!]_\rho & = & \rho(x) & \\ [\![r\, s]\!]_\rho & = & \mathsf{App}([\![r]\!]_\rho, [\![s]\!]_\rho) & \\ [\![\lambda x^A t]\!]_\rho & = & [\![\lambda x^A t']\!]_{\rho'} & \text{falls } [\![t]\!]_{\rho[x\mapsto d]} = [\![t']\!]_{\rho'[x\mapsto d]} \text{ für alle } d \in \mathsf{D} \end{array}$$

Lemma 8.1 *Wenn $\rho(x) = \rho'(x)$ für alle $x \in \mathsf{FV}(t)$, dann $[\![t]\!]_\rho = [\![t]\!]_{\rho'}$.*

Beweis. Durch Induktion über t. □

Lemma 8.2 (Substitution) $[\![t[s/x]]\!]_\rho = [\![t]\!]_{\rho[x\mapsto [\![s]\!]_\rho]}$.

Beweis. Durch Induktion über t. □

8.2 Wohldefiniertheit der Interpretation

Sei $\rho \in [\![\Gamma]\!]$ definiert als $\rho(x) \in [\![A]\!]$ für alle $(x\!:\!A) \in \Gamma$.
Für alle Typen A, B, C gelte $[\![A]\!] \subseteq \mathsf{D}$ und

$$\begin{array}{lcll} [\![\lambda x^A t]\!]_\rho & \in & [\![A \to B]\!] & \text{falls } [\![t]\!]_{\rho[x \mapsto d]} \in [\![B]\!] \text{ für alle } d \in [\![A]\!] \\ \mathsf{App}(f, d) & \in & [\![B]\!] & \text{falls } f \in [\![A \to B]\!] \text{ und } d \in [\![A]\!] \\ \\ [\![c]\!] & \in & [\![C]\!] & \text{falls } c : C \end{array}$$

Satz 8.3 *Wenn $\Gamma \vdash t : C$ und $\rho \in [\![\Gamma]\!]$, dann $[\![t]\!]_\rho \in [\![C]\!]$.*

Beweis. Durch Induktion über $\Gamma \vdash t : C$. □

8.2.1 Korrektheit von $\beta\eta$

Definiere folgende Operationen auf D.

$$\begin{array}{lcl} \mathsf{Pair}(d, e) & = & \mathsf{App}(\mathsf{App}([\![\mathsf{pair}]\!], d), e) \\ \mathsf{Fst}(d) & = & \mathsf{App}([\![\mathsf{fst}]\!], d) \\ \mathsf{Snd}(d) & = & \mathsf{App}([\![\mathsf{snd}]\!], d) \\ \mathsf{Inl}(d) & = & \mathsf{App}([\![\mathsf{inl}]\!], d) \\ \mathsf{Inr}(d) & = & \mathsf{App}([\![\mathsf{inr}]\!], d) \\ \mathsf{Case}(d, f, g) & = & \mathsf{App}(\mathsf{App}(\mathsf{App}([\![\mathsf{case}]\!], d), f), g) \end{array}$$

Für alle Typen A, B, C und alle $d \in [\![A]\!]$, $e \in [\![B]\!]$, $f \in [\![A \to C]\!]$ und $g \in [\![B \to C]\!]$ sollen folgende Gesetze gelten.

$$\mathsf{App}([\![\lambda x^A t]\!]_\rho, d) \;=\; [\![t]\!]_{\rho[x \mapsto d]} \quad \text{für alle } d \in [\![A]\!]$$

$$\begin{array}{lcl} \mathsf{Fst}(\mathsf{Pair}(d, e)) & = & d \\ \mathsf{Snd}(\mathsf{Pair}(d, e)) & = & e \\ \mathsf{Case}(\mathsf{Inl}(d), f, g) & = & f(d) \\ \mathsf{Case}(\mathsf{Inr}(e), f, g) & = & g(e) \end{array}$$

Satz 8.4 *Wenn $\Gamma \vdash t, t' : C$ und $t =_{\beta\eta} t'$ und $\rho \in \Gamma$, dann $[\![t]\!]\rho = [\![t']\!]\rho$.*

8.3 Mengentheoretische Interpretation

Die Interpretation $[\![a]\!]$ der Grundtypen a sei vorgegeben.

$$\begin{array}{lcl} [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\ [\![A \times B]\!] & = & [\![A]\!] \times [\![B]\!] \\ [\![A + B]\!] & = & \{(0, d), (1, e) \mid d \in [\![A]\!], e \in [\![B]\!]\} \end{array}$$

Sei $\bot$ ein ausgezeichnetes Element und $\mathsf{D} = \{\bot\} \uplus \biguplus_{A \in \mathsf{Ty}} [\![A]\!]$. Für eine Funktion $f : D \to E$ und $[\![A]\!] \subseteq D$, $[\![B]\!] \subseteq E$ schreiben wir $f \in [\![A]\!] \to [\![B]\!]$ falls $f(d) \in [\![B]\!]$ für alle $d \in [\![A]\!]$.

$$
\begin{array}{lcl}
[\![\lambda x^A t]\!]_\rho & = & \mathsf{Lam}^A(d \in [\![A]\!] \mapsto [\![t]\!]_{\rho[x \mapsto d]}) \\
\mathsf{Lam}^A & : & ([\![A]\!] \to \mathsf{D}) \to \mathsf{D} \\
\mathsf{Lam}^A(f) & = & \begin{cases} f & \text{falls es } B \text{ gibt mit } f \in [\![A]\!] \to [\![B]\!] \\ \bot & \text{sonst} \end{cases} \\
\mathsf{App} & : & \mathsf{D} \times \mathsf{D} \to \mathsf{D} \\
\mathsf{App}(f, d) & = & \begin{cases} f(d) & \text{falls definiert} \\ \bot & \text{sonst} \end{cases} \\
[\![\mathsf{pair}]\!] & : & \bigcup_{A,B} [\![A \to B \to A \times B]\!] \\
[\![\mathsf{pair}]\!](d)(e) & = & (d, e) \\
[\![\mathsf{fst}]\!] & : & \bigcup_{A,B} [\![A \times B \to A]\!] \\
[\![\mathsf{fst}]\!](d, e) & = & d \\
[\![\mathsf{snd}]\!] & : & \bigcup_{A,B} [\![A \times B \to B]\!] \\
[\![\mathsf{snd}]\!](d, e) & = & e \\
[\![\mathsf{inl}]\!] & : & \bigcup_{A,B} [\![A \to A + B]\!] \\
[\![\mathsf{inl}]\!](d) & = & (0, d) \\
[\![\mathsf{inl}]\!] & : & \bigcup_{A,B} [\![B \to A + B]\!] \\
[\![\mathsf{inr}]\!](e) & = & (1, e) \\
[\![\mathsf{case}]\!] & : & \bigcup_{A,B,C} [\![A + B \to (A \to C) \to (B \to C) \to C]\!] \\
[\![\mathsf{case}]\!](0, d)(f)(g) & = & f(d) \\
[\![\mathsf{case}]\!](1, e)(f)(g) & = & g(e)
\end{array}
$$

Die ξ-Regel gilt für $[\![\lambda x^A t]\!]_\rho$.

Für alle Typen A, B, C gilt

$$
\begin{array}{lcll}
\mathsf{Lam}^A & \in & ([\![A]\!] \to [\![B]\!]) \to [\![A \to B]\!] & \\
\mathsf{App} & \in & [\![A \to B]\!] \times [\![A]\!] \to [\![B]\!] & \\
[\![c]\!] & \in & [\![C]\!] & \text{falls } c : C
\end{array}
$$

8.4 Bereichstheoretische Interpretation

Im λ-Kalkül mit Fixpunkten gibt es nicht-terminierende Berechnungen, die mit $\bot$ modelliert werden sollen. Eine partielle Funktion $f : D \rightharpoonup E$ kann als totale Funktion $f' : D \rightharpoonup E_\bot$ verstanden werden wobei $f'(d) = \bot$ gdw. $f(d)$ undefiniert. Zwischen partiellen Funktionen gibt es eine natürliche partielle Ordnung. Wir können sagen g ist *mehr definiert* als f, in Zeichen $f \sqsubseteq g$, falls $g(d) \neq \bot$ impliziert, dass $f(d) = g(d)$. Die total undefinierte Funktion können wir mit $\bot$ identifizieren, also $\bot \sqsubseteq f$. Damit haben wir eine partielle Ordnung mit kleinstem Element.

Nun gibt es auch Funktionen höherer Ordnung, z.B. $f : (D \to E) \to (D \to E)$. Definitions- und Wertebereich dieser Funktion sind partielle Ordnungen. Es bietet sich also an, gleich jeden *Bereich (domain)* als partielle Ordnung anzunehmen. Grundtypen wie $\mathbb{N}$ können dabei als diskrete Ordnungen angenommen werden, d.h. $n \sqsubseteq n'$ gdw. $n = n'$. Ist das Ergebnis einer partiellen Funktion eine natürliche Zahl, so kann $\mathbb{N}_\perp = \{\perp\} \cup \mathbb{N}$ als Wertebereich genommen werden, mit der *flachen Ordnung* $n \sqsubseteq n'$ gdw. $n = \perp$ oder $n = n'$.

Jede Funktion ist *monoton*, d.h. $d \sqsubseteq d'$ impliziert $f(d) \sqsubseteq f(d')$, denn es ist klar, dass das Ergebnis einer Funktion nicht weniger definiert sein kann, wenn das Argument mehr definiert ist. Man kann ja keine Funktion implementieren, die nicht terminiert, wenn die Berechnung ihres Argument terminiert, aber terminiert, wenn die Berechnung ihres Argumentes nicht terminiert (Halteproblem!).

Die Semantik einer rekursiven Funktion $f = F(f)$, oder allgemeiner, eines Fixpunktes $\mathsf{fix}\, F$, gewinnt man durch *Approximation*. Man beginnt mit einer ersten Näherung $f_0 = \perp$. Die zweite Näherung $f_1 = F(f_0)$ ist trivialerweise nicht schlechter als die erste, also $f_0 \sqsubseteq f_1$. Die dritte Näherung $f_2 = F(f_1)$ ist wieder mindestens so definiert wie die zweite, $f_1 \sqsubseteq f_2$, weil F monoton ist. Treibt man das Spiel weiter, erhält man eine aufsteigende Kette $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$. Wird sie stationär, d.h. $f_i = f_{i+1}$ für ein f_i, so ist der Fixpunkt erreicht. Nicht jede Kette wird jedoch stationär. Betrachten wir eine Kette von Funktionen $f_i : \mathbb{N} \to \mathbb{N}_\perp$, wobei die i-te Funktion genau für die Zahlen $< i$ definiert ist, z.B. $f_i(j) = j$ falls $j < i$ und $f_i(j) = \perp$ sonst. Nun gilt $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$, die Kette wird jedoch nicht stationär. Trotzdem konvergiert die Kette, und zwar gegen die totale Identitätsfunktion f auf den natürlichen Zahlen. Diese Funktion f entsteht als Fixpunkt des Funktionals $F(f)(n) = 0$ falls $n = 0$ und $= 1+f(n-1)$ sonst. Damit solche Fixpunkte existieren, verlangen wir, dass jeder Bereich D Limiten für aufsteigende Ketten enthält, also $f_\omega := \bigsqcup_{i\in\mathbb{N}} f_i \in D$ falls $f_i \sqsubseteq f_{i+1}$ für alle $i \in \mathbb{N}$. Dies ist jedoch nur eine notwendige Bedingung für die Existenz eines Fixpunktes. Die Kette könnte sich ja transfinit fortsetzen: $\perp \sqsubseteq F(\perp) \sqsubseteq F(F(\perp)) \ldots \sqsubseteq f_\omega \sqsubseteq F(f_\omega) \sqsubseteq F(F(f_\omega)) \sqsubseteq \ldots$ Deswegen fordern wir noch $F(f_\omega) = f_\omega$, was sich allgemeiner schreiben lässt als $F(\bigsqcup_{i\in\mathbb{N}} f_i) = \bigsqcup_{i\in\mathbb{N}} F(f_i)$, also F erhält Limiten von aufsteigenden Ketten. Zusammengefasst existieren in Bereichen also Fixpunkte von monotonen, Limes-erhaltenden Funktionalen (oder Funktionen) F. Solche Fs nennen wir *stetig*.

Im Folgenden zeigen wir, dass Bereiche ein Modell des einfach getypten λ-Kalküls sind. Insbesondere müssen wir zeigen, dass alle Operationen, wie Abstraktion, Application etc. stetig sind.

Definition 8.5 (POSET, CPO, PCPO) *Eine* partielle Ordnung (partial ordered set, POSET) *ist eine Menge D mit einer reflexiven, transitiven und antisymmetrischen Relation $\sqsubseteq_D \subseteq D \times D$. Eine partielle Ordnung heisst* vollständig (complete partial order, CPO*, falls für jede Kette $d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_i \sqsubseteq d_{i+} \sqsubseteq \ldots$ ein Supremum $\bigsqcup_{i\in\mathbb{N}} d_i \in D$ existiert. Eine* partielle Ordnung mit kleinstem Element (pointed CPO, PCPO) *hat ein eindeutiges Element $\perp$ mit $\perp \sqsubseteq d$ für alle $d \in D$.*

Definition 8.6 (Strikte Funktion) *Seien D, E PCPOs. Eine Funktion $f : D \to E$ heisst strikt, falls $f(\bot_D) = \bot_E$.*

8.4.1 Konstruktion von Bereichen

Diskrete CPO Ist M eine Menge, dann auch eine CPO mit der diskreten Ordnung $d \sqsubseteq_M d'$ gdw. $d = d'$.

Lifting Ist D eine CPO und $\bot \notin D$, so ist $D_\bot$ eine PCPO mit $d \sqsubseteq_{D_\bot} d'$ gdw. $d = \bot$ oder $d \sqsubseteq_D d'$.

Unlifting Ist D eine PCPO, dann ist $D^\downarrow = D \setminus \{\bot\}$ eine CPO mit $d \sqsubseteq_{D^\downarrow} d'$ gdw. $d \sqsubseteq_D d'$.

Kartesisches Produkt Sind D, E CPOs, so ist $D \times E$ eine CPO mit $(d, e) \sqsubseteq_{D\times E} (d', e')$ gdw. $d \sqsubseteq_D d'$ und $e \sqsubseteq_E e'$. Paarbildung ist nicht-strikt.

Smash Product Sind D, E PCPOs, so ist $D \otimes E = (D^\downarrow \times E^\downarrow)_\bot$ eine PCPO. Paarbildung und Projektionen sind strikt.

Disjunkte Summe Sind D_i CPOs für $i \in I$, so ist $D = \sum_{i\in I} D_i$ eine CPO mit $(i, d) \sqsubseteq_D (i', d')$ gdw. $i = i'$ und $d \sqsubseteq_{D_i} d'$. Injektionen sind nicht strikt.

Coalesced Sum Sind D_i PCPOs für $i \in I$, so ist $\bigoplus_{i\in I} D_i = (\sum_{i\in I} D_i^\downarrow)_\bot$ eine PCPO. Injektionen und Fallunterscheidung sind strikt.

Funktionsraum Ist D eine Menge und E eine PCPO, dann ist der Raum $D \to E$ der Funktionen von D nach E eine PCPO. Die Ordnung ist punktweise, d.h., $f \sqsubseteq_{D\to E} f'$ gdw. $f(d) \sqsubseteq_E f'(d')$ für alle $d \sqsubseteq_D d'$. Damit ist $\bot_{D\to E}$ die Funktion konstant $\bot_E$. Wir definieren das Supremum auch punktweise, d.h. $(\bigsqcup i \in \mathbb{N} f_i)(d) = \bigsqcup i \in \mathbb{N}(f_i(d))$. Da E vollständig ist, gilt das auch für $D \to E$.

Stetige Funktionen Ist D eine CPO und E eine PCPO, dann ist der Raum $[D \to E]$ der stetigen Funktionen von D nach E eine PCPO. Ordnung und Supremum sind definiert wie beim normalen Funktionsraum. Es bleibt zu zeigen, dass das Supremum einer Kette stetiger Funktionen wieder stetig ist.

Stetige strikte Funktionen Sind D, E PCPOs, dann ist der Raum $[D \multimap E]$ der stetigen, strikten Funktionen eine PCPO.

Abhängige Funktionen Sind D_i PCPOs für $i \in I$, so ist $\prod_{i\in I} D_i = \{f \mid f(i) \in D_i\}$ eine PCPO. Ordnung und Supremum sind punktweise definiert, und die Funktion $f(i) = \bot_{D_i}$ ist das kleinste Element.

8.4.2 Operationen in Bereichen

Let D, E, F be PCPOs. We define the following functions:

$$
\begin{array}{lcll}
\mathrm{pair}_{D,E} & : & [D \to [E \to D \otimes E]] & \\
\mathrm{pair}_{D,E}(\bot) & = & \bot & \\
\mathrm{pair}_{D,E}(d)(\bot) & = & \bot & \text{falls } d \neq \bot \\
\mathrm{pair}_{D,E}(d)(e) & = & (d,e) & \text{falls } d, e \neq \bot \\
 & & & \\
\mathrm{fst}_{D,E} & : & [D \otimes E \to D] & \\
\mathrm{fst}_{D,E}(\bot) & = & \bot & \\
\mathrm{fst}_{D,E}(d,e) & = & d & \\
 & & & \\
\mathrm{snd}_{D,E} & : & [D \otimes E \to E] & \\
\mathrm{snd}_{D,E}(\bot) & = & \bot & \\
\mathrm{snd}_{D,E}(d,e) & = & e & \\
 & & & \\
\mathrm{inl}_{D,E} & : & [D \to D \oplus E] & \\
\mathrm{inl}_{D,E}(\bot) & = & \bot & \\
\mathrm{inl}_{D,E}(d) & = & (0,d) & \\
 & & & \\
\mathrm{inr}_{D,E} & : & [E \to D \oplus E] & \\
\mathrm{inr}_{D,E}(\bot) & = & \bot & \\
\mathrm{inr}_{D,E}(e) & = & (1,e) & \\
 & & & \\
\mathrm{case}_{D,E,F} & : & [D \oplus E \to [[D \to F] \to [[E \to F] \to F]]] & \\
\mathrm{case}_{D,E,F}(\bot) & = & \bot & \\
\mathrm{case}_{D,E,F}(0,d)(f)(g) & = & f(d) & \\
\mathrm{case}_{D,E,F}(1,e)(f)(g) & = & g(e) &
\end{array}
$$

Lemma 8.7 *Diese Funktionen sind wohldefiniert, insbesondere, stetig.*

Iteration von Funktionen. Ist $f \in D \to D$, so definieren wir die n-te Iterierte f^n von f wie folgt:

$$
\begin{array}{lcl}
f^0(d) & = & d \\
f^{n+1}(d) & = & f(f^n(d))
\end{array}
$$

Lemma 8.8 (Eigenschaften der Iteration)

1. *(Stetigkeit der Iterierten:) Wenn $f \in [D \to D]$, dann $f^n \in [D \to D]$.*
2. *(Monotonie der Iteration:) Sind $f, g \in D \to D$ monoton und $f \sqsubseteq g$, dann $f^n \sqsubseteq g^n$.*
3. *(Iteration erhält Suprema:) Sind $f_i \in [D \to D]$ für $i \in \mathbb{N}$ und $f_0 \sqsubseteq f_1 \sqsubseteq \ldots$, dann $(\bigsqcup_i f_i)^n = \bigsqcup_i f_i^n$.*

Beweis. Jeweils durch Induktion über n. Der Induktionsanfang $n = 0$ ist immer trivial. Wir zeigen jeweils den Induktionsschritt.

1. Sei $d \sqsubseteq d'$. Dann $f^{n+1}(d) = f(f^n(d)) \sqsubseteq f(f^n(d')) = f^{n+1}(d')$ mit Ind.hyp. und Monotonie von f. Dass f^n Suprema erhält, ist gleichermaßen einfach.
2. Für bel. d, $f^{n+1}(d) = f(f^n(d)) \sqsubseteq f(g^n(d)) \sqsubseteq g^{n+1}(d)$ mit Ind.hyp., Monotonie von f und $f \sqsubseteq g$.
3. Da Iteration monoton, gilt $f_i^{n+1} \sqsubseteq (\bigsqcup_i f_i)^{n+1}$, und damit $\bigsqcup_i f_i^{n+1} \sqsubseteq (\bigsqcup_i f_i)^{n+1}$. Die Rückrichtung ist etwas schwieriger.

$$\begin{array}{lcl}(\bigsqcup_i f_i)^{n+1}(d) & = & \bigsqcup_i f_i((\bigsqcup_i f_i)^n(d)) \\ & = & \bigsqcup_i f_i(\bigsqcup_j f_j^n(d)) \\ & = & \bigsqcup_{i,j} f_i(f_j^n(d)) \\ & \sqsubseteq & \bigsqcup_{i,j} f_{\max(i,j)}(f^n_{\max(i,j)}(d)) \\ & = & \bigsqcup_m f_m^{n+1}(d)\end{array}$$

   Hier haben wir benutzt, dass die f_i eine Kette bilden.

□

Interpretation von Fixpunkten.

$$\begin{array}{lcl}\mathrm{fix}_D & : & [D \to D] \to D \\ \mathrm{fix}_D(f) & = & \bigsqcup_{n\in\mathbb{N}} f^n(\bot)\end{array}$$

Lemma 8.9 (Eigenschaften der Fixpunkt-Operation)

1. fix_D *ist wohldefiniert.*
2. *Wenn* $f \in [D \to D]$, *dann* $f(\mathrm{fix}(f)) = \mathrm{fix}(f)$.
3. *Die Operation* fix *ist stetig, also* $\mathrm{fix}_D \in [[D \to D] \to D]$.

Beweis. Wegen Monotonie von f gilt $\bot = f^0(\bot) \sqsubseteq f^1(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \ldots$, also existiert der Limes $x = \bigsqcup_i f^i(\bot)$. Da f stetig, gilt $f(x) = f(\bigsqcup_i f^i(\bot)) = \bigsqcup_i f(f^i(\bot)) = x$. Die Stetigkeit der Fixpunkt-Operation folgt aus der Stetigkeit der Iteration. □

8.4.3 Interpretation des STL

Jeder Grundtyp a wird interpretiert als PCPO $[\![a]\!]$. Insbesondere wird $\mathcal{N}$ interpretiert als $\mathbb{N}_\bot$.

$$\begin{array}{lcl}[\![A \to B]\!] & = & [[\![A]\!] \to [\![B]\!]] \\ [\![A \times B]\!] & = & [\![A]\!] \otimes [\![B]\!] \\ [\![A + B]\!] & = & [\![A]\!] \oplus [\![B]\!]\end{array}$$

Nun ist $\mathsf{D} = (\bigcup_A [\![A]\!]^\downarrow)_\bot$.

Kontexte Γ werden interpretiert als abhängiges Produkt:

$$[\![\Gamma]\!] = \prod_{x\in\mathsf{dom}(\Gamma)} [\![\Gamma(x)]\!]$$

Lemma 8.10 (Stetigkeit der Update-Operation $\rho[x \mapsto d]$)

$$\_[x \mapsto \_] \in [[\![\Gamma]\!] \times [\![A]\!] \to [\![\Gamma, x\!:\!A]\!]]$$

Definition und Lemma 8.11 *Interpretation $[\![t]\!]_\rho \in [\![C]\!]$ von Termen $\Gamma \vdash t : C$ in der Umgebung $\rho \in [\![\Gamma]\!]$ mit gleichzeitigem Wohldefiniertheitsbeweis $[\![t]\!]_\_ \in [[\![\Gamma]\!] \to [\![C]\!]]$.*

Fall $\Gamma \vdash x : \Gamma(x)$. *Dann* $[\![x]\!]_\rho = \rho(x)$.

Fall

$$\frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

Dann $[\![r\, s]\!]_\rho = [\![r]\!]_\rho([\![s]\!]_\rho)$. *Applikation ist stetig, da* $[\![r]\!]_\rho \in [[\![A]\!] \to [\![B]\!]]$ *stetig.*

Fall

$$\frac{\Gamma, x\!:\!A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B}$$

Dann $[\![\lambda x t]\!]_\rho = f$ *mit* $f(d) = [\![t]\!]_{\rho[x \mapsto d]}$. *Stetigkeit von f folgt aus der Stetigkeit der Update-Operation und der Stetigkeit von* $[\![t]\!]$ *(nach I.H.).*

Fall $\Gamma \vdash c \in C$. *Dann ist* $[\![c]\!]_\rho$ *die entsprechende Operation, definiert im letzten Abschnitt.*

8.5 Trash

Lemma 8.12 (Abschlusseigenschaften von D)

1. $\mathsf{D} \otimes \mathsf{D} \subseteq \mathsf{D}$.
2. $\mathsf{D} \oplus \mathsf{D} \subseteq \mathsf{D}$.

Beweis.

1. Sei $d \in \mathsf{D} \otimes \mathsf{D}$. Falls $d = \bot$, dann $d \in \mathsf{D}$. Andernfalls $d = (a, b)$ und $a, b \neq \bot$. Also gibt es A, B mit $a \in [\![A]\!]$ und $b \in [\![B]\!]$. Es folgt $(a, b) \in [\![A \times B]\!] \subseteq \mathsf{D}$.
2. Sei $d \neq \bot \in \mathsf{D} \oplus \mathsf{D}$. Dann ist $d = (i, d')$ mit $d' \neq \bot$. Es gibt also ein A mit $d' \in [\![A]\!]$. Falls $i = 0$ ist $d \in [\![A + B]\!]$ für beliebiges B, andernfalls in $[\![B + A]\!]$.

□

Kapitel 9

Kategorien

Mittels Kategorien kann man abstrakte Modelle des einfach getypten Lambda-Kalküls angeben, deren Instanzen schon behandelte konkrete Modelle wie das mengentheorietische oder bereichstheoretische Modell sind.

Ein Kategorie fasst Strukturen mit ähnlichen Eigenschaften zusammen, wie sie z.B. aus der Algebra bekannt sind. So kann man eine Kategorie der Gruppen oder Ringe betrachten.

Definition 9.1 (Kategorie) *Eine* Kategorie $\mathcal{C}$ *besteht aus einer Sammlung von* Objekten $A, B, C, \ldots$ *und für je zwei Objekte* A, B *einer Sammlung* $\mathcal{C}(A, B)$ *von* Morphismen $f : A \to B$. *Sie hat folgende Eigenschaften:*

1. *Zu jedem Objekt* A *gibt es einen Automorphismus* $\mathsf{id}_A : A \to A$, *der Identitätsabbildung.*

2. *Zu je zwei Morphismen* $f : A \to B$ *und* $g : B \to C$ *gibt es einen Morphismus* $g \circ f : A \to C$ *(sprich: "g nach f"). Dabei gelten folgende Gesetze:*

   (a) *(Eins:)* $\mathsf{id}_B \circ f = f \circ \mathsf{id}_A = f$ *für alle* $f : A \to B$.

   (b) *(Assoziativität:)* $(h \circ g) \circ f = h \circ (g \circ f)$ *für alle* $f : A \to B$, $g : B \to C$, $h : C \to D$.

Die meisten Aussagen über Kategorien sind sehr abstrakt. Um trotzdem intuitiv mit ihnen umgehen zu können, ist es wichtig, eine Reihe von Beispielen für Kategorien ständig parat zu haben, um die abstrakten Aussagen mit etwas Konkretem verknüpfen zu können:

1. Algebraische Strukturen wie Monoide, Gruppen. Ringe, Körper bilden Kategorien. Die Morphismen sind dann entsprechend Monoidhomomorphismen, Gruppenhomomorphismen, etc., also strukturerhaltende Abbildungen. Morphismen können natürlich zusammengesetzt werden, und die Identitätsabbildung einer algebraischen Struktur ist trivialerweise ein Homomorphismus.

2. Die Kategorie SET der Mengen hat als Morphismen die totalen Abbildungen zwischen den Mengen.

3. Jede partielle (Prä-)Ordnung bildet eine triviale Kategorie, wobei $\mathcal{C}(A, B)$ nichtleer ist gdw. $A \leq B$. Reflexivität garantiert die Existenz von Identitätsmorphismen, und Transitivität die der Komposition. Jede CPO is also eine Kategorie.

4. Die Kategorie CPO der CPOs mit stetigen Funktionen zwischen CPOs als Morphismen. Dies motiviert Kategorientheorie als Basis der Semantik von Programmiersprachen.

5. Die Kategorie DCPO der CPOs mit kleinsten Element. Morphismen sind strikte stetige Funktionen.

6. Einfache Typen sind die Objekte einer Kategorie, deren Morphismen λ-Terme sind. So ist $\lambda x x : A \to A$ ein Identitätsmorphismus und $\lambda x.\, g\,(f\,x) : A \to C$ ist die Komposition von zwei Morphismen $f : A \to B$ und $g : B \to C$.

Eine Kategorie ist durch ihre Morphismen eindeutig bestimmt. Eine Kategorie heisst *klein*, falls ihre Objekte Mengen sind. Damit kann man eine Kategorie aller kleinen Kategorien bilden, ohne bekannte Paradoxien der Mengenlehre heraufzubeschwören.

Typischerweise beweist man in der Kategorientheorie die Gleichheit von Morphismen. Die Aussagen/Beweise werden typischerweise als Diagramme gezeichnet. Z.B. $\mathsf{id}_B \circ f = f \circ \mathsf{id}_A = f$ wird gezeichnet als

$$\mathsf{id}_A \circlearrowright A \xrightarrow{f} B \circlearrowright \mathsf{id}_B \ .$$

Man sagt "das Diagramm kommutiert", wenn alle Pfade, von einem Objekt zu einem anderen zu gelangen, identisch sind, also $f = f \circ \mathsf{id}_A = \mathsf{id}_B \circ f = \mathsf{id}_B \circ f \circ \mathsf{id}_A = \ldots$.

Definition 9.2 *Zwei Objekte A, B in einer Kategorie sind* isomorph*, geschrieben $A \cong B$, falls es Morphismen $f : A \to B$ und $g : B \to A$ gibt, so dass $g \circ f = \mathsf{id}_A$ und $f \circ g = \mathsf{id}_B$.*

9.1 Konstruktionen auf Kategorien

Das Produkt $\mathcal{C} \times \mathcal{C}'$ zweier Kategorien ist die Kategorie mit einem Objekt $A \times B$ für je zwei Objekte $A \in \mathcal{C}$ und $B \in \mathcal{C}'$ und einem Morphismus $f \times f' : A \times A' \to B \times B'$ für je zwei Morphismen $f \in \mathcal{C}(A, B)$ und $f' \in \mathcal{C}'(A', B')$.

Übung 9.3 *Zeigen Sie, dass $\mathcal{C} \times \mathcal{C}'$ eine tatsächlich Kategorie ist.*

Die Gegenkategorie $\mathcal{C}^{op}$ (opposite category) einer Kategorie $\mathcal{C}$ besteht aus denselben Objekten wie $\mathcal{C}$ und hat für jeden Morphismus $f \in \mathcal{C}(A, B)$ einen Morphismus $f^{op} \in \mathcal{C}(B, A)$. Es werden also alle Pfeile "umgedreht". Dabei werden jedoch keine Umkehrfunktionen gebildet (geht ja im Allgemeinen auch nicht), sondern nur die Schreibweise ändert sich (der Pfeil zeigt in die "falsche Richtung"). Die Komposition von $f : A \leftarrow B$ und $g : B \leftarrow C$ in $\mathcal{C}^{op}$ schreiben wir als $g; f : A \leftarrow C$. Dabei ist $g; f$ definiert als $f \circ g$.

9.2 Funktoren

Ein *Funktor* ist ein Kategorienhomomorphismus, also eine Abbildung $F : \mathcal{C} \to \mathcal{C}'$ von einer Kategorie $\mathcal{C}$ in eine Kategorie $\mathcal{C}'$, so dass

1. $F(A) \in \mathcal{C}'$ für alle $A \in \mathcal{C}$,
2. $F(f) \in \mathcal{C}'(F(A), F(B))$ für alle $f \in \mathcal{C}(A, B)$,
3. $F(\mathsf{id}_A) = \mathsf{id}_{F(A)}$ für alle $A \in \mathcal{C}$, und
4. $F(g \circ f) = F(g) \circ F(f)$ für alle $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$.

Strenggenommen ist ein Funktor F ein Paar von Abbildungen (F^o, F^m), wobei F^o die Objekte von $\mathcal{C}$ auf die von $\mathcal{C}'$ abbildet und F^m die Morphismen. Jedoch schreibt man beide Abbildungen schlicht F.

Beispiele für Funktoren:

1. Ein "Semantik-Funktor" von der Kategorie der Typen und Terme in die Kategorie der CPOs und stetigen Funktionen.
2. Ein "vergesslicher Funktor" von der Kategorie der CPOs und stetigen Abbildungen in die Kategorie der Mengen und totalen Funktionen, der die Ordnung vergisst.
3. Ein Funktor F von der Kategorie der CPOs in die Kategorie der PCPOs, wobei $F(D) = D_\perp$ und $F(f)(\perp) = \perp$ und $F(f)(d) = f(d)$ für $d \neq \perp$. Jede Funktion wird also auf eine strikte Funktion abgebildet.
4. Ein "besoffener Funktor" $F : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$, der jedes Objekt und jeden Morphismus verdoppelt: $F(A) = A \times A$ und $F(f) = f \times f$.

Nimmt man als Objekte die kleinen Kategorien und als Morphismen die Funktoren zwischen den Kategorien, erhält man die Kategorie CAT.

Eine *natürliche Transformation* $n : F \to G$ zwischen zwei Funktoren $F, G : \mathcal{C} \to \mathcal{D}$ besteht aus Abbildungen $n_A : F(A) \to G(A)$ für jedes Objekt $A \in \mathcal{C}$, so dass folgendes Diagramm kommutiert:

$$\begin{array}{ccc} A & & F(A) \xrightarrow{n_A} G(A) \\ \downarrow f & & \downarrow F(f) \qquad \downarrow G(f) \\ B & & F(B) \xrightarrow{n_B} G(B) \end{array}$$

Natürliche Transformationen entsprechen den polymorphen Funktionen $n : \forall A.\, F(A) \to G(A)$ in funktionalen Programmiersprachen: A ist eine Typvariable, $F(A)$ und $G(A)$ Typen abhängig von A. Z.B. könnte $F(A)$ die Binärbäume mit Knotenmarkierungen in A bezeichnen und $G(A)$ die Listen mit Elementen aus A. Die natürliche Transformation n könnte dann eine Durchlaufsfunktion sein, die alle Knotenmarkierungen in einer Liste aufsammelt. Die Natürlichkeit bedeutet: Werden vor dem Durchlauf die Knotenmarkierungen mit einer beliebigen Funktion f umgeschrieben und danach aufgesammelt, $n \circ F(f)$, ist der Effekt derselbe als sammelte man die Knotenmarkierungen vorher ein und schriebe dann jedes Element der Liste um, $G(f) \circ n$. Dabei wendet $F(f)$ die Funktion f auf jeden Knoten an und $G(f)$ auf jedes Listenelement.

Definition 9.4 (Funktorkategorie) *Gegeben zwei Kategorien $\mathcal{C}$ und $\mathcal{D}$. Die Funktorkategorie $\mathcal{D}^{\mathcal{C}}$ (oder $[\mathcal{C}, \mathcal{D}]$) hat als Objekte die Funktoren von $\mathcal{C}$ nach $\mathcal{D}$ und als Morphismen $\mathcal{D}^{\mathcal{C}}(F, G)$ die natürlichen Transformationen von F nach G. Identität und Komposition sind gegeben durch*

$$\begin{array}{lcl} (\mathsf{id}_F)_A & = & \mathsf{id}_{F(A)} \\ (m \circ n)_A & = & m_A \circ n_A. \end{array}$$

Übung 9.5 *Zeigen Sie, dass die Kategoriengesetze für $\mathcal{D}^{\mathcal{C}}$ gelten.*

9.3 Kartesisch Abgeschlossene Kategorien (CCC)

Kategorientheorie befasst sich kaum mit reinen Kategorien (darüber ist schon alles gesagt), sondern fast ausschliesslich mit Kategorien mit zusätzliche Strukturen. Diese werden durch universelle Eigenschaften charakterisiert.

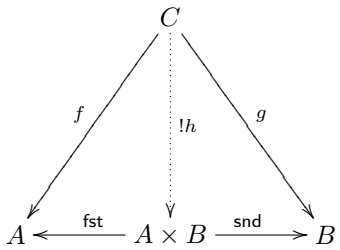
Der einfach getypte Lambda-Kalkül soll durch eine Kategorie modelliert werden. Wir haben bereits gesehen, das wir Typen als Objekte und Terme als Morphismen interpretieren können, allerdings ist das Bild noch unvollständig: Wir können einfache Funktionen $f : A \to B$ darstellen, jedoch (1) noch keine Funktionen mit mehreren Argumenten und (2) keine Funktionen höherer Ordnung wie die Linksapplikation $\mathsf{lapp} : A \to (A \to B) \to B$. Auch keine (3) Terme vom Grundtyp $t : A$. CCCs addressieren alle drei Probleme: (1) durch Postulation von Produktobjekten $A \times B$ für alle Objekte A, B, (2) durch Postulation von Funktionsobjekten B^A (Funktionen von A nach B) für alle A, B, und (3) durch Postulation eines *terminalen Objektes* 1 (mit einem einzigen Bewohner), so dass Terme vom Grundtyp A als Funktionen $1 \to A$ interpretiert werden können.

Definition 9.6 *Eine Kategorie* hat ein terminales Objekt*, genannt* 1*, falls es für jedes Objekt genau einen Morphismus* $\mathsf{unit}_A : 1 \to A$ *gibt.*

Übung 9.7 *Ein terminales Objekt ist bis auf Isomorphie eindeutig bestimmt.*

Definition 9.8 (Produktobjekt) *Ein Produkt zweier Objekte A, B in einer Kategorie $\mathcal{C}$ ist ein Objekt, genannt $A \times B$, mit zwei Morphismen* $\mathsf{fst} : A \times B \to A$

und $\mathsf{snd} : A \times B \to B$*, so dass für jedes Objekt* C *und jedes Paar von Morphismen* $f : C \to A$ *und* $g : C \to A$ genau *ein Morphismus* $h : C \to A \times B$ *existiert, so dass folgendes Diagramm kommutiert:*



Den Morphismus h *nennen wir* $\langle f, g\rangle$*.*

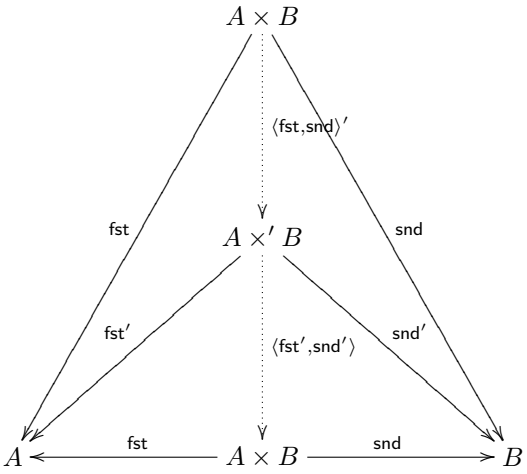
Eine Kategorie $\mathcal{C}$ hat Produkte*, falls zu je zwei Objekten* A, B *ein Produktobjekt* $A \times B$ *existiert.*

Übung 9.9 *Zeigen Sie* $\langle \mathsf{fst}, \mathsf{snd}\rangle = \mathsf{id}$*.*

Kategorielle Produkte sind also immer extensional.

Übung 9.10 *Zeigen Sie: Sind* $A \times B$ *und* $A \times' B$ *beides Produkte von* A *und* B*, so sind* $A \times B$ *und* $A \times' B$ *isomorph.*

Lösung 1 *Das folgende Diagramm kommutiert.*



Wegen der universellen Eigenschaft von $A \times B$ *gibt einen eindeutigen Morphismus* $\langle \mathsf{fst}', \mathsf{snd}'\rangle \circ \langle \mathsf{fst}, \mathsf{snd}\rangle' = \langle \mathsf{fst}, \mathsf{snd}\rangle = \mathsf{id}_{A\times B}$*.*

Da $B \times A$ vermöge $\mathsf{snd}, \mathsf{fst}$ ein Produkt von A und B ist, gilt also $A \times B \cong B \times A$.

Übung 9.11 *Zeigen Sie:* $\langle f,\, g\rangle \circ h = \langle f \circ h,\, g \circ h\rangle$*.*

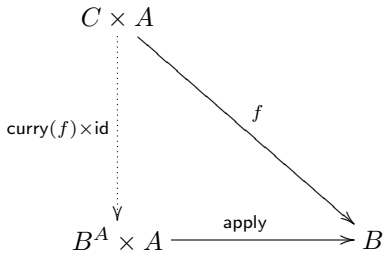
Übung 9.12 *Zeigen Sie:* $A \times 1 \cong A$.

Definition 9.13 (Morphismen-Produkt) *Für* $f : A \to C$ *und* $g : B \to D$ *definieren wir* $f \times g = \langle f \circ \mathsf{fst},\ g \circ \mathsf{snd} \rangle : A \times B \to C \times D$.

Insbesondere gilt $\mathsf{fst} \circ (f \times g) = f \circ \mathsf{fst}$.

Übung 9.14 *Die Kategorie* $\mathcal{C}$ *habe Produkte. Zeigen Sie, dass* $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, *gegeben durch* $F(A, B) = A \times B$ *und* $F(f, g) = f \times g$ *ein Bifunktor ist.*

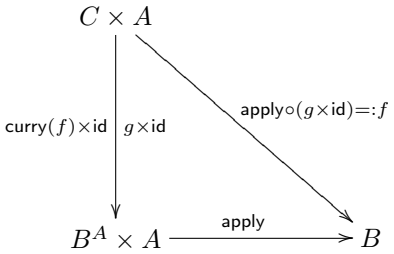
Definition 9.15 (Funktions-Objekt) *Ein* Funktionsobjekt *(oder* Exponent*) von* A *und* B *ist ein Objekt* B^A *mit einem Morphismus* $\mathsf{apply} : B^A \times A \to B$, *so dass es für alle* C *und* $f : C \times A \to B$ *genau ein* $g : C \to B^A$ *gibt, so dass* $\mathsf{apply} \circ (g \times \mathsf{id}) = f$. *Damit kommutiert dieses Diagramm:*



Übung 9.16 *Zeigen Sie:*

1. $\mathsf{curry}(\mathsf{apply}) = \mathsf{id}$.
2. $\mathsf{curry}(\mathsf{apply} \circ (g \times \mathsf{id})) = g$ *für* $g : C \longrightarrow B^A$.
3. $\mathsf{curry}(f) \circ g = \mathsf{curry}(f \circ (g \times \mathsf{id}))$ *für* $f : C \times A \longrightarrow B$ *und* $g : D \longrightarrow C$.

Lösung 2 *Betrachten wir folgendes kommutierende Diagramm:*

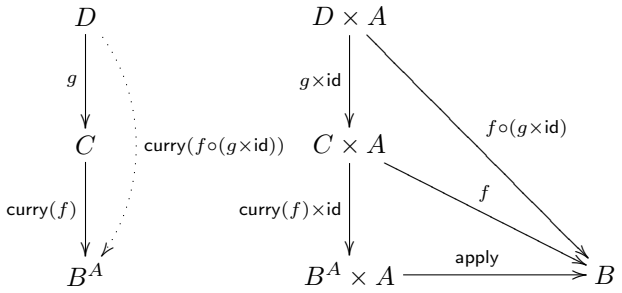


Wegen der Eindeutigkeit ist $\mathsf{curry}(\mathsf{apply} \circ (g \times \mathsf{id})) = g$, *daraus folgt auch die erste Aussage für* $g = \mathsf{id}$.

Für die dritte Aussage betrachte man die Gleichungskette:

$$\begin{aligned} \mathsf{curry}(f) \circ g &= \mathsf{curry}(\mathsf{apply} \circ ((\mathsf{curry}(f) \circ g) \times \mathsf{id})) \\ &= \mathsf{curry}(\mathsf{apply} \circ (\mathsf{curry}(f) \times \mathsf{id}) \circ (g \times \mathsf{id})) \\ &= \mathsf{curry}(f \circ (g \times \mathsf{id})) \end{aligned}$$

Alternativ können wir folgendes Diagramm studieren:



Wegen der Eindeutigkeit ist $\mathsf{curry}(f \circ (g \times \mathsf{id})) = \mathsf{curry}(f) \circ g$.

Übung 9.17 *Der Exponent ist bis auf Isomorphie eindeutig bestimmt.*

Übung 9.18 $A^1 \cong A$.

Übung 9.19 *Die Kategorie $\mathcal{C}$ habe Produkte und Exponenten. Zeigen Sie: $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$, mit $F(A, B) = B^A$ und $F(f, g) = g^f = \mathsf{curry}(g \circ \mathsf{apply} \circ (\mathsf{id} \times f))$ ist ein Bifunktor.*

Übung 9.20 *Sei S ein beliebiges, festes Objekt einer CCC $\mathcal{C}$. Der Objektteil eines Funktors* $\mathsf{Reader} : \mathcal{C} \to \mathcal{C}$ *sei gegeben durch* $\mathsf{Reader}(A) = A^S$. *Definieren Sie die Aktion von* Reader *auf Morphismen $f : A \longrightarrow B$ und beweisen Sie die Funktorgesetze.*

Definieren Sie nun neue Funktoren $F(A) = \mathsf{Reader}(A) \times S$ *und* $\mathsf{Id}(A) = A$ *und zeigen Sie, dass* $\mathsf{apply} : F \to G$ *eine natürliche Transformation ist.*

9.4 CCCs als Modell des Lambda-Kalküls

Wir erweitern den einfach getypten λ-Kalkül um einelementigen Typ, Produkttypen und Konstanten.

A, B	$::=$	$o \mid A \to B \mid 1 \mid A \times B$	Typen
r, s, t	$::=$	$c \mid x \mid \lambda x\!:\!A.\, t \mid r\, s \mid () \mid (r, s) \mid \mathsf{fst}\, r \mid \mathsf{snd}\, r$	Terme
$C[]$	$::=$	$(C[], s) \mid (r, C[]) \mid \mathsf{fst}\, C[] \mid \mathsf{snd}\, C[]$	

Die Typen der Konstanten c sind in einer Signatur Σ gespeichert, so dass $\Sigma(c)$ den Typ der Konstante c darstellt.

Wir erweitern die Typisierungsregeln wie folgt:

$$\frac{}{\Gamma \vdash c : \Sigma(c)} \qquad \frac{}{\Gamma \vdash () : 1}$$

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash (r, s) : A \times B} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \mathsf{fst}\, t : A} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \mathsf{snd}\, t : B}$$

Übung 9.21 (Eindeutigkeit der Typisierung) *Zeigen Sie, dass* $\Gamma \vdash t : A$ *und* $\Gamma \vdash t : B$ *die Gleichheit* $A = B$ *auch für den um obige Regeln erweiterten* λ*-Kalkül implizieren.*

Die getypte $\beta\eta$-Gleichheit wird erweitert um:

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \mathsf{fst}\,(r,\, s) = r : A} \qquad \frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \mathsf{snd}\,(r,\, s) = s : A}$$

$$\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash (\mathsf{fst}\, t,\, \mathsf{snd}\, t) = t : A \times B}$$

9.4.1 Typ- und Kontextinterpretation

Sei $(\mathcal{C}, 1, \times, \to)$ eine CCC. Im Folgenden schreiben wir $f \in A \longrightarrow B$ für $f \in \mathcal{C}(A, B)$.

Sei $\mathcal{C}[\![o]\!] \in \mathcal{C}$ ein Objekt für jeden Grundtyp $o \in \mathsf{Ty}^0$. Wir erweitern $\mathcal{C}[\![-]\!]$ homomorph auf alle Typen durch die folgende rekursive Definition:

$$\begin{array}{lcl} \mathcal{C}[\![1]\!] & = & 1 \\ \mathcal{C}[\![A \times B]\!] & = & \mathcal{C}[\![A]\!] \times \mathcal{C}[\![B]\!] \\ \mathcal{C}[\![A \to B]\!] & = & \mathcal{C}[\![A]\!] \to \mathcal{C}[\![B]\!] \end{array}$$

Kontexte $\Gamma = x_1 : A_1, \ldots, x_n : \mathcal{A}_n$ werden repräsentiert als n-stellige nach links geklammerte Produkte $((A_1 \times A_2) \times A_3) \times \ldots A_n$.

$$\begin{array}{lcll} \mathcal{C}[\![\emptyset]\!] & = & 1 & \text{leerer Kontext} \\ \mathcal{C}[\![\Gamma, x{:}A]\!] & = & \mathcal{C}[\![\Gamma]\!] \times \mathcal{C}[\![A]\!] & \text{Kontexterweiterung} \end{array}$$

Da $1 \times A \cong A$, ist diese rekursive Definition äquivalent zur zuvor informell gegebenen.

9.4.2 Terminterpretation

Sei $\mathcal{C}[\![\emptyset \vdash c : \Sigma(c)]\!] \in 1 \longrightarrow \mathcal{C}[\![\Sigma(c)]\!]$ eine Interpretation der Konstanten als Punkte in $\mathcal{C}$. Wir erweitern diese auf eine Interpretation $\mathcal{C}[\![\Gamma \vdash t : A]\!] \in \mathcal{C}[\![\Gamma]\!] \longrightarrow \mathcal{C}[\![A]\!]$ für alle wohlgetypten Terme durch die rekursive gegebenen Gleichungen:

$$\begin{array}{lcll} \mathcal{C}[\![\Gamma \vdash () : 1]\!] & = & \mathsf{unit} & \\ \mathcal{C}[\![\Gamma, x{:}A \vdash x : A]\!] & = & \mathsf{snd} & \\ \mathcal{C}[\![\Gamma, y{:}A \vdash x : B]\!] & = & \mathcal{C}[\![\Gamma \vdash x : B]\!] \circ \mathsf{fst} & \text{falls } x \neq y \\ \mathcal{C}[\![\Gamma \vdash \mathsf{fst}\, t : A]\!] & = & \mathsf{fst} \circ \mathcal{C}[\![\Gamma \vdash t : A \times B]\!] & \text{wobei } B \text{ eindeutig} \\ \mathcal{C}[\![\Gamma \vdash \mathsf{snd}\, t : B]\!] & = & \mathsf{snd} \circ \mathcal{C}[\![\Gamma \vdash t : A \times B]\!] & \text{wobei } A \text{ eindeutig} \\ \mathcal{C}[\![\Gamma \vdash (r, s) : A \times B]\!] & = & \langle \mathcal{C}[\![\Gamma \vdash r : A]\!], \mathcal{C}[\![\Gamma \vdash s : B]\!] \rangle & \\ \mathcal{C}[\![\Gamma \vdash r\, s : B]\!] & = & \mathsf{apply} \circ \langle \mathcal{C}[\![\Gamma \vdash r : A \to B]\!], \mathcal{C}[\![\Gamma \vdash s : A]\!] \rangle & \\ \mathcal{C}[\![\Gamma \vdash \lambda x{:}A.\, t : A \to B]\!] & = & \mathsf{curry}(\mathcal{C}[\![\Gamma, x{:}A \vdash t : B]\!]) & \end{array}$$

Übung 9.22 *Zeigen Sie, dass tatsächlich* $\mathcal{C}[\![\Gamma \vdash t : A]\!] \in \mathcal{C}[\![\Gamma]\!] \longrightarrow \mathcal{C}[\![A]\!]$.

9.4.3 Substitution

Parallele Substitutionen $\Delta \vdash \sigma : \Gamma$ interpretieren wir als Morphismen $\mathcal{C}[\![\Delta]\!] \longrightarrow \mathcal{C}[\![\Gamma]\!]$.

$$\begin{array}{lcl} \mathcal{C}[\![\Delta \vdash \sigma : \emptyset]\!] & = & \mathsf{unit} \\ \mathcal{C}[\![\Delta \vdash \sigma : \Gamma, x{:}A]\!] & = & \langle \mathcal{C}[\![\Delta \vdash \sigma : \Gamma]\!], \mathcal{C}[\![\Delta \vdash \sigma(x) : A]\!] \rangle \end{array}$$

Lemma 9.23 (Substitution ist Komposition) *Sei $\Gamma \vdash t : A$ und $\Delta \vdash \sigma : \Gamma$. Dann $\mathcal{C}[\![\Gamma \vdash t : A]\!] \circ \mathcal{C}[\![\Delta \vdash \sigma : \Gamma]\!] = \mathcal{C}[\![\Delta \vdash t\sigma : A]\!]$.*

Beweis. Durch lexikographische Induktion über (t, Γ). Sei $f = [\![\Delta \vdash \sigma : \Gamma]\!]$.

Fall $[\![\Gamma \vdash c : \Sigma(c)]\!] \circ f = [\![\emptyset \vdash c : \Sigma(c)]\!] \circ \mathsf{unit} \circ f = [\![\emptyset \vdash c : \Sigma(c)]\!] \circ \mathsf{unit} = [\![\Delta \vdash c : \Sigma(c)]\!] = [\![\Delta \vdash c\sigma : \Sigma(c)]\!]$.

Fall $[\![\Gamma, x{:}A \vdash x : A]\!] \circ [\![\Delta \vdash \sigma : \Gamma, x{:}A]\!] = \mathsf{snd} \circ \langle f, [\![\Delta \vdash \sigma(x) : A]\!] \rangle = [\![\Delta \vdash x\sigma : A]\!]$.

Fall $[\![\Gamma, y{:}B \vdash x : A]\!] \circ [\![\Delta \vdash \sigma : \Gamma, x{:}A]\!] = [\![\Gamma \vdash x : A]\!] \circ \mathsf{fst} \circ \langle f, [\![\Delta \vdash \sigma(y) : B]\!] \rangle = [\![\Gamma \vdash x : A]\!] \circ f = [\![\Delta \vdash x\sigma : A]\!]$, letzter Schritt nach I.H.

Fall $[\![\Gamma \vdash \lambda x{:}A.\, t : A \to B]\!] \circ f = \mathsf{curry}([\![\Gamma, x{:}A \vdash t : B]\!]) \circ f = \mathsf{curry}([\![\Gamma, x{:}A \vdash t : B]\!] \circ \langle f \circ \mathsf{fst}, \mathsf{snd} \rangle)) = \mathsf{curry}([\![\Gamma, x{:}A \vdash t : B]\!] \circ [\![\Delta, x{:}A \vdash \sigma[x \mapsto x] : \Gamma, x{:}A]\!]) = \mathsf{curry}([\![\Delta, x{:}A \vdash t\sigma[x \mapsto x] : B]\!]) = [\![\Delta \vdash (\lambda x{:}A.\, t)\sigma : B]\!]$.

Fall $[\![\Gamma \vdash r\, s : B]\!] \circ f = \mathsf{apply} \circ \langle [\![\Gamma \vdash r : A \to B]\!], [\![\Gamma \vdash s : A]\!] \rangle \circ f = \mathsf{apply} \circ \langle [\![\Gamma \vdash r : A \to B]\!] \circ f, [\![\Gamma \vdash s : A]\!] \circ f \rangle = \mathsf{apply} \circ \langle [\![\Delta \vdash r\sigma : A \to B]\!], [\![\Delta \vdash s\sigma : A]\!] \rangle = [\![\Delta \vdash (r\, s)\, \sigma : B]\!]$.

Fall $[\![\Gamma \vdash (r, s) : A \times B]\!] \circ f = \langle [\![\Gamma \vdash r : A]\!], [\![\Gamma \vdash s : B]\!] \rangle \circ f = \langle [\![\Gamma \vdash r : A]\!] \circ f, [\![\Gamma \vdash s : B]\!] \circ f \rangle = \langle [\![\Delta \vdash r\sigma : A]\!], [\![\Delta \vdash s\sigma : B]\!] \rangle = [\![\Delta \vdash (r, s)\sigma : A \times B]\!]$.

Fall $[\![\Gamma \vdash \mathsf{fst}\, t : A]\!] \circ f = \mathsf{fst} \circ [\![\Gamma \vdash t : A \times B]\!] \circ f = \mathsf{fst} \circ [\![\Delta \vdash t\sigma : A \times B]\!] = [\![\Delta \vdash (\mathsf{fst}\, t)\sigma : A]\!]$.

Fall $\Gamma \vdash \mathsf{snd}\, t : B$ analog.

□

Die Einzelsubstitution $[s/x]$ sei definiert als $[s/x](x) = s$ und $[s/x](y) = y$ für $x \neq y$. Wir haben $[\![\Gamma \vdash [s/x] : \Gamma, x{:}A]\!] = \langle \mathsf{id}, [\![\Gamma \vdash s : A]\!] \rangle$.

9.4.4 Korrektheit

Lemma 9.24 (Beta-Gleichheit für Funktionen) *Sei $\Gamma, x : A \vdash t : B$ und $\Gamma \vdash s : A$. Dann $[\![\Gamma \vdash (\lambda x{:}A.\, t)\, s : B]\!] = [\![\Gamma \vdash t[s/x] : B]\!]$.*

Beweis. $[\![\Gamma \vdash (\lambda x : A.\, t)\, s : B]\!] = \mathsf{apply} \circ \langle [\![\Gamma \vdash \lambda x : A.\, t : A \to B]\!], [\![\Gamma \vdash s : A]\!] \rangle = \mathsf{apply} \circ (\mathsf{curry}([\![\Gamma, x{:}A \vdash t : B]\!]) \times \mathsf{id}) \circ \langle \mathsf{id}, [\![\Gamma \vdash s : A]\!] \rangle = [\![\Gamma, x{:}A \vdash t : B]\!] \circ \langle \mathsf{id}, [\![\Gamma \vdash s : A]\!] \rangle = [\![\Gamma, x{:}A \vdash t : B]\!] \circ [\![\Gamma \vdash [s/x] : \Gamma, x{:}A]\!] = [\![\Gamma \vdash t[s/x] : B]\!]$. □

Korollar 9.25 (Abschwächung/Verstärkung) *Wenn* $x \notin \mathsf{FV}(t)$*, dann* $[\![\Gamma, x : A \vdash t : B]\!] = [\![\Gamma \vdash t : B]\!] \circ \mathsf{fst}$.

Beweis. Für die Identitätssubstitution σ_0 gilt $[\![\Gamma, x : A \vdash \sigma_0 : \Gamma]\!] = \mathsf{fst} \in [\![\Gamma, x : A]\!] \to [\![\Gamma]\!]$, also folgt das Korollar direkt aus Lemma 9.23. □

Lemma 9.26 (Eta-Gleichheit für Funktionen) *Wenn* $x \notin \mathsf{FV}(t)$*, dann* $[\![\Gamma \vdash \lambda x : A.\, t\, x : A \to B]\!] = [\![\Gamma \vdash t : A \to B]\!]$.

Beweis. $[\![\Gamma \vdash \lambda x : A.\, t\, x : A \to B]\!] = \mathsf{curry}([\![\Gamma, x : A \vdash t\, x : B]\!]) = \mathsf{curry}(\mathsf{apply} \circ \langle [\![\Gamma, x : A \vdash t : B]\!], [\![\Gamma, x : A \vdash x : A]\!] \rangle) = \mathsf{curry}(\mathsf{apply} \circ \langle [\![\Gamma \vdash t : B]\!] \circ \mathsf{fst}, \mathsf{snd} \rangle) = [\![\Gamma \vdash t : B]\!]$. □

Satz 9.27 (Korrektheit) *Wenn* $\Gamma \vdash t = t' : A$*, dann* $\mathcal{C}[\![\Gamma \vdash t : A]\!] = \mathcal{C}[\![\Gamma \vdash t' : A]\!]$ *in allen CCCs* $\mathcal{C}$.

9.5 Instanzen von CCCs

9.5.1 Einfache Instanzen

Übung 9.28 *Eine Kategorie* $\mathcal{P}$ *habe als Objekte die aussagenlogischen Formeln und*

$$\mathcal{P}(A, B) = \begin{cases} \{*\} & \text{falls } A \text{ impliziert } B \\ \emptyset & \text{sonst.} \end{cases}$$

Zeigen Sie, dass $\mathcal{P}$ *kartesisch abgeschlossen ist.*

Übung 9.29 *Zeigen Sie: Die Kategorie* SET *der Mengen und totalen Abbildungen ist kartesisch abgeschlossen.*

Übung 9.30 *Zeigen Sie: Die Kategorie* CPO *der vollständigen partiellen Ordnungen und stetigen Funktionen ist kartesisch abgeschlossen.*

9.5.2 Termkategorie

In der Termkategorie sind die Objekte Typen und die Morphismen $t : A \longrightarrow B$ Terme über einer freien Variablen vom Typ A. Kontexte werden als Produkttypen interpretiert, dadurch kann man beliebige Terme $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ als Term $x : (\ldots(1 \times A_1) \times \ldots) \times A_n \vdash t' : B$ darstellen, wobei $t' = t\sigma$ mit $\sigma(x_i) = \mathsf{snd}\,(\mathsf{fst}^{n-i}\, x)$. Um unterschiedliche Typzuweisungen für diesselbe freie Variable in unterschiedlichen Termen zu vermeiden, geben wir uns für jeden Typ A eine spezielle Variable x^A vor (A wird als Teil des Namens der Variable x^A betrachtet), die nur frei verwendet wird. Desweiteren haben wir noch unendlich viele Variablen $y, z, \ldots$, die wir nur gebunden verwenden.

Definition 9.31 (Termkategorie) *Wir definieren die Termkategorie $\mathcal{T}$ durch $\mathcal{T} = \mathsf{Ty}$ (Objekte) und $\mathcal{T}(A,B) = \{\overline{t}^B \mid x^A : A \vdash t : B\}$.*

$$\begin{array}{lcl} \mathsf{id}_A & = & \overline{x^A}^A \\ \overline{t}^C \circ \overline{s}^B & = & \overline{t[s/x^B]}^C \end{array}$$

Im Folgenden verwenden wir $x^A \vdash t : B$ als Abkürzung für $x^A : A \vdash t : B$.

Lemma 9.32 ($\mathcal{T}$ ist Kategorie) *Identität und Komposition sind wohldefiniert und erfüllen die Kategorien-Gesetze.*

Beweis. Übung. □

Lemma 9.33 ($\mathcal{T}$ hat ein terminales Objekt) *Der einelementige Typ* 1 *ist terminales Objekt von $\mathcal{T}$ mit $\mathsf{unit}_A = \overline{()}^1$.*

Beweis. Sei $\overline{t} : A \longrightarrow 1$ ein weiterer Morphismus in die 1. Nach Definition gilt $x^A \vdash t : 1$ und mit dem η-Gesetz für den einelementigen Typ $x^A \vdash t = () : 1$. Also ist $\overline{t} = \overline{()} = \mathsf{unit}_A$. □

Lemma 9.34 ($\mathcal{T}$ hat Produkte) *Der Produkttyp $A \times B$ ist Produktobjekt von A und B vermöge der Projektionen $\overline{\mathsf{fst}\, x^{A\times B}}^A$ und $\overline{\mathsf{snd}\, x^{A\times B}}^B$.*

Beweis. Seien $\overline{r} : C \longrightarrow A$ und $\overline{s} : C \longrightarrow B$ beliebige Morphismen in $\mathcal{T}$. Für den Morphismus $\overline{t} : C \longrightarrow A \times B$ muss gelten $\mathsf{fst} \circ \overline{t} = \overline{r}$, d.h. wegen $\mathsf{fst} \circ \overline{t} = \overline{(\mathsf{fst}\, x^{A\times B})[t/x^{A\times B}]} = \overline{\mathsf{fst}\, t}$, dass $x^C \vdash \mathsf{fst}\, t = r : A$, und analog $x^C \vdash \mathsf{snd}\, t = s : B$. Mit β für Produkte ist $t = (r,s)$ trivialerweise eine Lösung. Jede Lösung t erfüllt $x^C \vdash (\mathsf{fst}\, t, \mathsf{snd}\, t) = (r,s) : A \times B$, also mit η für Produkte auch $x^C \vdash t = (r,s) : A \times B$. Die Lösung $\overline{t} = \overline{(r,s)}$ ist also eindeutig, wir schreiben sie als $\langle \overline{r}, \overline{s} \rangle$. □

Lemma 9.35 ($\mathcal{T}$ hat Exponenten) *Der Funktionstyp $A \to B$ ist Funktionsobjekt von A und B vermöge des Applikationsmorphismus'*

$$\mathsf{apply} = \overline{(\mathsf{fst}\, x^{(A\to B)\times A})\,(\mathsf{snd}\, x^{(A\to B)\times A})}^B : (A \to B) \times A \longrightarrow B.$$

Beweis. Wir stellen zuerst fest, dass für jedes $\overline{t} : C \longrightarrow B$ gilt, dass $\overline{t} \times \mathsf{id}_A = \langle \overline{t} \circ \mathsf{fst}, \mathsf{snd} \rangle = \overline{(t[\mathsf{fst}\, x^{C\times A}/x^C], \mathsf{snd}\, x^{C\times A})}$. Sei $\overline{t} : C \times A \longrightarrow B$ ein beliebiger Morphismus in $\mathcal{T}$. Für den Morphismus $\overline{t'} = \mathsf{curry}(\overline{t}) : C \longrightarrow (A \to B)$ muss gelten $\mathsf{apply} \circ (\mathsf{curry}(\overline{t}) \times \mathsf{id}_A) = \overline{t}$. Dies ist äquivalent zu $x^{C\times A} \vdash$

$(t'[\mathsf{fst}\, x^{C\times A}/x^C])\,(\mathsf{snd}\, x^{C\times A}) = t : B$. Zuerst überzeugen wir uns, dass $t' = \lambda y\colon A.\, t[(x^C, y)/x^{C\times A}]$ die Gleichung löst:

$$\begin{aligned} x^{C\times A} &\vdash (\lambda y\colon A.\, t[(x^C, y)/x^{C\times A}][\mathsf{fst}\, x^{C\times A}/x^C])\,(\mathsf{snd}\, x^{C\times A}) \\ &=_\beta t[(\mathsf{fst}\, x^{C\times A}, \mathsf{snd}\, x^{C\times A})/x^{C\times A}] \\ &=_\eta t[x^{C\times A}/x^{C\times A}] = t : B \end{aligned}$$

(Hier haben wir benutzt, dass $y \notin \mathsf{FV}(t) = \{x^C\}$.) Für eine beliebige Lösung t' gilt ausserdem

$$\begin{aligned} x^C &\vdash \lambda y\colon A.\, t[(x^C, y)/x^{C\times A}] \\ &= \lambda y\colon A.\, ((t'[\mathsf{fst}\, x^{C\times A}/x^C])\,(\mathsf{snd}\, x^{C\times A}))[(x^C, y)/x^{C\times A}] \\ &=_\beta \lambda y\colon A.\, t'[x^C/x^C]\, y \\ &=_\eta t' : A \to B \end{aligned}$$

da $y \notin \mathsf{FV}(t') = \{x^C\}$. Also ist $\mathsf{curry}(t) = t' = \lambda y\colon A.\, t[(x^C, y)/x^{C\times A}]$ die eindeutige Lösung. □

Satz 9.36 ($\mathcal{T}$ ist eine CCC) *Die Kategorie $\mathcal{T}$ ist kartesisch abgeschlossen.*

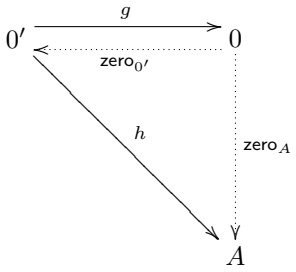
9.6 Endliche Summen

Definition 9.37 (Initiales Objekt) *Ein Objekt* 0 *heisst* initial, *falls es für jedes Objekt* A *genau einen Morphismus* $f : 0 \to A$ *gibt. Dieser Morphismus wird als* zero_A *bezeichnet.*

Übung 9.38 *Zwei initiale Objekte sind isomorph.*

Übung 9.39 *Ein zu* 0 *isomorphes Objekt ist selbst initial.*

Lösung 3 *Sei* $0'$ *isomorph zu* 0*. Betrachte folgendes Diagramm.*



Es gilt $h \circ \mathsf{zero}_{0'} = \mathsf{zero}_A$*, also* $h \circ \mathsf{zero}_{0'} \circ g = \mathsf{zero}_A \circ g$ *und wegen* $\mathsf{zero}_{0'} \circ g = \mathsf{id}$ *auch* $h = \mathsf{zero}_A \circ g$*, das Diagramm kommutiert also und* h *ist eindeutig.*

Definition 9.40 (Null-Objekt) *Ein initial-terminales Objekt heißt auch Null-Objekt (Zero-Objekt).*

Beispiele für Kategorien mit Null-Objekt:

- Die Kategorie der Relationen. Dort ist die leere Menge gleichzeitig initial und terminal, da die Pfeile immer in beide Richtungen gehen.
- Die Kategorie der Gruppen. Dort ist die triviale Gruppe, die nur aus dem Einselement besteht, initial und terminal.
- Analog dazu: Die Kategorie der Mengen A mit einem ausgezeichneten Element $a \in A$. Ein Morphismus $f : (A, a) \to (B, b)$ muss $f(a) = b$ erfüllen. Jede einelementige Menge ist initial und terminal (die leere Menge gibt es in dieser Kategorie nicht).
- Sonderfall: Die Kategorie der PCPOs und strikten Abbildungen.

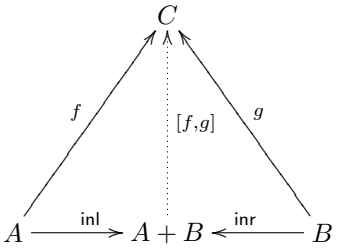
Konsequenz: Es gibt Kategorien mit $f : C \to 0$ und C ist nicht initial.

FRAGE: Gilt folgendes Metatheorem?: Ein durch eine universelle Eigenschaft charakterisiertes Objekt kann immer durch ein isomorphes ausgetauscht werden.

Übung 9.41 *Wenn* $C \times 0 \cong 0$ *für alle* C*, dann ist* $A^0 \cong 1$.

Bemerkung 9.42 *$C \times 0$ ist nicht immer initial, z.B. ist in der Kategorie der Gruppen 0 die einelementige Gruppe, also $0 = 1$, und $C \times 0 \cong C$.*

Definition 9.43 (Koprodukt) *Das Koprodukt zweier Objekte A und B ist ein Objekt, genannt $A + B$, mit zwei Morphismen $\mathsf{inl} : A \longrightarrow A + B$ und $\mathsf{inr} : B \longrightarrow A + B$, so dass es für alle C und Morphismen $f : A \longrightarrow C$ und $g : B \longrightarrow C$ genau einen Morphismus, genannt $[f, g]$, gibt, so dass folgendes Diagramm kommutiert.*



Alternativ können wir auch durch *Dualität* definieren, d.h., $(A + B, \mathsf{inl}, \mathsf{inr})$ ist Koprodukt von A, B in Kategorie $\mathcal{C}$, wenn es Produkt von A, B in C^{op} ist. Damit erben wir die Eindeutigkeit des Koprodukts bis auf Isomorphie von der des Produkts.

Übung 9.44 *Zeigen sie, dass $A + B = \{(0, a) \mid a \in A\} \cup \{(1, b) \mid b \in B\}$ Koprodukt von A und B in der Kategorie SET der Mengen und totalen Funktionen ist.*

Übung 9.45 *Definieren Sie das Produkt $A \times B$ in der Kategorie AB der abelschen Gruppen. Zeigen Sie, dass $A \times B$ auch das Koprodukt von A und B in AB ist.*

Lösung 4 *$A \times B$ ist einfach das kartesische Produkt der Gruppen A und B mit punktweiser Null, Addition und Negation (auch direktes Produkt genannt).*

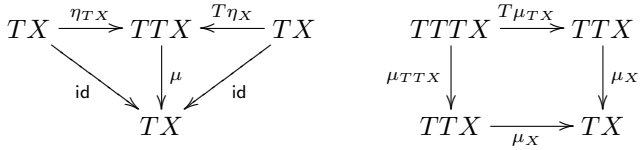
Wir definieren $\mathsf{inl} : A \longrightarrow A \times B$ durch $\mathsf{inl}(a) = (a, 0)$ und analog $\mathsf{inr}(b) = (0, b)$. Es ist leicht zu sehen, dass dies Gruppenhomomorphismen sind. Seien nun $f : A \longrightarrow C$ und $g : B \longrightarrow C$ beliebige Gruppenhomomorphismen. Wir definieren $[f, g] : A \times B \longrightarrow C$ durch $[f, g](a, b) = f(a) + g(b)$. Dies ist ein Gruppenhomomorphismus wegen $[f, g](0, 0) = f(0) + g(0) = 0$, mit Kommuntativität $[f, g]((a, b) + (a', b')) = f(a + a') + g(b + b') = f(a) + g(b) + f(a') + g(b') = [f, g](a, b) + [f, g](a', b')$ und $[f, g](-(a, b)) = f(-a) + g(-b) = -(f(a) + g(b)) = -[f, g](a, b)$.

Nun gilt $[f, g](\mathsf{inl}(a)) = f(a) + g(0) = f(a)$ und analog $[f, g] \circ \mathsf{inr} = g$. Andererseits folgt für jeden Gruppenhomomorphismus $h : A \times B \longrightarrow C$ aus $h(a, 0) = f(a)$ und $h(0, b) = g(b)$ durch Addition $h(a, b) = h(a, 0) + h(0, b) = f(a) + g(b)$, also ist $h = [f, g]$ eindeutig bestimmt.

NB: Die Konstruktion von $[f, g]$ geht auch in kommuntativen Monoiden, also fallen auch der Kategorie der kommutativen Monoide Produkt und Koprodukt zusammen.

9.7 Monaden

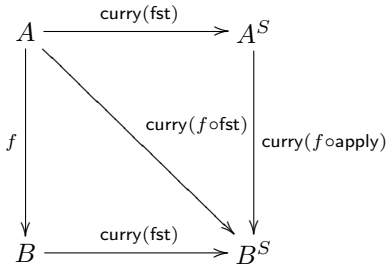
Definition 9.46 *Eine* Monade *auf einer Kategorie C ist ein Funktor $T : C \to C$ zusammen mit zwei natürlichen Transformationen $\eta_X : X \to TX$ (Unit) und $\mu_x : TTX \to TX$ (Multiplikation), so dass folgende Diagramme für alle X kommutieren:*



Übung 9.47 *In einer CCC $\mathcal{C}$ ist* Reader $: \mathcal{C} \to \mathcal{C}$ *mit* Reader$(A) = A^S$ *eine Monade.*

Lösung 5 *Es gilt* Reader$(f : A \longrightarrow B) =$ curry$(f \circ$ apply$) : A^S \longrightarrow B^S$. *Eins ist* $\eta_A =$ curry$($fst$_{A,B}) : A \longrightarrow A^S$ *und Multiplikation ist* $\mu_A =$ curry$($apply $\circ$ $\langle$apply, snd$\rangle)$.

1. *Die Einheit* $\eta :$ Id $\to$ Reader *ist natürliche Transformation, da das folgende Diagramm kommutiert.*

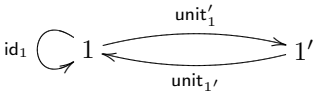


2. $\mu \circ \eta =$ id.

9.8 Lösungen

Übung 9.48 *Ein terminales Objekt ist bis auf Isomorphie eindeutig bestimmt.*

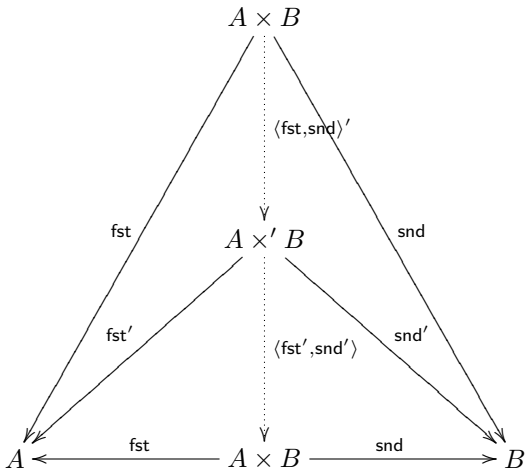
Lösung 6



Es gibt einen eindeutigen Morphismus $\mathsf{unit}_1 = \mathsf{id}_1 : 1 \to 1$. *Also ist* $\mathsf{unit}_{1'} \circ \mathsf{unit}'_1 = \mathsf{id}_1$.

Übung 9.49 *Zeigen Sie: Sind* $A \times B$ *und* $A \times' B$ *beides Produkte von* A *und* B*, so sind* $A \times B$ *und* $A \times' B$ *isomorph.*

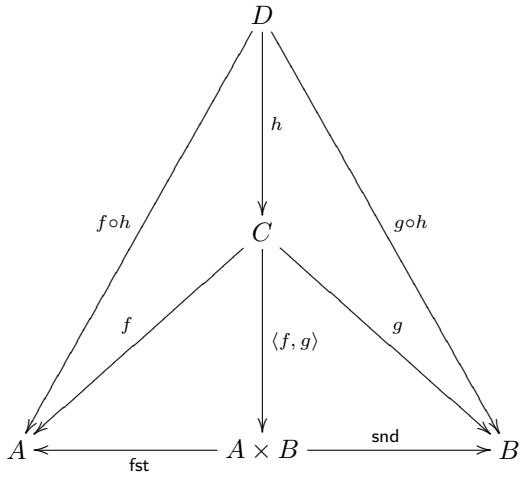
Lösung 7 *Das folgende Diagramm kommutiert.*



Wegen der universellen Eigenschaft von $A \times B$ *gibt einen eindeutigen Morphismus* $\langle \mathsf{fst}', \mathsf{snd}' \rangle \circ \langle \mathsf{fst}, \mathsf{snd} \rangle' = \langle \mathsf{fst}, \mathsf{snd} \rangle = \mathsf{id}_{A \times B}$.

Übung 9.50 *Zeigen Sie:* $\langle f, g \rangle \circ h = \langle f \circ h, g \circ h \rangle$.

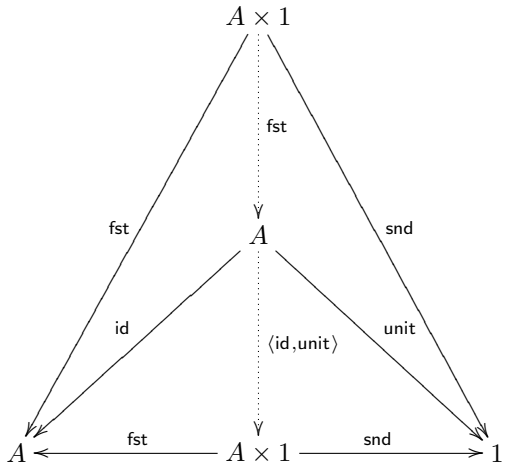
Lösung 8



Da der durch $f \circ h$ und $g \circ h$ induzierte Morphismus $D \longrightarrow A \times B$ eindeutig ist, gilt $\langle f, g\rangle \circ h = \langle f \circ h, g \circ h\rangle$.

Übung 9.51 *Zeigen Sie: $A \times 1 \cong A$.*

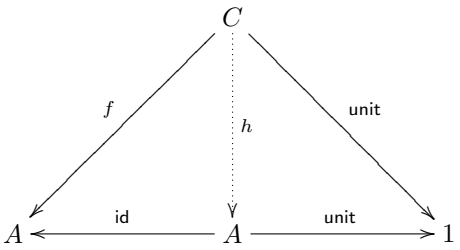
Lösung 9 *Es gibt Morphismen $\mathsf{fst} : A \times 1 \to A$ und $\langle \mathsf{id}_A, \mathsf{unit}_A\rangle : A \to A \times 1$.*



Es gilt $\mathsf{fst} \circ \langle \mathsf{id}, \mathsf{unit}\rangle = \mathsf{id}$ (kleines Dreieck unten links), und $\langle \mathsf{id}, \mathsf{unit}\rangle \circ \mathsf{fst} = \langle \mathsf{fst}, \mathsf{snd}\rangle = \mathsf{id}$ (große Achse).

Lösung 10 *(Eleganter.) Wir zeigen, dass A Produkt von A und 1 ist. Dann*

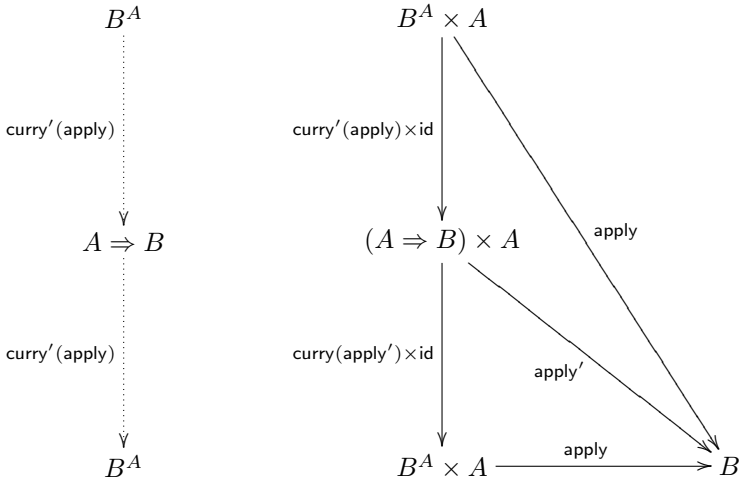
gilt $A \cong A \times 1$. Sei C beliebig und das folgende Diagramm kommutierend.



Zu zeigen: es gibt genau ein h. Es gibt: $h = f$. Genau ein: Es gilt $\mathsf{id} \circ h = f$.

Übung 9.52 *Der Exponent ist bis auf Isomorphie eindeutig bestimmt.*

Lösung 11 *Sei $(A \Rightarrow B, \mathsf{apply}')$ ein weiteres Funktionsobjekt von A nach B mit* $\mathsf{curry}' : \mathcal{C}(C \times A, B) \to \mathcal{C}(C, A \Rightarrow B)$.

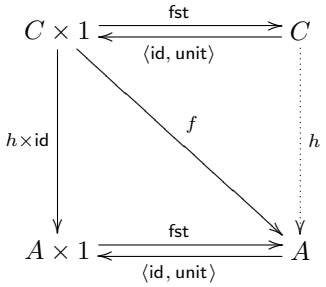


Wegen der Eindeutigkeit ist $\mathsf{curry}'(\mathsf{apply}) \circ \mathsf{curry}(\mathsf{apply}') = \mathsf{curry}(\mathsf{apply}) = \mathsf{id}_{B^A}$ *(großes Dreieck). Analog gilt* $\mathsf{curry}(\mathsf{apply}') \circ \mathsf{curry}'(\mathsf{apply}) = \mathsf{curry}'(\mathsf{apply})' = \mathsf{id}_{A \Rightarrow B}$. *Also ist $A \Rightarrow B$ isomorph zu B^A.*

Übung 9.53 $A^1 \cong A$.

Lösung 12 *Wir benutzen $A \times 1 \cong A$ und zeigen dann, dass A ein Funktions-*

objekt mit Definitionsbereich 1 *und Wertebereich* A *ist.*



Wegen $C \cong C \times 1$ *ist* $h = f \circ \langle \mathsf{id}, \mathsf{unit} \rangle$ *durch* f *eindeutig bestimmt. Beachte* $\mathsf{fst} \circ (h \times \mathsf{id}) = h \times \mathsf{fst} = \mathsf{fst}$*, also kommutiert das Dreieck links unten.*

Übung 9.54 *Die Kategorie* $\mathcal{C}$ *habe Produkte und Exponenten. Zeigen Sie:* $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$*, mit* $F(A, B) = B^A$ *und* $F(f, g) = g^f = \mathsf{curry}(g \circ \mathsf{apply} \circ (\mathsf{id} \times f))$ *ist ein Bifunktor.*

Lösung 13 *Das erste Funktorgesetz,* $F(\mathsf{id}, \mathsf{id}) = \mathsf{id}$ *gilt nach Üebung 9.16.*

Für das zweite Funktorgesetz, nehmen wir an, dass

$$\begin{array}{rcl@{\qquad}rcl} f & : & A \leftarrow B & h & : & D \to E \\ g & : & B \leftarrow C & i & : & E \to F. \end{array}$$

Zu zeigen ist $F(g; f,\ i \circ h) = F(g, i) \circ F(f, h)$*. Es gelten*

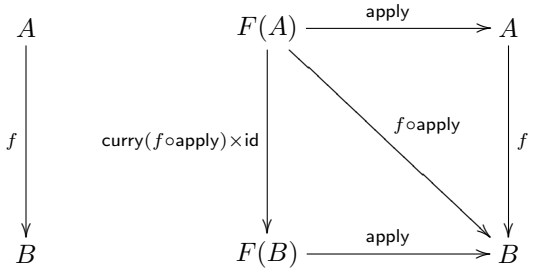
$$\begin{array}{lcl} (i \circ h)^{(g;f)} & = & \mathsf{curry}(i \circ h \circ \mathsf{apply} \circ (\mathsf{id} \times (f \circ g))) \\ i^g & = & \mathsf{curry}(i \circ \mathsf{apply} \circ (\mathsf{id} \times g)) \\ h^f & = & \mathsf{curry}(h \circ \mathsf{apply} \circ (\mathsf{id} \times f)) \\ & =: & \mathsf{curry}(k) \\ \\ i^g \circ h^f & = & \mathsf{curry}(i \circ \mathsf{apply} \circ (\mathsf{id} \times g)) \circ \mathsf{curry}(k) \\ & = & \mathsf{curry}(i \circ \mathsf{apply} \circ (\mathsf{id} \times g) \circ (\mathsf{curry}(k) \times \mathsf{id})) \\ & = & \mathsf{curry}(i \circ \mathsf{apply} \circ (\mathsf{curry}(k) \times g)) \\ & = & \mathsf{curry}(i \circ \mathsf{apply} \circ (\mathsf{curry}(k) \times \mathsf{id}) \circ (\mathsf{id} \times g)) \\ & = & \mathsf{curry}(i \circ k \circ (\mathsf{id} \times g)) \\ & = & \mathsf{curry}(i \circ (h \circ \mathsf{apply} \circ (\mathsf{id} \times f)) \circ (\mathsf{id} \times g)) \\ & = & \mathsf{curry}(i \circ h \circ \mathsf{apply} \circ (\mathsf{id} \times (f \circ g))) \\ & = & (i \circ h)^{(g;f)} \end{array}$$

Übung 9.55 *Sei* S *ein beliebiges, festes Objekt einer CCC* $\mathcal{C}$*. Der Objektteil eines Funktors* $\mathsf{Reader} : \mathcal{C} \to \mathcal{C}$ *sei gegeben durch* $\mathsf{Reader}(A) = A^S$*. Definieren Sie die Aktion von* Reader *auf Morphismen* $f : A \longrightarrow B$ *und beweisen Sie die Funktorgesetze.*

Definieren Sie nun neue Funktoren $F(A) = \mathsf{Reader}(A) \times S$ *und* $\mathsf{Id}(A) = A$ *und zeigen Sie, dass* $\mathsf{apply} : F \to G$ *eine natürliche Transformation ist.*

Lösung 14 $\mathsf{Reader}(f : A \longrightarrow B) = \mathsf{curry}(f \circ \mathsf{apply}) : A^S \longrightarrow B^S$. *Es gilt* $\mathsf{Reader}(\mathsf{id}) = \mathsf{curry}(\mathsf{id} \circ \mathsf{apply}) = \mathsf{curry}(\mathsf{apply}) = \mathsf{id}$. *Ausserdem* $\mathsf{Reader}(f) \circ \mathsf{Reader}(g) = \mathsf{curry}(f \circ \mathsf{apply}) \circ \mathsf{curry}(g \circ \mathsf{apply}) = \mathsf{curry}(f \circ \mathsf{apply} \circ (\mathsf{curry}(g \circ \mathsf{apply}) \times \mathsf{id})) = \mathsf{curry}(f \circ g \circ \mathsf{apply}) = \mathsf{Reader}(f \circ g)$.

Wir setzen $F(f : A \longrightarrow B) = \mathsf{Reader}(f) \times \mathsf{id}$. *Damit kommutiert das folgende Natürlichkeitsdiagramm für* $\mathsf{apply} : F \to \mathsf{Id}$.



Kapitel 10

Polymorphismus

Einführendes Beispiel: Polymorphe Listen.

- JAVA-Polymorphismus `Object` $\geq A$.

```
class List {
  ...
  int    length();
  void   cons(Object a);
  void   append(List l);
  Object front();
}
```

- Nachteil: Hat man homogene Listen, z.B., `Integer` Listen, muss man trotzdem immer casten: `(Integer)l.front()`.
- F-Polymorphismus in JAVA 1.5 behebt dieses Problem:

```
class List<A> {
  ...
  int    length();
  void   cons(A a);
  void   append(List<A> l);
  A      front();
}
```

- Polymorphismus in Haskell:

```
data List a = Nil
            | Cons a (List a)

map :: forall a b. (a -> b) -> List a -> List b
map f Nil        = Nil
map f (Cons a l) = Cons (f a) (map f l)
```

Wir studieren nun den Polymorphismus in Rein-Form, d.h., im λ-Kalkül.

10.1 System F, Curry-Stil

Syntaktische Kategorien.

Λ	$\ni$	r, s, t	$::=$	$x \mid \lambda x t \mid r\, s$	ungetypte Terme
Ty	$\ni$	A, B, C	$::=$	$A \to B \mid \forall X A \mid X$	einfache und polymorphe Typen
Cxt	$\ni$	Γ	$::=$	$\diamond \mid \Gamma, x\!:\!A \mid \Gamma, X$	Typisierungs-Kontexte

Typisierung.

$$\textsc{ty-var}\ \frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \textsc{ty-abs}\ \frac{\Gamma, x\!:\!A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B} \qquad \textsc{ty-app}\ \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

$$\textsc{ty-gen}\ \frac{\Gamma, X \vdash t : A}{\Gamma \vdash t : \forall X A} \qquad \textsc{ty-inst}\ \frac{\Gamma \vdash t : \forall X A}{\Gamma \vdash t : A[B/X]}$$

Übung 10.1 *Geben Sie Typisierungsherleitungen für*

1. $\vdash \lambda x x : \forall X.\, X \to X$
2. $\vdash \lambda x.\, x\, x : (\forall X X) \to \forall X X.$

Beispiel 10.2 (Nicht typisierbare Terme) *Die Terme Ω und $\underline{2}\,\underline{2}\,\mathsf{K}$ sind in System F nicht typisierbar.*

Typisierung ist unentscheidbar (Wells, 1994). Wohlgetypte Terme sind stark normalisierend, jedoch nicht umgekehrt (Beispiel: $\underline{2}\,\underline{2}\,\mathsf{K}$). Es gilt Typerhaltung unter β-Reduktion, jedoch nicht unter η-Reduktion.

Beispiel 10.3 (η-Reduktion erhält Typen nicht) *Sei $X \notin \mathsf{FV}(A)$. Folgende Typisierung ist gültig:*

$$y\!:\!A \to \forall X B \vdash \lambda x.\, y\, x : \forall X.\, A \to B$$

Jedoch hat y nicht den Typ $\forall X.\, A \to B$.

10.1.1 Imprädikative Kodierungen

Einige Konstruktionen im ungetypten λ-Kalkül, wie z.B. Church-Ziffern und Tupel, können wir in System F typisieren.

$$\begin{array}{lcll} \mathsf{N} & = & \forall X.\, (X \to X) \to X \to X & \\ \underline{n} & = & \lambda f x.\, f^n\, x & : \quad \mathsf{N} \\ \underline{+} & = & \lambda n m f x.\, n\, f\, (m\, f\, x) & : \quad \mathsf{N} \\ \underline{\cdot} & = & \lambda n m f.\, n\, (m\, f) & : \quad \mathsf{N} \\ \mathsf{exp} & = & \lambda n m.\, n\, m & : \mathsf{N} \end{array}$$

Übung 10.4 *Programmieren Sie die Ackermann-Funktion in System F!*

Lösung 15 $\mathsf{ack} = \lambda n.\, n\,(\lambda r \lambda m.\, m\, r\, \underline{1})\,(\lambda m.\, m \underline{+} \underline{1})$.

Im Allgemeinen kann man eine Datenstruktur imprädikativ kodieren, wenn sie einen *universalen Eliminator* hat, d.h., jede Verwendung der Datenstruktur kann mit dem u.E. programmiert werden. Der universale Eliminator für Paare ist split.

$$\begin{array}{lcl}
\mathsf{split} & : & \forall AB.\, A \times B \to \forall C.\, (A \to B \to C) \to C \\
\mathsf{split}\,(\mathsf{pair}\, a\, b)\, k & \longrightarrow & k\, a\, b \\
\mathsf{fst} & = & \lambda p.\, \mathsf{split}\, p\, \mathsf{T} \\
\mathsf{snd} & = & \lambda p.\, \mathsf{split}\, p\, \mathsf{F}
\end{array}$$

Nun nimmt man als Definition für $A \times B$ den Zieltyp $\forall C.\, (A \to B \to C) \to C$ des universalen Eliminators.

$$\begin{array}{lcl}
A \times B & = & \forall C.\, (A \to B \to C) \to C \\
\\
\mathsf{pair} & : & \forall AB.\, A \to B \to A \times B \\
\mathsf{pair} & = & \lambda ab.\, \lambda k.\, k\, a\, b \\
\\
\mathsf{split} & : & \forall AB.\, A \times B \to \forall C.\, (A \to B \to C) \to C \\
\mathsf{split} & = & \lambda xx
\end{array}$$

Der universale Eliminator für Summentypen ist $\mathsf{case} : \forall AB.\, A + B \to \forall C.\, (A \to C) \to (B \to C) \to C$. Also funktioniert folgende Kodierung.

$$\begin{array}{lcl}
A + B & = & \forall C.\, (A \to C) \to (B \to C) \to C \\
\\
\mathsf{inl} & : & \forall AB.\, A \to A + B \\
\mathsf{inl} & = & \lambda a.\, \lambda f g.\, f\, a \\
\\
\mathsf{inr} & : & \forall AB.\, B \to A + B \\
\mathsf{inr} & = & \lambda b.\, \lambda f g.\, g\, b \\
\\
\mathsf{case} & : & \forall AB.\, A + B \to \forall C.\, (A \to C) \to (B \to C) \to C \\
\mathsf{case} & = & \lambda xx
\end{array}$$

Der universale Eliminator für Listen ist foldr:

$$\begin{array}{lcl}
\mathsf{foldr} & : & \forall A.\, \mathsf{List}\, A \to \forall B.\, (A \to B \to B) \to B \to B \\
\mathsf{foldr}\,[a_1, \ldots, a_n]\,(\otimes)\, e & \longrightarrow & a_1 \otimes (a_2 \otimes \ldots (a_n \otimes e) \ldots) \\
\mathsf{foldr}\,\mathsf{nil}\,(\otimes)\, e & \longrightarrow & e \\
\mathsf{foldr}\,(\mathsf{cons}\, a\, l)\,(\otimes)\, e & \longrightarrow & a \otimes \mathsf{foldr}\, l\,(\otimes)\, e
\end{array}$$

Wir kodieren Listen also als

$$\begin{array}{lcl} \mathsf{List}\,A & = & \forall B.\,(A \to B \to B) \to B \to B \\ \\ \mathsf{nil} & : & \forall A.\,\mathsf{List}\,A \\ \mathsf{nil} & = & \lambda f e.\,e \\ \\ \mathsf{cons} & : & \forall A.\,A \to \mathsf{List}\,A \to \mathsf{List}\,A \\ \mathsf{cons} & = & \lambda a l.\,\lambda f e.\,f\,a\,(l\,f\,e) \\ \\ \mathsf{foldr} & = & \lambda x x \end{array}$$

Übung 10.5 *Implementieren Sie* map *und* append *mit dieser Listen-Kodierung.*

Eine andere Sichtweise auf die imprädikative Kodierung ist: Sei X ein Datentyp mit den Konstruktoren $c_1 : F_1(X) \ldots c_n : F_n(X)$. Dann ist $\forall X.\,F_1(X) \to \cdots \to F_n(X) \to X$ die imprädikative Kodierung von X.

Übung 10.6 *Machen Sie sich diese Sichtweise an den Beispielen* N, $\times$, $+$ *und* $\mathsf{List}\,A$ *klar.*

Übung 10.7 *Kodieren Sie den Typ* $\mathsf{BTree}\,A$ *der Binärbäume mit Knotenbeschriftungen in* A *imprädikativ.*

Kapitel 11

Typinferenz

11.1 Einfache Typen

Im Folgenden stehen X, Y, Z für instantiierbare Typvariablen (Unifikationsvariablen), A, B, C für Typen mit Variablen und τ für Typen ohne Variablen.

$$\begin{array}{lcl} \tau & ::= & o \mid \tau_1 \to \tau_2 \\ A, B, C & ::= & X \mid o \mid A \to B \end{array}$$

Sei $\mathsf{EV}(A)$ die Menge der in A vorkommenden Variablen X. Eine Typsubstitution ξ ist eine Abbildung von Typvariablen X auf Typen A. Der Definitionsbereich $\mathsf{dom}(\xi)$ ist definiert als $\{X \mid \xi(X) \neq X\}$. Wir erlauben nur Substitutionen mit endlichem Definitionsbereich.

Definition 11.1 (Anwendung einer Substitution) *Die Anwendung einer Substitution ξ auf einen Typen A, einen Kontext Γ und eine Substitution ξ' ist wie folgt definiert:*

$$\begin{array}{lcl} X\xi & = & \xi(X) \\ o\xi & = & o \\ (A \to B)\xi & = & A\xi \to B\xi \\ (\Gamma\xi)(x) & = & (\Gamma(x))\xi \\ (\xi'\xi)(X) & = & (\xi'(X))\xi \end{array}$$

Definition 11.2 (Instantiierungsquasiordnung) *Wir sagen A ist* allgemeiner *als B, falls B eine* (Substitutions-)Instanz *von A ist, falls also $A\xi = B$ für*

ein ξ. Wir definieren folgende Begriffe:

$$\begin{array}{lcl} A \leq_\xi B & \iff & A\xi = B \\ \Gamma \leq_\xi \Gamma' & \iff & \forall x \in \mathsf{dom}(\Gamma).\ \Gamma(x) \leq_\xi \Gamma'(x) \\ (\Gamma \vdash A) \leq_\xi (\Gamma' \vdash A') & \iff & \Gamma \leq_\xi \Gamma' \textit{ und } A \leq_\xi A' \end{array}$$

$$\begin{array}{lcl} A \leq B & \iff & \exists \xi.\ A \leq_\xi B \\ \Gamma \leq \Gamma' & \iff & \exists \xi.\ \Gamma \leq_\xi \Gamma' \\ (\Gamma \vdash A) \leq (\Gamma' \vdash A') & \iff & \exists \xi.\ (\Gamma \vdash A) \leq_\xi (\Gamma' \vdash A') \end{array}$$

Die Relation $\leq$ ist nur eine Quasi-Ordnung (engl. *preorder*), d.h., reflexiv und transitiv, aber nicht antisymmetrisch, da z.B. $X \leq Y$ und $Y \leq X$, aber natürlich $X \neq Y$.

Definition 11.3 (Isomorphe Substitutionen) *Zwei Substitutionen ξ, ξ' sind* isomorph*, falls $\xi \leq \xi'$ und $\xi' \leq \xi$.*

Definition 11.4 (Unifikator) *Ein* Unifikator *zweier Typen A, A' ist eine Substitution ξ mit $A\xi = A'\xi$. Entsprechend ist ξ Unifikator zweier Kontexte Γ und Γ' falls $\Gamma(x)\xi = \Gamma'(x)\xi$ für alle $x \in \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma')$. Analog definiert man einen Unifikator zweier Substitutionen.*

Definition 11.5 (Allgemeinster Unifikator) *Der* allgemeinste *Unifikator* $\mathsf{mgu}(A, B)$ *(engl.* most general unifier*) zweier Typen A, B hat die zusätzliche Eigenschaft, dass für alle weiteren Unifikatoren ξ von A und B gilt: $\mathsf{mgu}(A, B) \leq \xi$. Entsprechend definiert man den allgemeinsten Unifikator zweier Kontexte, Substitutionen etc.*

Lemma 11.6 (Existenz) *Sind zwei Entitäten (Typen, Kontexte, etc.) unifizierbar, so existiert ein allgemeinster Unifikator.*

Beweis. Z.B. durch Angabe eines Unifikationsalgorithmusses. □

Lemma 11.7 (Eindeutigkeit) *Der allgemeinste Unifikator ist bis auf Isomorphie eindeutig bestimmt.*

Beweis. Seien ξ und ξ' zwei allgemeinste Unifikatoren. Dann gilt nach Definition $\xi \leq \xi'$ und $\xi' \leq \xi$. □

Wir schreiben $\Gamma \setminus x$ für den Kontext Γ' mit $\mathsf{dom}(\Gamma') = \mathsf{dom}(\Gamma) \setminus \{x\}$ und $\Gamma'(y) = \Gamma(y)$ für $y \in \mathsf{dom}(\Gamma')$. Falls $\Gamma(x) = \Gamma'(x)$ für alle $x \in \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma')$, dann ist $\Gamma + \Gamma'$ definiert als $\Gamma \cup \Gamma'$. Es gilt dann $(\Gamma + \Gamma')\xi = \Gamma\xi + \Gamma'\xi$.

Definition 11.8 (Typinferenz) *Gegeben ein λ-Term t, berechnet der als Relation $t \rightrightarrows \Gamma \vdash C$ gegebene Typinferenz-Algorithmus einen Kontext Γ und einen*

Typ C von t in Γ.

$$\frac{}{x \rightrightarrows x{:}X \vdash X}\ X \text{ neu}$$

$$\frac{t \rightrightarrows \Gamma \vdash B}{\lambda x t \rightrightarrows \Gamma \vdash X \to B}\ x \notin \mathsf{dom}(\Gamma), X \text{ neu} \qquad \frac{t \rightrightarrows \Gamma \vdash B}{\lambda x t \rightrightarrows \Gamma \setminus x \vdash \Gamma(x) \to B}\ x \in \mathsf{dom}(\Gamma)$$

$$\frac{r \rightrightarrows \Gamma_1 \vdash C_1 \qquad s \rightrightarrows \Gamma_2 \vdash C_2}{r\,s \rightrightarrows \Gamma_1\xi + \Gamma_2\xi \vdash X\xi}\ X \text{ neu}, \xi = \mathsf{mgu}(\Gamma_1 \vdash C_1, \Gamma_2 \vdash C_2 \to X)$$

Typinferenz terminiert immer, da durch strukturelle Rekursion über den Term t definiert.

Satz 11.9 (Korrektheit) *Falls $t \rightrightarrows \Gamma \vdash C$, dann $\Gamma \vdash t : C$.*

Beweis. Durch Induktion über $t \rightrightarrows \Gamma \vdash C$.

Fall

$$\frac{r \rightrightarrows \Gamma_1 \vdash C_1 \qquad s \rightrightarrows \Gamma_2 \vdash C_2}{r\,s \rightrightarrows \Gamma_1\xi + \Gamma_2\xi \vdash X\xi}\ X \text{ neu}, \xi = \mathsf{mgu}(\Gamma_1 \vdash C_1, \Gamma_2 \vdash C_2 \to X)$$

Wir zeigen $\Gamma_1\xi + \Gamma_2\xi \vdash r\,s : X\xi$. Es gilt $C_1\xi = C_2\xi \to X\xi$. Wir setzen $A \to B := C_1\xi$. Da ξ die Kontexte Γ_1 und Γ_2 unifiziert, ist $\Gamma_1\xi + \Gamma_2\xi =: \Gamma$ wohldefiniert. Es gilt $\Gamma_1\xi \subseteq \Gamma$, also mit der Induktionsvoraussetzung, $\Gamma \vdash r : A \to B$. Analog gilt $\Gamma \vdash s : A$, also $\Gamma \vdash r\,s : B$.

□

Typinferenz berechnet immer die allgemeinste Typisierung $\Gamma' \vdash C'$ eines Terms t. Dies ist sogar notwendig für die Vollständigkeit des Algorithmusses im Fall $r\,s$.

Satz 11.10 (Vollständigkeit) *Falls $\Gamma \vdash t : C$, dann $t \rightrightarrows \Gamma' \vdash C'$ und $(\Gamma' \vdash C') \leq (\Gamma \vdash C)$.*

Beweis. Durch Induktion über $\Gamma \vdash t : C$.

Fall

$$\frac{}{\Gamma \vdash x : \Gamma(x)}$$

Dann $x \rightrightarrows x : X \vdash X$ für eine neue Typvariable X. Die Substitution $\xi(X) = \Gamma(x)$ validiert $(x{:}X \vdash X) \leq (\Gamma \vdash \Gamma(x))$.

Fall

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B}$$

Dann $t \rightrightarrows \Gamma' \vdash B'$ und nach I.V. ist $(\Gamma' \vdash B') \leq_\xi (\Gamma, x : A \vdash B)$. Falls $x \in \mathsf{dom}(\Gamma')$, dann $\Gamma' = \Gamma'', x : A'$ mit $\Gamma'' = \Gamma' \setminus x$ and $A' = \Gamma'(x)$. Es gilt also $(\Gamma'', x : A' \vdash B') \leq_\xi (\Gamma, x : A \vdash B)$, und damit $(\Gamma'' \vdash A' \to B') \leq_\xi (\Gamma \vdash A \to B)$. Wir sind fertig, da $\lambda x t \rightrightarrows \Gamma'' \vdash A' \to B'$. Ähnlich argumentiert man im Fall $x \notin \mathsf{dom}(\Gamma')$.

Fall

$$\frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

Nach I.V., $r \rightrightarrows \Gamma_1 \vdash C_1$ mit $(\Gamma_1 \vdash C_1) \leq_{\xi_1} (\Gamma \vdash A \to B)$ und $s \rightrightarrows \Gamma_2 \vdash C_2$ mit $(\Gamma_2 \vdash C_2) \leq_{\xi_2} (\Gamma \vdash A)$. Da wir stets neue Typvariablen erzeugen, können wir annnehmen, dass $\Gamma_1 \vdash C_1$ sich keine Variablen mit $\Gamma_2 \vdash C_2$ teilt. Sei X eine neue Typvariable. Dann ist $\xi' = \xi_1 \cup \xi_2 \cup (X \mapsto B)$ ein Unifikator von $\Gamma_1 \vdash C_1$ und $\Gamma_2 \vdash C_2 \to X$. Es existiert also auch ein allgemeinster Unifikator $\xi \leq_{\xi_0} \xi'$. Daraus folgt $r\, s \rightrightarrows \Gamma_1\xi + \Gamma_2\xi \vdash X\xi$, und $(\Gamma_1\xi + \Gamma_2\xi \vdash X\xi) \leq_{\xi_0} (\Gamma \vdash B)$.

□

11.1.1 Optimierung

Der Typinferenzalgorithmus lässt sich noch dahingehend optimieren, dass nicht jede Termvariable eine neue Typvariable zugeordnet bekommt und dann konflingiernde Verwendungen einer Variable unifiziert werden, sondern dass bei jeder λ-Abstraktion die Termvariable mit einer neuen, festen Variable in einen Kontext eingefügt wird. Dieser Kontext Γ wird als Eingabe an das Typinferenzproblem geliefert, zu Beginn muss er für jede freie Variable eines Programms definiert sein. Zurückgegeben wird ein Typ A und eine Substitution ξ für die sich in Γ befindenden Typvariablen. Sei id die Identitätssubstitution.

$$\frac{}{\Gamma \vdash x \rightrightarrows \Gamma(x), \mathsf{id}} \qquad \frac{\Gamma, x{:}X \vdash t \rightrightarrows B, \xi}{\Gamma \vdash \lambda x t \rightrightarrows X\xi \to B, \xi}\ X \text{ neu}$$

$$\frac{\Gamma \vdash r \rightrightarrows C, \xi \qquad \Gamma\xi \vdash s \rightrightarrows A, \xi'}{\Gamma \vdash r\, s \rightrightarrows X\xi'', \xi\xi'\xi''}\ X \text{ neu}, \xi'' = \mathsf{mgu}(C\xi', A \to X)$$

Dieser Algorithmus hat den Vorteil, dass keine Kontexte unifiziert werden müssen, außerdem ist er ökonomischer, da er weniger Typvariablen verbraucht. Allerdings ist er nun reihenfolgenabhängig, da nun zuerst der Typ von r inferiert wird und dann der von s. Wir eine Variable x und r und s mit unterschiedlichen Typen benutzt, hat det Typ von x in r Vorrang, der Fehler wird dann bei der ersten Benutzung von x in s gemeldet.

Der Algorithums W von Milner ist eine Erweiterung dieses optimierten Algorithmusses um let-Polymorphismus.

11.2 Naiver Let-Polymorphismus

Eine einfache Form von let-Polymorphismus wird durch die Regel

$$\frac{\Gamma \vdash s : A \qquad \Gamma \vdash t[s/x] : C}{\Gamma \vdash \mathsf{let}\ x = s\ \mathsf{in}\ t : C}$$

implementiert. Michell beschreibt folgenden Typinferenzalgorithmus dafür:

Typinferenz $\Delta \vdash t \rightrightarrows \Gamma \vdash C$. Eingabe: Term t und ein Kontext Δ von Typisierungen. Ausgabe: Monotyp mit Variablen C und ein Kontext Γ von Monotypen mit Variablen.

$$\frac{\Delta(x) = (\Gamma \vdash A)}{\Delta \triangleright x \rightrightarrows \vdash \Gamma[\vec{X}/\vec{Y}] \vdash A[\vec{X}/\vec{Y}]}\ \vec{Y} = \mathsf{EV}(\Gamma \vdash A), \vec{X} \text{ neu} \qquad \frac{x \notin \mathsf{dom}(\Delta)}{\Delta \triangleright x \rightrightarrows x : X \vdash X}\ X \text{ neu}$$

$$\frac{\Delta \triangleright t \rightrightarrows \Gamma \vdash B}{\Delta \triangleright \lambda x t \rightrightarrows \Gamma \vdash X \to B}\ x \notin \mathsf{dom}(\Gamma), X \text{ neu} \qquad \frac{\Delta \triangleright t \rightrightarrows \Gamma \vdash B}{\Delta \triangleright \lambda x t \rightrightarrows \Gamma \setminus x \vdash \Gamma(x) \to B}\ x \in \mathsf{dom}(\Gamma)$$

$$\frac{\Delta \triangleright r \rightrightarrows \Gamma_1 \vdash C_1 \qquad \Delta \triangleright s \rightrightarrows \Gamma_2 \vdash C_2}{\Delta \triangleright r\, s \rightrightarrows \Gamma_1\xi + \Gamma_2\xi \vdash X\xi}\ X \text{ neu}, \xi = \mathsf{mgu}(\Gamma_1 \vdash C_1, \Gamma_2 \vdash C_2 \to X)$$

$$\frac{\Delta \triangleright r \rightrightarrows \Gamma \vdash A \qquad \Delta, x : (\Gamma \vdash A) \triangleright s \rightrightarrows \Gamma' \vdash C}{\Delta \triangleright \mathsf{let}\ x = r\ \mathsf{in}\ s \rightrightarrows \Gamma' \vdash C}$$

Damas und Milner beschreiben ein Typsystem, das diesselbe Stärke wie der naive let-Polymorphismus hat, aber sinnvollere Typisierungsregeln.

11.3 Let-Polymorphismus mit Typschemata

TODO: look at Joe Well's "The essence of principal typings". (ICALP 2002)

Typinferenz für System F ist unentscheidbar. Der Polymorphismus in System F ist imprädikativ: Typvariablen können mit polymorphen Typen (kurz: Polytypen) instantiiert werden. Im Gegensatz dazu können bei prädikativen Polymorphismus Typvariablen nur mit nicht-polymorphen Typen (monomorphen Typen, kurz Monotypen) instantiiert werden. Ein Subsystem des prädikativen System F erlaubt die Bildung polymorpher Objekte nur an gewissen stellen, in einer let-Definition. Typinferenz für dieses Subsystem ist entscheidbar (Milner 1978); darauf basieren die funktionalen Sprachen ML und Haskell.

A, B, C	::=	$X \mid \alpha \mid o \mid A \to B$	Typ mit Variablen
τ	::=	$\alpha \mid o \mid \tau_1 \to \tau_2$	monomorpher Typ, Monotyp
σ	::=	$\tau \mid \forall \alpha \sigma$	polymorpher Typ, Polytyp oder Typschema

Typsystem (Damas-Milner).

$$\frac{}{\Gamma \vdash_{\mathsf{DM}} x : \Gamma(x)} \qquad \frac{\Gamma, x : \tau \vdash_{\mathsf{DM}} t : \tau'}{\Gamma \vdash_{\mathsf{DM}} \lambda x t : \tau \to \tau'} \qquad \frac{\Gamma \vdash_{\mathsf{DM}} r : \tau \to \tau' \qquad \Gamma \vdash_{\mathsf{DM}} s : \tau}{\Gamma \vdash_{\mathsf{DM}} r\, s : \tau'}$$

$$\frac{\Gamma \vdash_{\mathsf{DM}} r : \sigma \qquad \Gamma, x : \sigma \vdash_{\mathsf{DM}} s : \sigma'}{\Gamma \vdash_{\mathsf{DM}} \mathsf{let}\ x = r\ \mathsf{in}\ s : \sigma'} \qquad \frac{\Gamma \vdash_{\mathsf{DM}} t : \forall \alpha \sigma}{\Gamma \vdash_{\mathsf{DM}} t : \sigma[\tau/\alpha]} \qquad \frac{\Gamma \vdash_{\mathsf{DM}} t : \sigma}{\Gamma \vdash_{\mathsf{DM}} t : \forall \alpha \sigma}\ \alpha \notin \mathsf{FV}(\Gamma)$$

Man beachte das in der Instantiierungsregel nur ein Monotyp τ für die Variable α eingesetzt werden kann.

Lemma 11.11 (Abschluss unter Substitution) *Wenn $\mathcal{D} :: \Gamma \vdash_{\mathsf{DM}} t : \sigma$, dann gibt es eine Herleitung $\mathcal{D}' :: \Gamma\xi \vdash_{\mathsf{DM}} t : \sigma\xi$ von derselben Länge wie $\mathcal{D}$.*

Dieses Typsystem implementiert noch keinen Algorithmus, da die Instantiierungs- und Generalisierungsregeln an jeder Stelle angewendet werden können. Man kann sie jedoch in die Regeln für Variablen und let einbauen.

$$\frac{\Gamma(x) = \forall\vec{\alpha}\tau}{\Gamma \vdash x : \tau[\vec{\tau}/\vec{\alpha}]} \qquad \frac{\Gamma, x{:}\tau \vdash t : \tau'}{\Gamma \vdash \lambda x t : \tau \to \tau'} \qquad \frac{\Gamma \vdash r : \tau \to \tau' \qquad \Gamma \vdash s : \tau}{\Gamma \vdash r\, s : \tau'}$$

$$\frac{\Gamma \vdash r : \tau \qquad \vec{\alpha} = \mathsf{FV}(\tau) \setminus \mathsf{FV}(\Gamma) \qquad \Gamma, x{:}\forall\vec{\alpha}\tau \vdash s : \tau'}{\Gamma \vdash \mathsf{let}\ x = r\ \mathsf{in}\ s : \tau'}$$

Dieses System ist nun syntaxgerichtet. Im Kontext befinden sich Polytypen, rechts von $\vdash$ nur Monotypen. Abschluss unter Substitution ist nun für die let-Regel nicht mehr trivial, folgt aber, falls wir zeigen können, dass $\Gamma \vdash t : \tau$ gdw. $\Gamma \vdash_{\mathsf{DM}} t : \tau$.

Satz 11.12 (Korrektheit) *Wenn* $\Gamma \vdash t : \tau$*, dann* $\Gamma \vdash_{\mathsf{DM}} t : \tau$*.*

Beweis. Einfach, durch Induktion über die Herleitung. Im Falle der Variablenregel muss man die Instantiierungsregel von Damas-Milner wiederholt anwenden, im Falle der let-Regel die Generalisierungsregel. □

Wir definieren induktiv das Urteil $\Gamma' \prec \Gamma$ (Γ' ist allgemeiner als Γ).

$$\frac{}{\diamond \prec \diamond} \qquad \frac{\Gamma' \prec \Gamma}{(\Gamma', x{:}\forall\vec{\alpha}\sigma) \prec (\Gamma, x{:}\sigma)}$$

Satz 11.13 (Vollständigkeit) *Wenn* $\mathcal{D} :: \Gamma \vdash_{\mathsf{DM}} t : \forall\vec{\alpha}\tau$ *und* $\Gamma' \prec \Gamma$*, dann* $\Gamma' \vdash t : \tau[\vec{\tau}/\vec{\alpha}]$.

Beweis. Durch Induktion über $\mathcal{D}$.

Fall

$$\frac{\Gamma(x) = \forall\vec{\alpha}\tau}{\Gamma \vdash_{\mathsf{DM}} x : \forall\vec{\alpha}\tau}$$

Da $\Gamma' \prec \Gamma$, ist $\Gamma'(x) = \forall\vec{\alpha}'\vec{\alpha}\tau$. Damit $\Gamma' \vdash x : \tau[\vec{\tau}/\vec{\alpha}]$.

Fall

$$\frac{\Gamma \vdash_{\mathsf{DM}} r : \forall\vec{\alpha}'\tau' \qquad \Gamma, x{:}\forall\vec{\alpha}'\tau' \vdash_{\mathsf{DM}} s : \forall\vec{\alpha}\tau}{\Gamma \vdash_{\mathsf{DM}} \mathsf{let}\ x = r\ \mathsf{in}\ s : \forall\vec{\alpha}\tau}$$

Nach I.V., $\Gamma' \vdash r : \tau'$. Es gilt $\vec{\alpha}' \notin \mathsf{FV}(\Gamma') \subseteq \mathsf{FV}(\Gamma)$. Seien $\vec{\alpha}'' = \mathsf{FV}(\forall\vec{\alpha}'\tau') \setminus \mathsf{FV}(\Gamma')$. Nun gilt $(\Gamma', x{:}\forall\vec{\alpha}''\vec{\alpha}'\tau') \prec (\Gamma, x{:}\forall\vec{\alpha}'\tau')$. Nach I.V., $\Gamma', x{:}\forall\vec{\alpha}''\vec{\alpha}'\tau' \vdash s : \tau[\vec{\tau}/\vec{\alpha}]$, also $\Gamma' \vdash \mathsf{let}\ x = r\ \mathsf{in}\ s : \tau[\vec{\tau}/\vec{\alpha}]$.

Fall O.B.d.A., $\alpha, \vec{\alpha} \notin \mathsf{FV}(\tau, \vec{\tau})$.

$$\frac{\Gamma \vdash_{\mathsf{DM}} t : \forall\alpha\vec{\alpha}\tau'}{\Gamma \vdash_{\mathsf{DM}} t : (\forall\vec{\alpha}\tau')[\tau/\alpha]}$$

Nach I.V., $\Gamma' \vdash t : \tau'[\tau, \vec{\tau}/\alpha, \vec{\alpha}]$, und wir sind fertig.

Fall O.B.d.A., $\alpha, \vec{\alpha} \notin \mathsf{FV}(\tau, \vec{\tau})$.

$$\frac{\Gamma \vdash_{\mathsf{DM}} t : \forall \vec{\alpha} \tau'}{\Gamma \vdash_{\mathsf{DM}} t : \forall \alpha \vec{\alpha} \tau'} \; \alpha \notin \mathsf{FV}(\Gamma)$$

Zu zeigen ist $\Gamma' \vdash t : \tau'[\tau, \vec{\tau}/\alpha, \vec{\alpha}]$. Unter Benutzung der Variablenbedingung und des Substitutionslemmas erhalten wir eine Herleitung $\mathcal{D}' :: \Gamma \vdash_{\mathsf{DM}} t : \forall \vec{\alpha}.\tau'[\tau/\alpha]$, auf die wir die Induktionshypothese anwenden können. Damit sind wir fertig.

□

Korollar 11.14 (Substitution) *Wenn* $\Gamma \vdash t : \tau$*, dann* $\Gamma\xi \vdash t : \tau\xi$.

Beweis. Mit Korrektheit gilt zuerst $\Gamma \vdash_{\mathsf{DM}} t : \tau$, damit $\Gamma\xi \vdash_{\mathsf{DM}} t : \tau\xi$. Mit Vollständigkeit, $\Gamma\xi \vdash t : \tau\xi$. □

11.3.1 Algorithmus W

Wie bei den einfachen Typen können wir den Typinferenzalgorithmus noch optimieren: wir erhalten den *Algorithmus W*.

$$\frac{\Gamma(x) = \forall \vec{\alpha} A}{\Gamma \vdash x \rightrightarrows A[\vec{X}/\vec{\alpha}], \mathsf{id}} \; \vec{X} \text{ neu} \qquad \frac{\Gamma, x{:}X \vdash t \rightrightarrows B, \xi}{\Gamma \vdash \lambda x t \rightrightarrows X\xi \to B, \xi} \; X \text{ neu}$$

$$\frac{\Gamma \vdash r \rightrightarrows C, \xi \qquad \Gamma\xi \vdash s \rightrightarrows A, \xi'}{\Gamma \vdash r\, s \rightrightarrows X\xi'', \xi\xi'\xi''} \; X \text{ neu}, \xi'' = \mathsf{mgu}(C\xi', A \to X)$$

$$\frac{\Gamma \vdash r \rightrightarrows A, \xi \qquad \Gamma\xi, x{:}\mathsf{Gen}_{\Gamma\xi} A \vdash s \rightrightarrows B, \xi'}{\Gamma \vdash \mathsf{let}\; x = r \;\mathsf{in}\; s \rightrightarrows B, \xi\xi'}$$

Lemma 11.15 *Wenn* $\Gamma \vdash t \rightrightarrows C, \xi$*, dann* $\mathsf{FV}(\xi)$ *neu und* $\mathsf{FV}(C)$ *disjunkt zu* $\mathsf{dom}(\xi)$*. Insbesondere ist* ξ *idempotent und* $C\xi = C$.

Im folgenden Satz betrachten wir eine Erweiterung der Typisierung auf Typen C mit existentiellen Variablen X. Diese werden wie universelle Variablen, also Monotypen, behandelt.

Satz 11.16 (Korrektheit) *Wenn* $\Gamma \vdash t \rightrightarrows C, \xi$*, dann* $\Gamma\xi \vdash_{\mathsf{DM}} t : C$.

Beweis. Durch Induktion über die Herleitung von $\Gamma \vdash t \rightrightarrows C, \xi$.

Fall

$$\frac{\Gamma \vdash r \rightrightarrows C, \xi \qquad \Gamma\xi \vdash s \rightrightarrows A, \xi'}{\Gamma \vdash r\, s \rightrightarrows X\xi'', \xi\xi'\xi''} \; X \text{ neu}, \xi'' = \mathsf{mgu}(C\xi', A \to X)$$

Nach I.V., $\Gamma\xi \vdash_{\mathsf{DM}} r : C$, also $\Gamma\xi\xi'\xi'' \vdash_{\mathsf{DM}} r : C\xi'\xi''$. Außerdem $\Gamma\xi\xi' \vdash_{\mathsf{DM}} s : A$, also $\Gamma\xi\xi'\xi'' \vdash_{\mathsf{DM}} s : A\xi''$. Da ξ'' die Typen $C\xi'$ und $A \to X$ unifiziert, gilt $\Gamma\xi\xi'\xi'' \vdash_{\mathsf{DM}} r\, s : X\xi''$.

Fall

$$\frac{\Gamma \vdash r \rightrightarrows A, \xi \qquad \Gamma\xi, x{:}\mathsf{Gen}_{\Gamma\xi,A} \vdash s \rightrightarrows B, \xi'}{\Gamma \vdash \mathsf{let}\ x = r\ \mathsf{in}\ s \rightrightarrows B, \xi\xi'}$$

Nach I.V., $\Gamma\xi \vdash_{\mathsf{DM}} r : A$, und unter Verwendung der Generalisierungsregel $\Gamma\xi \vdash_{\mathsf{DM}} r : \mathsf{Gen}_{\Gamma\xi}A$. Nach Substitution, $\Gamma\xi\xi' \vdash_{\mathsf{DM}} r : (\mathsf{Gen}_{\Gamma\xi}A)\xi'$. Ebenfalls nach I.V., $\Gamma\xi\xi', x{:}(\mathsf{Gen}_{\Gamma\xi}A)\xi' \vdash_{\mathsf{DM}} s : B$, also $\Gamma\xi\xi' \vdash_{\mathsf{DM}} \mathsf{let}\ x = r\ \mathsf{in}\ s : B$.

□

Alternativ können wir auch beweisen: Wenn $\Gamma \vdash t \rightrightarrows C, \xi$, dann $\Gamma\xi\xi' \vdash_{\mathsf{DM}} t : C\xi'$ für alle abschließenden Substitutionen ξ'.

Wir definieren induktiv

$$\frac{}{\diamond \prec \diamond} \qquad \frac{\Gamma' \prec \Gamma}{(\Gamma', x{:}\forall\vec{\alpha}.A[\vec{\alpha}/\vec{X}]) \prec (\Gamma, x{:}\forall\vec{\beta}.A[\vec{\tau}/\vec{X}])}\ \vec{\beta} \notin \mathsf{FV}(A)$$

Dabei ist die Seitenbedingung trivial erfüllt, wenn A überhaupt keine freie universellen Variablen enthält, wie es bei den Typen in den Kontexten des Algorithmus W der Fall ist.

Lemma 11.17 *Sei* $\Gamma' \prec \Gamma$.

1. $\Gamma'\xi \prec \Gamma\xi$.
2. $X \in \mathsf{FV}(\Gamma')$ *gdw.* $X \in \mathsf{FV}(\Gamma)$.

Satz 11.18 (Vollständigkeit) *Wenn* $\Gamma\xi \vdash_{\mathsf{DM}} t : \forall\vec{\beta}\tau$ *und* $\Gamma' \prec \Gamma$, *dann* $\Gamma' \vdash t \rightrightarrows A, \xi'$ *und es gibt eine schließende Substitution* η *mit* $A\eta = \tau$ *und* $Y\xi'\eta = Y\xi$ *für alle* $Y \in \mathsf{FV}(\Gamma)$ *und* $\vec{\beta} \notin \mathsf{FV}(\eta \restriction \mathsf{FV}(\Gamma'\xi))$.

Beweis.

Fall

$$\frac{}{\Gamma\xi \vdash_{\mathsf{DM}} x : \Gamma(x)\xi}$$

Da $\Gamma' \prec \Gamma$ ist $\Gamma'(x) = \forall\vec{\alpha}.A[\vec{\alpha}/\vec{X}]$ für neue existentielle Variablen $\vec{X}$ und $\Gamma(x) = \forall\vec{\beta}.A[\vec{\tau}/\vec{X}]$. Es gilt $\Gamma \vdash x \rightrightarrows A, \mathsf{id}$. Mit $\eta = [\vec{\tau}/\vec{X}]\xi$ haben wir $A\eta = (A[\vec{\tau}/\vec{X}])\xi$ und $Y\mathsf{id}\eta = Y\xi$ für alle $Y \in \mathsf{FV}(\Gamma)$, da die Variablen $\vec{X}$ neu waren. Die Bedingung $\vec{\beta} \notin \mathsf{FV}(\eta \restriction \mathsf{FV}(\Gamma'\mathsf{id})) = \mathsf{FV}(\xi \restriction \mathsf{FV}(\Gamma'))$ kann durch mögliche Variablenumbenennung leicht erfüllt werden.

Fall

$$\frac{\Gamma\xi, x{:}\tau \vdash_{\mathsf{DM}} t : \tau'}{\Gamma\xi \vdash_{\mathsf{DM}} \lambda x t : \tau \to \tau'}$$

Sei X neu, dann gilt $(\Gamma', x : X) \prec (\Gamma, x : X)$. Mit $\xi' = \xi[\tau/X]$ gilt $(\Gamma, x : X)\xi' \vdash_{\mathsf{DM}} t : \tau'$, also nach I.V. $\Gamma, x{:}X \vdash t \rightrightarrows A, \xi_1$ mit $A\eta = \tau'$ und $\xi_1\eta = \xi'$. Da X neu war, ist $X\xi_1\eta = X\eta = \tau$. Damit gilt $\Gamma \vdash t \rightrightarrows (X\xi_1 \to A), \xi_1$ mit $(X\xi_1 \to A)\eta = \tau \to \tau'$ und $\xi_1\eta \restriction \mathsf{FV}(\Gamma) = \xi$.

Fall

$$\frac{\Gamma\xi \vdash_{\mathsf{DM}} r : \tau \to \tau' \qquad \Gamma\xi \vdash_{\mathsf{DM}} s : \tau}{\Gamma\xi \vdash_{\mathsf{DM}} r\, s : \tau'}$$

Nach I.V., $\Gamma \vdash r \rightrightarrows C, \xi_1$ mit $C\eta = \tau \to \tau'$ und $\xi_1\eta = \xi$. Nach Annahme, $(\Gamma\xi_1)\eta \vdash_{\mathsf{DM}} s : \tau$, also wieder nach I.V., $\Gamma\xi_1 \vdash s \rightrightarrows A, \xi_2$ mit $A\eta' = \tau$ und $\xi_2\eta' = \eta$. Sei X neu. Da $\eta'[\tau'/X]$ die Typen $C\xi_2$ und $A \to X$ unifiziert, gibt es auch eine allgemeinsten Unifikator ξ_3 mit $\xi_3\eta'' = \eta'$. Also $\Gamma \vdash r\, s \rightrightarrows X\xi_3, \xi_1\xi_2\xi_3$ mit $X\xi_3\eta'' = \tau'$ und $\xi_1\xi_2\xi_3\eta'' = \xi_1\xi_2\eta' = \xi_1\eta = \xi$.

Fall

$$\frac{\Gamma\xi \vdash_{\mathsf{DM}} r : \forall\vec{\beta}\tau \qquad \Gamma\xi, x{:}\forall\vec{\beta}\tau \vdash_{\mathsf{DM}} s : \forall\vec{\beta}'\tau'}{\Gamma \vdash_{\mathsf{DM}} \mathsf{let}\ x = r\ \mathsf{in}\ s : \forall\vec{\beta}'\tau'}$$

Nach I.V. gilt $\Gamma \vdash r \rightrightarrows A, \xi_1$ mit $A\eta = \tau$ und $\xi_1\eta = \xi \restriction \mathsf{FV}(\Gamma)$. Seien nun $\vec{X} = \mathsf{FV}(A) \setminus \mathsf{FV}(\Gamma'\xi_1)$ und sei $\eta_1 = \eta \restriction \vec{X}$. Da $\vec{\beta} \notin \mathsf{FV}(\eta \restriction \mathsf{FV}(\Gamma'\xi_1))$ ist $\forall\vec{\beta}\tau = \forall\vec{\beta}.A\eta = (\forall\vec{\beta}.A\eta_1)\eta$. Man beachte, dass η schließende Substitution ist. Nun ist $(\Gamma'\xi_1, x : \forall\vec{\alpha}.A[\vec{\alpha}/\vec{X}]) \prec (\Gamma\xi_1, x : \forall\vec{\beta}.A\eta_1)$ und nach Voraussetzung gilt $(\Gamma\xi_1, x : \forall\vec{\beta}.A\eta_1)\eta \vdash_{\mathsf{DM}} s : \forall\vec{\beta}'\tau'$. Wir können also die I.V. anwenden und folgern $\Gamma'\xi_1, x{:}\forall\vec{\alpha}.A[\vec{\alpha}/\vec{X}] \vdash s \rightrightarrows A', \xi_2$ mit $A'\eta' = \tau'$ und $\xi_2\eta' = \eta \restriction \mathsf{FV}(\Gamma\xi_1, x{:}\forall\vec{\beta}.A\eta_1)$. Damit $\Gamma' \vdash \mathsf{let}\ x = r\ \mathsf{in}\ s \rightrightarrows A', \xi_1\xi_2$ und $\xi_1\xi_2\eta' = \xi_1\eta = \xi \restriction \mathsf{FV}(\Gamma)$. Die Bedingung $\vec{\beta}' \notin \mathsf{FV}(\eta' \restriction \mathsf{FV}(\Gamma'\xi_1\xi_2)$ folgt direkt aus der zweiten I.V.

Fall

$$\frac{\Gamma\xi \vdash_{\mathsf{DM}} t : \forall\vec{\beta}\tau}{\Gamma\xi \vdash_{\mathsf{DM}} t : \forall\beta\vec{\beta}\tau}\ \beta \notin \mathsf{FV}(\Gamma\xi)$$

Nach I.V., $\Gamma \vdash t \rightrightarrows A, \xi_1$ mit $A\eta = \tau$, $\xi_1\eta = \xi$ und $\vec{\beta} \notin \mathsf{FV}(\eta \restriction \mathsf{FV}(\Gamma'\xi_1))$. Da $\beta \notin \mathsf{FV}(\Gamma\xi_1\eta)$ und die freien existentiellen Variablen von Γ' mit denen von Γ übereinstimmen und Γ keine freien universellen Variablen enthält, ist $\beta \notin \mathsf{FV}(\Gamma'\xi_1\eta) = \mathsf{FV}(\eta \restriction \mathsf{FV}(\Gamma'\xi))$.

Fall

$$\frac{\Gamma\xi \vdash_{\mathsf{DM}} t : \forall\beta\vec{\beta}\tau}{\Gamma\xi \vdash_{\mathsf{DM}} t : \forall\beta.\tau[\tau'/\beta]}$$

Nach I.V., $\Gamma \vdash t \rightrightarrows A, \xi_1$ mit $A\eta = \tau$ und $\xi_1\eta = \xi$. Definiere $\eta' = \eta[\tau'/\beta]$. Dann ist $A\eta' = \tau[\tau'/\beta]$ und $(\xi_1\eta' = \xi[\tau'/\beta] = \xi) \restriction \mathsf{FV}(\Gamma)$, da $\beta \notin \mathsf{FV}(\Gamma\xi)$.

□

11.4 Unentscheidbarkeit von Typprüfung in System F

Wiederholung: Typisierungsregeln für System F.

$$\text{TY-VAR}\ \frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \text{TY-ABS}\ \frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B} \qquad \text{TY-APP}\ \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

$$\text{TY-GEN}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X A}\ X \notin \mathsf{FV}(\Gamma) \qquad \text{TY-INST}\ \frac{\Gamma \vdash t : \forall X A}{\Gamma \vdash t : A[B/X]}$$

Eine normale Herleitung eines Urteils $\Gamma \vdash r : A$ in Curry-System F enthält keine Anwendung einer Instantiierungsregel direkt nach einer Generalisierungsregel. Solche Umwege können entfernt werden:

$$\frac{\dfrac{\mathcal{D}}{\Gamma \vdash t : A}\ X \notin \mathsf{FV}(\Gamma)}{\Gamma \vdash t : A[B/X]} \qquad \longrightarrow \qquad \mathcal{D}[B/X] :: \Gamma \vdash t : A[B/X]$$

Nicht normale Typisierungsherleitungen können zu normalen reduziert werden, daher reicht es, letztere zu betrachen.

Definition 11.19 (Subtyping) *In System F ist Subtyping $C <:_\Gamma D$ gegeben als die kleinste Relation erzeugt von dem Axiomenschema:*

$$\forall \vec{X} A <:_\Gamma \forall \vec{Y}.\, A[\vec{B}/\vec{X}] \qquad \textit{wobei kein } Y_i \in \mathsf{FV}(\Gamma)$$

Lemma 11.20 (Korrektheit des Subtyping) *Wenn $\Gamma \vdash t : C$ und $C <:_\Gamma D$, dann $\Gamma \vdash t : D$.*

Beweis. Eine Instantiierungsregel für jedes B_i, eine Generalisierungsregel für jedes Y_i. □

Satz 11.21 *Subtyping ist unentscheidbar.*

Definition 11.22 (Syntaxgerichtete Typisierung)

$$\frac{\Gamma(x) <:_\Gamma A}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A \qquad B <:_\Gamma B'}{\Gamma \vdash r\, s : B'}$$

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x t : \forall \vec{X}.\, A \to B}\, no\ X_i \in \mathsf{FV}(\Gamma)$$

Lemma 11.23 (Vollständigkeit) *Gibt es eine normale Herleitung von $\Gamma \vdash t : A$, dann auch eine syntaxgerichtete.*

Definition 11.24 (Semi-Unifikation) *Ein* Semi-Unifikationsproblem *(SUP) ist eine Menge von Paaren* $\{(A_i, B_i) \mid i = 1..n\}$ *von* einfachen *Typen mit Variablen. Das Problem ist* lösbar*, falls es eine Substitution* ξ *gibt, so dass* $A_i\xi \leq B_i\xi$*.*

Bemerkung 11.25 *Eine Lösung des entsprechenden* Unifikationsproblems *verlangt* $A_i\xi = B_i\xi$*.*

Beispiel 11.26 *Ein lösbares SUP ist* $\{((Y \to Y) \to Z,\ Z \to (X \to X)),\ ((U \to U) \to (U \to U),\ Z)\}$*.*

Ein unlösbares SUP ist $\{((Y \to Y) \to Z,\ Z \to (X \to X)),\ (U \to U \to U,\ Z)\}$*.*

Unifikation ist entscheidbar, Semi-Unifikation jedoch nicht.

Satz 11.27 (Kfoury,Tiurin,Urzyczyn (1993)) *Semi-Unifikation ist unentscheidbar, bereits für* $n = 2$*.*

Definition 11.28 (Typprüfung) *Gegeben* Γ, t, A *ist* $\Gamma \vdash t : A$*?*

Wells bewies 1994, dass Typprüfung und Typinferenz für System F unentscheidbar sind, durch Reduktion auf ein SUP.

Satz 11.29 (Wells, 1994) *Gegeben einfache Typen* A_1, A_2, B_1, B_2 *mit Variablen* $\vec{X}$*. Seien* Y_1, Y_2 *zwei neue Typvariablen und*

$$T = (B_1 \to Y_1) \to (Y_2 \to B_2) \to A_1 \to A_2$$

Das SUP $\{(A_1, B_1), (A_2, B_2)\}$ *ist lösbar gdw.* $\Gamma \vdash z\,(\lambda x.\, y\, x\, x) : o$ *für*

$$\Gamma = z : \forall X.\, (X \to X) \to o,\ y : \forall \vec{X}, Y_1, Y_2.\, T$$

Beweis. Es genügt, eine syntaxgerichtete Typisierung zu betrachten. Diese existiert gdw. es einen Typen C gibt mit

$$\frac{\Gamma \vdash z : (C \to C) \to o \qquad \Gamma \vdash \lambda x.\, y\, x\, x : C \to C}{\Gamma \vdash z\,(\lambda x.\, y\, x\, x) : o}$$

Es muss also C geben mit $\Gamma' = (\Gamma, x{:}C) \vdash y\, x\, x : C$. Dies tritt genau dann ein wenn es auch noch eine Substitution ξ mit $\mathsf{dom}(\xi) = \vec{X}, Y_1, Y_2$ so dass $y : T\xi$ die Typisierung von $y\, x\, x$ impliziert, also

$$\begin{array}{lll} C & <:_{\Gamma'} & (B_1 \to Y_1)\xi \\ C & <:_{\Gamma'} & (Y_2 \to B_2)\xi \\ (A_1 \to A_2)\xi & <:_{\Gamma'} & C \end{array}$$

Die freien Variablen von Γ' sind genau die freien Variablen von C, also gilt die letzte Ungleichung gdw. es Variablen $\vec{Z} \subseteq \mathsf{FV}(\xi)$ gibt, so dass $C = \forall \vec{Z}.\, (A_1 \to A_2)\xi$. Die ersten beiden Gleichungen sind genau dann erfüllt, wenn

$$\begin{array}{lcl} (A_1 \to A_2)\xi\xi_1 & = & (B_1 \to Y_1)\xi \\ (A_1 \to A_2)\xi\xi_2 & = & (Y_2 \to B_2)\xi \end{array}$$

für zwei Substitutionen ξ_1, ξ_2 mit Definitionsbereich $\vec{Z}$. Dies ist äquivalent zu

$$\begin{array}{rcl} A_2\xi\xi_1 & = & Y_1\xi \\ A_1\xi\xi_2 & = & Y_2\xi \end{array}$$

$$\begin{array}{rcl} A_1\xi\xi_1 & = & B_1\xi \\ A_2\xi\xi_2 & = & B_2\xi. \end{array}$$

Die ersten beiden Gleichungen sind immer erfüllbar durch entsprechende Setzungen von Y_1 und Y_2 in ξ, und diese Setzungen beeinflussen die Erfüllbarkeit der letzen beiden Gleichungen nicht, da Y_1 und Y_2 in den anderen Typen nicht vorkommen.

Daraus folgt: Ist das SUP lösbar, so gibt es eine Typherleitung. Gibt es eine Typherleitung so gibt es Substitutionen ξ, ξ_1, ξ_2 die die obigen Gleichungen lösen. Das ist fast eine Lösung des SUP, nur dass die Substitutionen noch polymorphe Typen enthalten können. Wir können jedoch alle Quantoren aus den Substitutionen auslöschen, wobei wir alle gebundenen Variablen durch o ersetzen. Dies modifizierten Substitutionen lösen das SUP noch immer.

Das Typisierungsproblem ist also zu einem SUP äquivalent, und dieses ist unentscheidbar. □

Kapitel 12

Modelle

Definition 12.1 (Applikative Struktur) *Eine* applikative Struktur $(M, \cdot)$ *ist eine Menge M mit einer linksassoziativen Operation $\cdot : M \times M \to M$. Eine applikative Struktur ist* extensional *falls gilt:*

$$\forall d \in M.\ f \cdot d = g \cdot d \implies f = g.$$

Definition 12.2 (Terme) *Sei M eine applikative Struktur und V eine abzählbar unendliche Menge von Variablen. Die Terme $\mathcal{T}(M)$ über M sind gegeben durch folgende Grammatik:*

$$\begin{array}{rcll} T(M) \ni a, b & ::= & x & \textit{Variable } x \in \mathsf{V} \\ & | & c_d & \textit{Konstante für } d \in M \\ & | & a\,b & \textit{Applikation} \end{array}$$

Gewöhnlich schreiben wir einfach d anstatt c_d. Substitution $a[b/x]$ und freie Variablen $\mathsf{FV}(a)$ seien für $\mathcal{T}(M)$ definiert wie für Λ.

Definition 12.3 (Interpretation) *Sei $a \in \mathcal{T}(M)$ ein Term über M und $\rho \in \mathsf{V} \to M$ eine* Belegung *der Variablen durch Elemente in M (eine* Umgebung*). Die Interpretation $[\![a]\!]_\rho$ des Termes a in Umgebung ρ ist definiert durch Rekursion über a:*

$$\begin{array}{lcl} [\![x]\!]_\rho & = & \rho(x) \\ [\![c_d]\!]_\rho & = & d \\ [\![a\,b]\!]_\rho & = & [\![a]\!]_\rho \cdot [\![b]\!]_\rho \end{array}$$

Wir schreiben $M, \rho \models a = b$ falls $[\![a]\!]_\rho = [\![b]\!]_\rho$ in M und $M \models a = b$ falls $M, \rho \models a = b$ für alle Belegungen ρ. Ist klar, welches M gemeint ist, schreiben wir auch $\rho \models a = b$ und $\models a = b$.

Lemma 12.4 (Gesetze der Interpretation)

$$\begin{array}{lcll} [\![a[b/x]]\!]_\rho & = & [\![a]\!]_{\rho[x \mapsto [\![b]\!]_\rho]} & \\ [\![a]\!]_\rho & = & [\![a]\!]_{\rho'} & \textit{falls } \rho(x) = \rho'(x) \textit{ für alle } x \in \mathsf{FV}(a) \end{array}$$

Beweis. Induktion über a. □

Definition 12.5 (Kombinatorische Algebra) *Eine* kombinatorische Algebra $(M, \cdot, \mathsf{k}, \mathsf{s})$ *ist eine applikative Struktur mit zwei ausgezeichneten Elementen* $\mathsf{k}, \mathsf{s} \in M$*, die für alle* $d, e, f, g \in M$ *folgende Gleichungen erfüllen:*

$$\begin{aligned} \mathsf{k} \cdot d \cdot e &= d \\ \mathsf{s} \cdot f \cdot g \cdot d &= (f \cdot d) \cdot (g \cdot d) \end{aligned}$$

Wir definieren $\mathsf{i} := \mathsf{s} \cdot \mathsf{k} \cdot \mathsf{k}$.

Für alle $d \in M$ gilt $i \cdot d = d$.

Definition 12.6 (bracket abstraction) *Sei* $a \in \mathcal{T}(M)$ *und* $x \in \mathsf{V}$*. Die* Abstraktion $[x]a \in \mathcal{T}(M)$ *von* x *in* a *ist definiert durch Rekursion über* a*:*

$$\begin{array}{lcll} [x]x & = & \mathsf{i} & \\ [x]a & = & \mathsf{k}\, a & \text{falls } x \notin \mathsf{FV}(a) \\ [x](a\, b) & = & \mathsf{s}\,([x]a)\,([x]b) & \text{falls } x \in \mathsf{FV}(a\, b) \end{array}$$

Lemma 12.7 (Rechenregeln für die Abstraktion)

1. *Falls* $y \notin \mathsf{FV}(b)$*, dann* $([y]a)[b/x] = [y](a[b/x])$.
2. $[\![[x]a]\!]_\rho \cdot d = [\![a]\!]_{\rho[x \mapsto d]}$.

Beweis. Jeweils durch Induktion über a. □

Korollar 12.8 $\models ([x]a)\, b = a[b/x]$.

Wir definieren nun die Abbildung $\ulcorner \cdot \urcorner : \Lambda \to \mathcal{T}(M)$. In $\ulcorner t \urcorner$ setzen wir für t voraus, dass alle gebundenen Variablen nur an einem λ gebunden werden.

$$\begin{array}{lcl} \ulcorner x \urcorner & = & x \\ \ulcorner r\, s \urcorner & = & \ulcorner r \urcorner \ulcorner s \urcorner \\ \ulcorner \lambda x t \urcorner & = & \ulcorner [x] t \urcorner \end{array}$$

Eine Interpretation von Lambda-Termen in einer komb.Alg. M ist nun gegeben durch $[\![t]\!]_\rho := [\![\ulcorner t \urcorner]\!]_\rho$.

Lemma 12.9 (Gesetze der Interpretation)

$$\begin{array}{lcll} [\![x]\!]_\rho & = & \rho(x) & \\ [\![r\, s]\!]_\rho & = & [\![r]\!]_\rho \cdot [\![s]\!]_\rho & \\ [\![\lambda x t]\!]_\rho \cdot d & = & [\![t]\!]_{\rho[x \mapsto d]} & \\ [\![t[s/x]]\!]_\rho & = & [\![t]\!]_{\rho[x \mapsto [\![s]\!]_\rho]} & \\ [\![t]\!]_\rho & = & [\![t]\!]_{\rho'} & \text{falls } \rho(x) = \rho'(x) \text{ für alle } x \in \mathsf{FV}(t) \end{array}$$

12.1 Zusammenfassung

Struktur	modelliert
applikative Struktur	Applikation
extensionale a. S.	App., Ext.
kombinatorische Algebra	schwache Kopfgleichheit (β, Substitution, aber nicht ξ)

Tabelle 12.1: Modellbegriffe zunehmender Stärke

Literaturverzeichnis

H. Barendregt (1984). *The Lambda Calculus: Its Syntax and Semantics*. North Holland, Amsterdam.

N. G. de Bruijn (1972). 'Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem'. *Indagationes Mathematicae* **34**:381–392.

M. Gabbay & A. M. Pitts (1999). 'A New Approach to Abstract Syntax Involving Binders'. In *Logics in Computer Science (LICS'99)*, pp. 214–224.

J.-Y. Girard, et al. (1989). *Proofs and Types*, vol. 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

A. D. Gordon (1994). 'A Mechanisation of Name-Carrying Syntax up to Alpha-Conversion'. In J. J. Joyce & C.-J. H. Seger (eds.), *Higher Order Logic Theorem Proving and its Applications, 6th International Workshop, HUG '93, Vancouver, BC, Canada, August 11-13, 1993, Proceedings*, vol. 780 of *Lecture Notes in Computer Science*, pp. 413–425. Springer-Verlag.

R. Harper, et al. (1993). 'A Framework for Defining Logics'. *Journal of the Association of Computing Machinery* **40**(1):143–184.

J. W. Klop (1980). 'Combinatory Reduction Systems'. *Mathematical Center Tracts* **27**.

R. Loader (1998). 'Notes on Simply Typed Lambda Calculus'. Tech. rep., Laboratory for Foundations of Computer Science, University of Edinburgh.

B. C. Pierce (2002). *Types and Programming Languages*. MIT Press.

G. Plotkin (1975). 'Call-by-name, Call-by-value, and the λ-Calculus'. *Theoretical Computer Science* **1**:125–159.

P. Selinger (2006). 'Lecture Notes on the Lambda Calculus'. Available on the author's homepage.

M. Takahashi (1995). 'Parallel reductions in λ-calculus'. *Information and Computation* **118**(1):120–127.