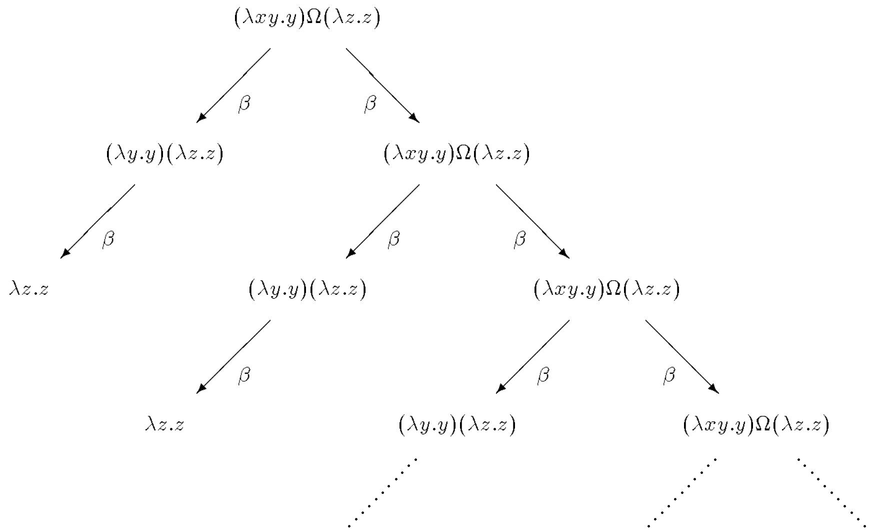
Monomorphe Typsysteme des λ-Kalküls

Überarbeitete Version der Diplomarbeit von Michael Marz
zuletzt aktualisiert: 13. Oktober 1995

Betreuer: Dr. A. Jung
Fachbereich Mathematik



Technische Hochschule Darmstadt
Juni 1995

Inhaltsverzeichnis

Einleitung

Welche Typsysteme des λ-Kalküls gewährleisten starke Normalisierbarkeit? Diese Frage hat eine zentrale Bedeutung in der Theorie des λ-Kalküls. Auf die Funktionale Programmierung übertragen, entspricht starke Normalisierbarkeit genau der Terminierung von Programmen. Für zahlreiche Typsysteme ist diese Frage bereits positiv beantwortet. N. Mendler untersucht in [Men91] in diesem Zusammenhang Systeme mit rekursiven Typen und speziellen Typgleichungen. Seine Ergebnisse liegen dieser Arbeit zugrunde.

Das erste Kapitel dieser Arbeit gibt eine kurze Einführung in die Theorie des ungetypten λ-Kalküls. Im zweiten Kapitel werden Typrelationen auf monomorphen Typsystemen in einer allgemeinen Form eingeführt und grundlegende Eigenschaften wie beispielsweise Typtreue („subject reduction") untersucht. Zu diesem Zweck werden K-Typsysteme eingeführt; die dazu benötigten Grundbegriffe aus der Allgemeinen Algebra werden am Anfang des Kapitels bereitgestellt. Im dritten Kapitel wird eine graphentheoretische Charakterisierung von Mendlers Kriterium zur starken Normalisierbarkeit gegeben und dieses Ergebnis auf den allgemeinen Fall übertragen. Durch den allgemeinen Ansatz ist es möglich, auch das Entscheidbarkeitsproblem der Typisierung für normalisierende K-Typsysteme positiv zu beantworten.

An dieser Stelle möchte ich denjenigen danken, die mir während meines Studiums und bei der Erstellung der Arbeit geholfen haben. Hervorheben möchte ich hierbei Prof. Dr. Klaus Keimel, der mich durch seine engagierten Vorlesungen mathematisch stark beeinflußt hat.

Desweiteren möchte ich Prof. Dr. Paweł Urzyczyn aus Warschau für sein großes Interesse und einige nützliche Hinweise danken.

Ein besonderer Dank geht an Mathias Kegelmann für die langen und leider manchmal fruchtlosen Diskussionen sowie für sein penibles Korrekturlesen der Arbeit.

Schließlich möchte ich mich bei Dr. Achim Jung für die hervorragende Betreuung bedanken, bei der er in schwierigen Phasen großes Verständnis zeigte und durch zahlreiche Ideen stets neue Impulse gab. Durch ihn war es mir möglich, die Arbeit auch mit ausländischen Wissenschaftlern zu diskutieren.

Darmstadt, 19. Juni 1995 Michael Marz

1 Ungetypter λ-Kalkül

Der λ-Kalkül wurde 1932 von A. CHURCH entwickelt. CHURCH verfolgte ein neues Konzept zur Begründung der Mathematik und benutzte den λ-Kalkül als einfache Darstellungsform für Funktionen. Sein Grundkonzept scheiterte, dennoch hat sich der λ-Kalkül zu einer eigenständigen, ausdrucksstarken Theorie mit Anwendungen in der *Berechenbarkeitstheorie* und in letzter Zeit zunehmend in der *Funktionalen Programmierung* entwickelt. [Bar92] liefert einen guten Überblick über die Grundlagen des λ-Kalküls, eine ausführliche Einführung findet man in [HS86]. Für weitergehende Ergebnisse siehe [Bar84].

In diesem Kapitel werden die Grundbegriffe des λ-Kalküls und einige Eigenschaften erklärt, die zum Verständnis der Arbeit nötig sind.

Im Umgang mit den in der Mathematik üblichen Symbolen zur Bezeichnung von Funktionen entstehen häufig Probleme, wenn Funktionale höherer Ordnung auftreten. Zu jedem Funktional müssen stets Definitions- und Wertebereich angegeben werden. Die λ-Schreibweise vermeidet dies, indem prinzipiell jede Funktion auf alles angewendet werden kann. Durch Einführung von *Typsystemen* wird dieser Willkür vorgebeugt, dies ist Gegenstand der folgenden Kapitel.

Ab sofort sei X eine abzählbar unendliche Variablenmenge.

Definition 1.1 *Die Menge Λ aller λ-Terme ist induktiv wie folgt definiert:*

$$\begin{array}{rcll} x \in X & \Rightarrow & x \in \Lambda & \\ s, t \in \Lambda & \Rightarrow & (st) \in \Lambda & \textit{(Applikation)} \\ x \in X,\, t \in \Lambda & \Rightarrow & (\lambda x.t) \in \Lambda & \textit{(Abstraktion)} \end{array}$$

Ein λ-Term der Form (st) kann daher gelesen werden als: wende die Funktion s auf das Argument t an. Der Term $(\lambda x.t)$ stellt eine Funktion dar, die x auf t abbildet, wobei t von x abhängen darf. Oft lassen wir die äußeren Klammern eines λ-Terms einfach weg, wir schreiben stu statt $((st)u)$ und $\lambda x.st$ statt $(\lambda x.(st))$. Desweiteren benutzen wir $\lambda x_1 x_2 \ldots x_n.t$ als Kurzschreibweise für $\lambda x_1.\lambda x_2.\ldots.\lambda x_n.t$. Die **Länge** $\mathrm{len}(t)$ eines λ-Terms t ist induktiv definiert durch

$$\mathrm{len}(t) = \begin{cases} 1 & \text{falls } t \in X \\ \mathrm{len}(t_1) + \mathrm{len}(t_2) & \text{falls } t = t_1 t_2 \\ \mathrm{len}(t_1) + 1 & \text{falls } t = \lambda x.t_1 \end{cases}$$

Definition 1.2 *Das Vorkommen einer Variablen $x \in X$ in einem λ-Term t heißt **gebunden**, falls es in einem in t enthaltenen λ-Term der Form $\lambda x.s$ enthalten ist,* *ansonsten heißt es* ***frei.*** BV(t) *bezeichnet die Menge aller Variablen, die in* t *gebunden vorkommen,* FV(t) *ist die Menge aller in* t *frei vorkommenden Variablen.*

Beispiel: Im λ-Term $t := \lambda y.zy(\lambda z.zx)$ kommt x nur frei, y nur gebunden und z sowohl frei als auch gebunden vor.

Definition 1.3 *Seien* s, t λ*-Terme und* $x \in X$ *eine Variable. Die* ***Substitution*** $t[s/x]$ *(„s für x in t") ist induktiv definiert:*

- $t = x$:
  $x[s/x] := s$
- $t = y \neq x,\ y \in X$:
  $y[s/x] := y$
- $t = t_1t_2$:
  $(t_1t_2)[s/x] := (t_1[s/x])(t_2[s/x])$
- $t = \lambda x.t_1$:
  $(\lambda x.t_1)[s/x] := \lambda x.t_1$
- $t = \lambda y.t_1,\ y \neq x$:
  $(\lambda y.t_1)[s/x] := \lambda z.(t_1[z/y][s/x])$ *für ein* $z \in X$ *mit* $z \notin \mathrm{FV}(s) \cup \mathrm{FV}(t_1)$

Weil im letzten Fall die Wahl von z willkürlich ist, ist die Substitution nicht eindeutig festgelegt. Dies ist aber kein großer Nachteil. λ-Terme werden als Funktionen betrachtet, somit ist die Bezeichnung der Funktionsargumente an sich unerheblich.

Definition 1.4 *Falls zwei* λ*-Terme* s *und* t *sich nur durch die Benennung gebundener Variablen unterscheiden, so heißen sie* α***-äquivalent****, wir schreiben:* $s \underset{\alpha}{=} t$.

Man überprüft leicht, daß die α-Äquivalenz tatsächlich eine Äquivalenzrelation ist, auf deren Klassen die Substitution wohldefiniert ist. Für den Rest des Kapitels fassen wir jeden λ-Term als Repräsentant seiner α-Äquivalenzklasse auf und schreiben beispielsweise $\lambda x.xz = \lambda y.yz$.

Um wirklich sinnvoll im λ-Kalkül „rechnen" zu können, muß sich ein λ-Term der Form $(\lambda x.t)s$ zu $t[s/x]$ auswerten lassen. Dies leistet die β-Reduktion:

Definition 1.5 *Ein β-**Redex** ist ein λ-Term der Form $(\lambda x.t)s$, der Term $t[s/x]$ heißt sein **Kontraktum**. Ist u' derjenige λ-Term, den man erhält, wenn in $u \in \Lambda$ das Auftreten eines β-Redex durch sein Kontraktum ersetzt wird, so ist $u \xrightarrow[\beta]{} u'$ eine β-**Kontraktion**. Eine β-**Reduktion** $u \twoheadrightarrow_{\beta} u'$ ist eine endliche (möglicherweise leere) Folge von β-Kontraktionen.*

Die β-Reduktion ist mit der Substitution verträglich, denn es gilt:

Lemma 1.6 *Seien $s, t, u \in \Lambda$ und $x \in X$ mit $s \twoheadrightarrow_{\beta} t$. Dann gilt $u[s/x] \twoheadrightarrow_{\beta} u[t/x]$ und $s[u/x] \twoheadrightarrow_{\beta} t[u/x]$.*

Beweis: Jede β-Kontraktion in s kann auch in $u[s/x]$ durchgeführt werden, daraus folgt $u[s/x] \twoheadrightarrow_{\beta} u[t/x]$. Zum Beweis des zweiten Teils der Behauptung zeigen wir $((\lambda y.w)v)[u/x] \xrightarrow[\beta]{} w[v/y][u/x]$. Es gilt:

$$
\begin{aligned}
((\lambda y.w)v)[u/x] &= ((\lambda y.w)[u/x])(v[u/x]) \\
&= (\lambda z.(w[z/y][u/x]))(v[u/x])
\end{aligned}
$$

für ein $z \in X$, $z \notin \mathrm{FV}(u) \cup \mathrm{FV}(w)$ nach Definition der Substitution. Weiter folgt

$$
\begin{aligned}
(\lambda z.(w[z/y][u/x]))(v[u/x]) &\xrightarrow[\beta]{} w[z/y][u/x][v[u/x]/z] \\
&= w[z/y][v/z][u/x] \qquad (\text{da } z \notin \mathrm{FV}(u)) \\
&= w[v/y][u/x] \qquad (\text{da } z \notin \mathrm{FV}(w)).
\end{aligned}
$$

□

Es gibt noch weitere Reduktionsregeln im λ-Kalkül:

Definition 1.7 *Ein η-**Redex** ist ein λ-Term der Form $\lambda x.tx$ mit $x \notin \mathrm{FV}(t)$, der Term t heißt sein **Kontraktum**. Ist u' derjenige λ-Term, den man erhält, wenn in $u \in \Lambda$ das Auftreten eines η-Redex durch sein Kontraktum ersetzt wird, so ist $u \xrightarrow[\eta]{} u'$ eine η-**Kontraktion**. Eine $\beta\eta$-**Reduktion** $u \twoheadrightarrow_{\beta\eta} u'$ ist eine endliche (möglicherweise leere) Folge von β- und η-Kontraktionen.*

Diese Definition wird dadurch motiviert, daß Funktionen miteinander identifiziert werden sollen, wenn sie zu gleichen Eingaben gleiche Ergebnisse liefern. Dies ist für $\lambda x.tx$ und t mit $x \notin \mathrm{FV}(t)$ der Fall, denn es gilt:

$$(\lambda x.tx)s \xrightarrow[\beta]{} (tx)[s/x] = (t[s/x])(x[s/x]) = ts.$$

Folgendes Lemma zeigt, daß auch β- und $\beta\eta$-Reduktionen mit der Substitution verträglich sind:

Lemma 1.8 *Seien* $s, t, u \in \Lambda$ *und* $x \in X$ *mit* $s \underset{\beta\eta}{\twoheadrightarrow} t$. *Dann gilt* $u[s/x] \underset{\beta\eta}{\twoheadrightarrow} u[t/x]$ *und* $s[u/x] \underset{\beta\eta}{\twoheadrightarrow} t[u/x]$.

Beweis: Jede η-Kontraktion in s kann auch in $u[s/x]$ durchgeführt werden, mit Lemma 1.6 folgt $u[s/x] \underset{\beta\eta}{\twoheadrightarrow} u[t/x]$. Zum Beweis des zweiten Teils der Aussage zeigen wir $(\lambda y.vy)[u/x] \xrightarrow[\eta]{} v[u/x]$ für $y \notin \mathrm{FV}(v)$. Nach Definition ist

$$\begin{aligned}(\lambda y.vy)[u/x] &= \lambda z.((vy)[z/y][u/x]) \\ &= \lambda z.(v[u/x])z\end{aligned}$$

mit $z \notin \mathrm{FV}(u) \cup \mathrm{FV}(v) \cup \{y\}$. Da auch $z \notin \mathrm{FV}(v[u/x])$, gilt $(\lambda y.vy)[u/x] \xrightarrow[\eta]{} v[u/x]$. Mit Lemma 1.6 folgt $s[u/x] \underset{\beta\eta}{\twoheadrightarrow} t[u/x]$. $\square$

Sowohl die β- als auch die $\beta\eta$-Reduktion ist **konfluent**, das heißt, die Reihenfolge der Reduktionsschritte ist unerheblich. Der folgende Satz präzisiert diese Eigenschaft:

Satz 1.9 (Church-Rosser) *Für beliebige* λ*-Terme* s, t_1, t_2 *mit* $s \underset{\beta\eta}{\twoheadrightarrow} t_1$ *sowie* $s \underset{\beta\eta}{\twoheadrightarrow} t_2$ *gibt es einen* λ*-Term* u *mit* $t_1 \underset{\beta\eta}{\twoheadrightarrow} u$ *und* $t_2 \underset{\beta\eta}{\twoheadrightarrow} u$ *(analog für* $\underset{\beta}{\twoheadrightarrow}$ *).*

$$\begin{tikzcd}
 & s \arrow[dl, two heads, "\beta\eta"] \arrow[dr, two heads, "\beta\eta"'] & \\
t_1 \arrow[dr, dotted, two heads, "\beta\eta"'] & & t_2 \arrow[dl, dotted, two heads, "\beta\eta"] \\
 & u &
\end{tikzcd}$$

Beweis: Wir geben die Skizze eines Beweises, der auf P. Martin-Löf und W. Tait zurückgeht, siehe [Tak92]. Dazu benötigen wir den Begriff der parallelen Reduktion $\Rightarrow$, die wie folgt definiert ist:

- $x \Rightarrow x$ für $x \in X$
- $st \Rightarrow s't'$, falls $s \Rightarrow s'$ und $t \Rightarrow t'$
- $\lambda x.t \Rightarrow \lambda x.t'$, falls $t \Rightarrow t'$
- $(\lambda x.t)s \Rightarrow t'[s'/x]$, falls $s \Rightarrow s'$ und $t \Rightarrow t'$

- $\lambda x.tx \Rightarrow t'$, falls $x \notin \mathrm{FV}(t)$ und $t \Rightarrow t'$

Die $\beta\eta$-Reduktion ist die transitive Hülle der parallelen Reduktion $\Rightarrow$. Daher genügt es zum Beweis von Satz 1.9 zu zeigen, daß aus $s \Rightarrow t_1$ und $s \Rightarrow t_2$ auch schon die Existenz eines λ-Terms u mit $t_1 \Rightarrow u$ und $t_2 \Rightarrow u$ folgt.

Zu jedem λ-Term t sei t^* wie folgt definiert:

- $x^* = x$ für $x \in X$
- $(st)^* = s^*t^*$, falls st kein β-Redex ist
- $(\lambda x.t)^* = \lambda x.t^*$, falls $\lambda x.t$ kein η-Redex ist
- $((\lambda x.t)s)^* = t^*[s^*/x]$
- $(\lambda x.tx)^* = t^*$, falls $x \notin \mathrm{FV}(t)$

Mit Induktion über den Aufbau von s zeigt man, daß für ein beliebiges $t \in \Lambda$ mit $s \Rightarrow t$ auch $t \Rightarrow s^*$ gilt. Daraus folgt die Konfluenz von $\Rightarrow$, was den Beweis vollendet. $\square$

2 Freie Gruppoide als Typsysteme des λ-Kalküls

Im ungetypten λ-Kalkül gibt es keine Einschränkungen für den Definitionsbereich von Termen. Insbesondere kann jeder λ-Term sogar auf sich selbst angewandt werden. Typsysteme bilden eine natürliche Form der Regulierung dieser Willkür: eine Funktion kann nur auf Argumente eines entsprechenden Typs angewandt werden. Dies hat zur Folge, daß zu einem gegebenen Typsystem nicht jeder λ-Term typisierbar ist, andererseits haben die mit geeigneten Typsystemen getypten λ-Terme schöne Eigenschaften, die wir in Kapitel 3 genauer untersuchen werden.

In diesem Kapitel sollen Typsysteme untersucht werden, die nur aus Grundtypen und Funktionentypen bestehen. Diese stellen die einfachsten Typsysteme der **monomorphen Systeme** dar, die im Gegensatz zu **polymorphen Systemen** wie beispielsweise GIRARDs System F keine Typquantoren besitzen. Daher liegt es nahe, für Typsysteme Gruppoide zu benutzen. Dazu benötigen wir einige Begriffe und Resultate aus der *Allgemeinen Algebra*, die wir in die Sprache der Gruppoidtheorie übersetzen. Für eine ausführliche Einführung siehe z. B. [Ihr88].

Definition 2.1 *Ein* ***Gruppoid*** $\underline{\mathcal{T}} = (\mathcal{T}, \rightarrow)$ *ist eine Menge* $\mathcal{T}$ *mit einer zweistelligen Operation* $\rightarrow$*, wobei mit* $\sigma, \tau \in \mathcal{T}$ *auch* $\sigma \rightarrow \tau \in \mathcal{T}$ *gilt. Die Elemente von* $\mathcal{T}$ *heißen* ***Typen****. Wir verwenden die übliche Konvention:* $\tau_1 \rightarrow \tau_2 \rightarrow \tau_3$ *steht für* $\tau_1 \rightarrow (\tau_2 \rightarrow \tau_3)$*. Eine* ***Kongruenzrelation*** *(oder kurz:* ***Kongruenz****) auf einem Gruppoid* $\underline{\mathcal{T}}$ *ist eine Relation* $\approx$ *auf* $\mathcal{T}$ *mit:*

$$\begin{array}{ll}
\forall \tau \in \mathcal{T}:\ \tau \approx \tau & \textit{(Reflexivität)} \\
\forall \sigma, \tau \in \mathcal{T}:\ \sigma \approx \tau \Rightarrow \tau \approx \sigma & \textit{(Symmetrie)} \\
\forall \sigma, \tau, \rho \in \mathcal{T}:\ \sigma \approx \tau, \tau \approx \rho \Rightarrow \sigma \approx \rho & \textit{(Transitivität)} \\
\forall \sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{T}:\ \sigma_1 \approx \tau_1, \sigma_2 \approx \tau_2 \ \Rightarrow\ \sigma_1 \rightarrow \sigma_2 \approx \tau_1 \rightarrow \tau_2 & \textit{(Verträglichkeit)}
\end{array}$$

Die Menge aller Kongruenzen auf $\underline{\mathcal{T}}$ heißt **Con** $\underline{\mathcal{T}}$. Auf jedem Gruppoid $\mathcal{T}$ sind die Relationen Δ und ∇ mit

$$\sigma \Delta \tau \ \Leftrightarrow\ \sigma = \tau \text{ und}$$

$$\sigma \nabla \tau \text{ für alle } \sigma, \tau \in \mathcal{T}$$

stets Kongruenzen.

Zu einer Kongruenzrelation $\approx$ auf einem Gruppoid $\underline{\mathcal{T}} = (\mathcal{T}, \rightarrow)$ nennt man die Menge $[\tau]_\approx := \{\xi \in \mathcal{T} \mid \xi \approx \tau\}$ eine **Kongruenzklasse** auf $\underline{\mathcal{T}}$. Wegen der Verträglichkeit von $\approx$ mit der Operation $\rightarrow$ ist $\rightarrow_\approx$ mit $[\sigma]_\approx \rightarrow_\approx [\tau]_\approx = [\sigma \rightarrow \tau]_\approx$ auf der Menge $\mathcal{T}/_{\approx} := \{[\tau]_{\approx} \mid \tau \in \mathcal{T}\}$ wohldefiniert. Daher ist auch $\underline{\mathcal{T}}/_{\approx} = (\mathcal{T}/_{\approx}, \rightarrow_{\approx})$ ein Gruppoid, das sogenannte **Faktor-** oder **Quotientengruppoid**. Die Abbildung $\pi_{\approx}\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{T}}/_{\approx}$ mit $\pi_{\approx}(\tau) = [\tau]_{\approx}$ heißt **kanonische Quotientenabbildung**.

Lemma 2.2 *Jede Relation $\sim$ auf einem Gruppoid $\underline{\mathcal{T}}$ läßt sich eindeutig zu einer kleinsten Kongruenzrelation $\approx$ mit $\sigma \sim \tau \Rightarrow \sigma \approx \tau$ fortsetzen.*

Beweis: Der Schnitt von Kongruenzrelationen ist wieder eine Kongruenzrelation. Die gesuchte Kongruenz ist also gerade der Schnitt aller Kongruenzen, die $\sim$ enthalten. □

Im folgenden sei zu einer gegebenen Relation $\sim$ mit $\approx$ stets die von $\sim$ erzeugte Kongruenz bezeichnet. Gilt $\sigma \approx \tau$ für $\sigma, \tau \in \mathcal{T}$ und eine Relation $\sim$, so gibt es dafür eine Herleitung in folgendem Kalkül:

$$\frac{}{\sigma \approx \tau}\,(\mathrm{c}) \quad \text{falls } \sigma \sim \tau \qquad\qquad \frac{}{\tau \approx \tau}\,(\mathrm{r}) \quad \text{für alle } \tau \in \mathcal{T}$$

$$\frac{\sigma \approx \tau}{\tau \approx \sigma}\,(\mathrm{s}) \qquad \frac{\sigma \approx \tau \qquad \tau \approx \rho}{\sigma \approx \rho}\,(\mathrm{t}) \qquad \frac{\sigma \approx \sigma' \qquad \tau \approx \tau'}{\sigma \rightarrow \tau \approx \sigma' \rightarrow \tau'}\,(\mathrm{v})$$

Definition 2.3 *Ein **(Gruppoid-) Homomorphismus** ist eine Abbildung $\varphi\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{S}}$ zwischen zwei Gruppoiden $(\underline{\mathcal{T}}, \rightarrow_{\mathcal{T}})$, $(\underline{\mathcal{S}}, \rightarrow_{\mathcal{S}})$, so daß gilt:*

$$\textit{(Hom)} \quad \forall \tau, \tau' \in \underline{\mathcal{T}}\colon\; \varphi(\tau \rightarrow_{\mathcal{T}} \tau') = \varphi(\tau) \rightarrow_{\mathcal{S}} \varphi(\tau')$$

*Existiert die Umkehrabbildung $\varphi^{-1}\colon \underline{\mathcal{S}} \rightarrow \underline{\mathcal{T}}$, so erfüllt auch diese die Homomorphiebedingung. In diesem Fall ist φ ein **(Gruppoid-) Isomorphismus**, und die Gruppoide $\underline{\mathcal{S}}$ und $\underline{\mathcal{T}}$ heißen zueinander **isomorph**, in Zeichen: $\underline{\mathcal{S}} \cong \underline{\mathcal{T}}$.*

Die kanonische Quotientenabbildung $\pi_\theta\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{T}}/_\theta$ und die Identität $\mathrm{id}_{\mathcal{T}}\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{T}}$ mit $\mathrm{id}_{\mathcal{T}}(\tau) = \tau$ sind stets Homomorphismen. Auch die Komposition $\varphi \circ \psi\colon \underline{\mathcal{R}} \rightarrow \underline{\mathcal{T}}$ zweier Homomorphismen $\varphi\colon \underline{\mathcal{S}} \rightarrow \underline{\mathcal{T}}$ und $\psi\colon \underline{\mathcal{R}} \rightarrow \underline{\mathcal{S}}$ ist wieder ein Homomorphismus.

Wir werden sehen, daß der Begriff der Kongruenz mit dem der Homomorphie nahe verwandt ist. Dazu definieren wir den **Kern** eines Homomorphismus $\varphi\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{S}}$ als $\ker(\varphi) = \{(\tau, \tau') \in \mathcal{T} \times \mathcal{T} \mid \varphi(\tau) = \varphi(\tau')\}$. Für jeden Homomorphismus $\varphi\colon \underline{\mathcal{T}} \rightarrow \underline{\mathcal{S}}$ ist $\ker(\varphi)$ eine Kongruenzrelation, es gilt sogar:

Satz 2.4 (Homomorphiesatz) *Sei* $\varphi\colon \underline{\mathcal{T}} \to \varphi(\underline{\mathcal{T}})$ *ein Homomorphismus. Dann gibt es einen Isomorphismus* $\overline{\varphi}\colon \underline{\mathcal{T}}/_{\ker(\varphi)} \to \varphi(\underline{\mathcal{T}})$ *mit* $\overline{\varphi} \circ \pi_{\ker(\varphi)}(\tau) = \varphi(\tau)$ *für alle* $\tau \in \underline{\mathcal{T}}$.

Beweis: Man rechne nach, daß $\overline{\varphi}\colon \underline{\mathcal{T}}/_{\ker(\varphi)} \to \varphi(\underline{\mathcal{T}})$ mit $\overline{\varphi}([\tau]_{\ker(\varphi)}) = \varphi(\tau)$ wohldefiniert, bijektiv und ein Homomorphismus ist. $\square$

Definition 2.5 *Sei* $\mathcal{J}$ *eine beliebige Menge. Ihre Elemente wollen wir* ***Grundtypen*** *nennen. Die Grammatik*

$$\tau ::= \iota \,|\, \tau \to \tau \quad (\iota \text{ beliebig aus } \mathcal{J})$$

definiert die Menge $\mathcal{F}_{\mathcal{J}}$ *aller Typen über* $\mathcal{J}$. *Zusammen mit der Operation* $\to$ *bildet* $\underline{\mathcal{F}}_{\mathcal{J}} = (\mathcal{F}_{\mathcal{J}}, \to)$ *ein Gruppoid.* $\underline{\mathcal{F}}_{\mathcal{J}}$ *heißt* ***Termgruppoid****,* ***von*** $\mathcal{J}$ ***frei erzeugt*** *oder einfach* ***frei****. Um Verwechslungen mit* λ*-Termen zu vermeiden, benutzen wir die Bezeichnung* ***freies Gruppoid****. Die* ***Länge*** $\mathrm{len}(\tau)$ *eines Typs* $\tau \in \mathcal{F}$ *(*$\underline{\mathcal{F}}$ *frei), ist gerade die Anzahl der Pfeile in* τ.

Für jedes $\mathcal{J}$ ist $\underline{\mathcal{F}}_{\mathcal{J}}$ bis auf Isomorphie eindeutig bestimmt. Es gilt sogar:

Satz 2.6 *Jedes Gruppoid* $\underline{\mathcal{T}}$ *ist homomorphes Bild eines freien Gruppoids.*

Beweis: Setze $\varphi\colon \underline{\mathcal{F}}_{\mathcal{T}} \to \underline{\mathcal{T}}$ mit $\varphi(\tau) := \tau$ und $\varphi(\sigma \to \tau) := \varphi(\sigma) \to_{\underline{\mathcal{T}}} \varphi(\tau)$ für alle $\sigma, \tau \in \mathcal{T}$. Dann ist φ der gesuchte Homomorphismus. $\square$

Mit dem Homomorphiesatz 2.4 folgt daraus:

Korollar 2.7 *Für jedes Gruppoid* $\underline{\mathcal{T}}$ *gibt es ein freies Gruppoid* $\underline{\mathcal{F}}$ *und eine Kongruenzrelation* $\approx$ *mit* $\underline{\mathcal{T}} \cong \underline{\mathcal{F}}/_{\approx}$.

Beweis: Wähle $\underline{\mathcal{F}} = \underline{\mathcal{F}}_{\mathcal{T}}$, φ wie im Beweis von Satz 2.6 und $\ker(\varphi)$ als Kongruenzrelation. $\square$

Korollar 2.7 erlaubt es, sich bei monomorphen Typsystemen des λ-Kalküls auf freie Gruppoide mit Kongruenzen zu beschränken. Dies verschafft Vorteile in der Darstellung getypter λ-Terme.

Definition 2.8 *Ein **Typsystem** ist ein Paar* $(\underline{\mathcal{F}}, \approx)$*, das aus einem freien Gruppoid* $\underline{\mathcal{F}}$ *und einer Kongruenzrelation* $\approx$ *besteht.*

Prinzipiell gibt es zwei Möglichkeiten, Typsysteme des λ-Kalküls zu betrachten. Im „CHURCH-Style" enthält dabei jeder λ-Term so viele Informationen wie möglich über seinen Typ. Dadurch ist die Frage, ob ein λ-Term einen gewissen Typ hat, auch in komplizierten Typsystemen immer entscheidbar. Im „CURRY-Style" betrachtet man Typen als Prädikate für ungetypte λ-Terme. Dies ermöglicht, Probleme der Typisierbarkeit von Termen zu untersuchen. Solche Fragen spielen insbesondere in der *Funktionalen Programmierung* eine wichtige Rolle. Da wir uns mit Typisierbarkeitsfragen beschäftigen, benutzen wir den „CURRY-Style".

Im folgenden sei X stets eine abzählbar unendliche Variablenmenge und $(\underline{\mathcal{F}}, \approx)$ ein Typsystem.

Definition 2.9 *Eine **Basis** ist eine endliche Menge* Γ *von Ausdrücken* $(x : \tau)$ *mit* $x \in X$ *und* $\tau \in \mathcal{F}$*, wobei es für jedes* $x \in X$ *höchstens ein* $\tau \in \mathcal{F}$ *mit* $(x : \tau) \in \Gamma$ *gibt.*

Für die Definition getypter λ-Terme betrachten wir folgenden Kalkül:

$$\frac{}{\Gamma \vdash x : \tau}\,(\mathrm{Var}) \qquad \text{falls } (x : \tau) \in \Gamma$$

$$\frac{\Gamma \vdash s : \tau \to \sigma \quad \Gamma \vdash t : \tau}{\Gamma \vdash st : \sigma}\,(\mathrm{App})$$

$$\frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x.t : \sigma \to \tau}\,(\mathrm{Abs})$$

$$\frac{\Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau}\,(\mathrm{Kon}) \qquad \text{falls } \sigma \approx \tau$$

wobei $\Gamma, x : \tau$ für $\Gamma \cup \{(x : \tau)\}$ steht und es kein $\sigma \in \mathcal{F} \setminus \{\tau\}$ mit $(x : \sigma) \in \mathcal{F}$ gibt. Anstelle von $\emptyset \vdash t : \tau$ schreibt man kurz $\vdash t : \tau$.

Definition 2.10 *Die im obigen Kalkül herleitbaren Ausdrücke* $\Gamma \vdash t : \tau$ *heißen **Urteile**. Ein bezüglich* $(\underline{\mathcal{F}}, \approx)$ ***getypter** λ**-Term** (kurz:* $\lambda_{(\underline{\mathcal{F}}, \approx)}$***-Term**) ist ein Ausdruck* $t : \tau$*, für den es eine Basis* Γ *gibt, so daß* $\Gamma \vdash t : \tau$ *ein Urteil ist.*

Getypte λ-Terme erfüllen folgende Eigenschaften, die Rückschlüsse über den Aufbau des Terms und damit über seine Typisierbarkeit zulassen:

Lemma 2.11 (Erzeugungslemma) *Sei* Γ *eine Basis mit* $\Gamma \vdash t : \tau$ *für* $t \in \Lambda$ *und* $\tau \in \mathcal{F}$. *Dann gilt:*

a) Ist $t = x \in X$, *so gibt es ein* $\tau_1 \in \mathcal{F}$ *mit* $\tau_1 \approx \tau$ *und* $(x : \tau_1) \in \Gamma$.

b) Ist $t = t_1 t_2$, *so gibt es ein* $\tau_1 \in \mathcal{F}$ *mit* $\Gamma \vdash t_1 : \tau_1 \to \tau$ *und* $\Gamma \vdash t_2 : \tau_1$.

c) Ist $t = \lambda x.t_1$, *so gibt es* $\tau_1, \tau_2 \in \mathcal{F}$ *mit* $\tau \approx \tau_2 \to \tau_1$ *und* $\Gamma, x : \tau_2 \vdash t_1 : \tau_1$.

Beweis: Wir beweisen alle drei Aussagen parallel mit Induktion über die Herleitung von $\Gamma \vdash t : \tau$, wobei wir nach der zuletzt angewandten Regel unterscheiden. Die Fälle (Var), (App) und (Abs) sind trivial. Im Fall

$$\frac{\Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau}\,(\text{Kon}) \qquad \text{mit } \sigma \approx \tau$$

unterscheiden wir nach der Struktur von t:

a) $t = x \in X$: Es gilt $\Gamma \vdash x : \sigma$ mit $\sigma \approx \tau$. Nach Induktionsannahme existiert ein $\tau_1 \in \mathcal{F}$ mit $\tau_1 \approx \sigma$ und $(x : \tau_1) \in \Gamma$. Wegen der Transitivität von $\approx$ folgt $\tau_1 \approx \tau$.

b) $t = t_1 t_2$: Es gilt $\Gamma \vdash t_1 t_2 : \sigma$ mit $\sigma \approx \tau$. Nach Induktionsannahme existiert ein τ_1 mit $\Gamma \vdash t_1 : \tau_1 \to \sigma$ und $\Gamma \vdash t_2 : \tau_1$. Wegen der Reflexivität und der Verträglichkeit von $\approx$ ist $\tau_1 \to \sigma \approx \tau_1 \to \tau$, also mit (Kon) auch $\Gamma \vdash t_1 : \tau_1 \to \tau$.

c) $t = \lambda x.t_1$: Es gilt $\Gamma \vdash \lambda x.t_1 : \sigma$ mit $\sigma \approx \tau$. Nach Induktionsannahme existieren $\tau_1, \tau_2 \in \mathcal{F}$ mit $\sigma \approx \tau_2 \to \tau_1$ und $\Gamma, x : \tau_2 \vdash t_1 : \tau_1$. Wegen der Symmetrie und der Transitivität von $\approx$ folgt $\tau \approx \tau_2 \to \tau_1$.

□

Es fällt auf, daß die Verträglichkeit von $\approx$ nur teilweise benutzt wurde, nämlich in der Form

$$\sigma \approx \tau \quad \Rightarrow \quad \tau' \to \sigma \approx \tau' \to \tau.$$

Jede andere Eigenschaft von $\approx$ wird zum Beweis des Erzeugungslemmas in vollem Umfang benötigt.

Das Erzeugungslemma charakterisiert die Herleitungen getypter λ-Terme. Viele Beweise im getypten λ-Kalkül werden mit Induktion über den Termaufbau geführt, fast immer kommt dabei das Erzeugungslemma zum Tragen. So auch beim Beweis des folgenden Lemmas, das die Eigenschaften von Typbasen beschreibt:

Lemma 2.12 (Basislemma) *Sei Γ eine Basis, $t \in \Lambda$ und $\tau \in \mathcal{F}$.*

a) Gilt $\Gamma \vdash t : \tau$ und ist $y \in X$ eine Variable, die weder in Γ noch in t enthalten ist, so gilt auch $\Gamma, y : \sigma \vdash t : \tau$ für alle $\sigma \in \mathcal{F}$.

b) Gilt $\Gamma \vdash t : \tau$, so gibt es für jede Variable $x \in \mathrm{FV}(t)$ einen Typ $\tau_x \in \mathcal{F}$ mit $(x : \tau_x) \in \Gamma$.

c) Gilt $\Gamma \vdash t : \tau$ mit $(y : \sigma) \in \Gamma$ und $y \notin \mathrm{FV}(t)$, so gilt auch $\Gamma \setminus \{(y : \sigma)\} \vdash t : \tau$.

Beweis:

a) Induktion über den Termaufbau von t:

- $t = x \in X$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\tau_1 \in \mathcal{F}$ mit $(x : \tau_1) \in \Gamma \subseteq \Gamma, y : \sigma$ und $\tau_1 \approx \tau$. Mit (Var) und (Kon) folgt $\Gamma, y : \sigma \vdash x : \tau$.
- $t = t_1 t_2$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\tau_1 \in \mathcal{F}$ mit $\Gamma \vdash t_1 : \tau_1 \to \tau$ und $\Gamma \vdash t_2 : \tau_1$. Da y weder in t_1 noch in t_2 enthalten ist, folgt nach Induktionsannahme $\Gamma, y : \sigma \vdash t_1 : \tau_1 \to \tau$ und $\Gamma, y : \sigma \vdash t_2 : \tau_1$. Anwenden von (App) liefert $\Gamma, y : \sigma \vdash t_1 t_2 : \tau$.
- $t = \lambda x.t_1$:
  Nach dem Erzeugungslemma 2.11 gibt es $\tau_1, \tau_2 \in \mathcal{F}$ mit $\tau \approx \tau_2 \to \tau_1$ und $\Gamma, x : \tau_2 \vdash t_1 : \tau_1$. Nach Induktionsannahme folgt $\Gamma, x : \tau_2, y : \sigma \vdash t_1 : \tau_1$, Anwenden von (Abs) und (Kon) liefert $\Gamma, y : \sigma \vdash \lambda x.t_1 : \tau$.

b) Induktion über den Termaufbau von t:

- $t = x \in X$:
  Folgt direkt aus dem Erzeugungslemma 2.11.
- $t = t_1 t_2$:
  Alle freien Variablen von $t_1 t_2$ liegen in t_1 oder in t_2. Mit dem Erzeugungslemma 2.11 folgt die Behauptung aus der Induktionsannahme.
- $t = \lambda x.t_1$:
  Jede freie Variable von t ist bereits frei in t_1. Daher folgt die Behauptung erneut mit dem Erzeugungslemma 2.11 aus der Induktionsannahme.

c) Induktion über den Termaufbau von t:

- $t = x \in X$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\tau_1 \in \mathcal{F}$ mit $(x : \tau_1) \in \Gamma$ und $\tau_1 \approx \tau$. Wegen $x \neq y$ ist auch $(x : \tau_1) \in \Gamma \setminus \{(y : \sigma)\}$, Anwenden von (Var) und (Kon) liefert $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; x : \tau$.
- $t = t_1 t_2$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\tau_1 \in \mathcal{F}$ mit $\Gamma \;\vdash\; t_1 : \tau_1 \to \tau$ und $\Gamma \;\vdash\; t_2 : \tau_1$. Nach Induktionsannahme gilt wegen $y \notin \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$ auch $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; t_1 : \tau_1 \to \tau$ und $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; t_2 : \tau_1$. Anwenden von (App) liefert $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; t_1 t_2 : \tau$.
- $t = \lambda y.t_1$:
  Nach dem Erzeugungslemma 2.11 gibt es $\tau_1, \tau_2 \in \mathcal{F}$ mit $\tau \approx \tau_2 \to \tau_1$ und $\Gamma,\, y : \tau_2 \;\vdash\; t_1 : \tau_1$. Da jede Variable höchstens einmal in Γ vorkommt, ist mit $(y : \sigma) \in \Gamma$ also $\sigma = \tau_2$. Damit gilt $\Gamma, y : \tau_2 = \Gamma \setminus \{(y : \sigma)\} \cup \{(y : \tau_2)\}$, und es folgt $\Gamma \setminus \{(y : \sigma)\} \cup \{(y : \tau_2)\} \;\vdash\; t_1 : \tau_1$. Anwenden von (Abs) und (Kon) liefert $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; \lambda y.t_1 : \tau$.
- $t = \lambda x.t_1$ mit $x \neq y$:
  Nach dem Erzeugungslemma 2.11 gibt es $\tau_1, \tau_2 \in \mathcal{F}$ mit $\tau \approx \tau_2 \to \tau_1$ und $\Gamma,\, x : \tau_2 \;\vdash\; t_1 : \tau_1$. Nach Induktionsannahme gilt $\Gamma \cup \{(x : \tau_2)\} \setminus \{(y : \sigma)\} \;\vdash\; t_1 : \tau_1$, Anwenden von (Abs) und (Kon) liefert $\Gamma \setminus \{(y : \sigma)\} \;\vdash\; \lambda x.t_1 : \tau$.

$\square$

Aus dem Basislemma folgt sofort:

Korollar 2.13 *Zu jedem Urteil* $\Gamma \;\vdash\; t : \tau$ *ist auch* $\Gamma' \;\vdash\; t : \tau$ *mit* $\Gamma' := \{(x : \tau_x) \mid x \in \mathrm{FV}(t)\}$ *ein Urteil.*

Auch im getypten λ-Kalkül benötigt man den Begriff der Substitution. Daß diese keine Probleme bereitet, ist die Aussage des folgenden Lemmas:

Lemma 2.14 (Substitutionslemma) *Sei* Γ *eine Basis,* $\sigma, \tau \in \mathcal{F}$*,* $x \in X$ *und* $s, t \in \Lambda$*. Dann folgt aus* $\Gamma, x : \sigma \;\vdash\; t : \tau$ *und* $\Gamma \;\vdash\; s : \sigma$ *auch* $\Gamma \;\vdash\; t[s/x] : \tau$*.*

Beweis: Induktion über die Länge $\mathrm{len}(t)$:

- Induktionsanfang $n = 1$:
  - $t = x \in X$:
    Aus $\Gamma, x : \sigma \;\vdash\; x : \tau$ folgt nach dem Erzeugungslemma 2.11 $\sigma \approx \tau$. Wegen $x[s/x] = s$ folgt mit $\Gamma \;\vdash\; s : \sigma$ und (Kon) auch $\Gamma \;\vdash\; x[s/x] : \tau$.

- $t = y \in X$ mit $y \neq x$:
  Aus $\Gamma, x : \sigma \;\vdash\; y : \tau$ folgt wegen $y[s/x] = y$ mit dem Basislemma 2.12 $\Gamma \vdash y[s/x] : \tau$.

- Induktionsschluß:

  - $t = t_1 t_2$:
    Nach dem Erzeugungslemma 2.11 existiert ein $\tau_1 \in \mathcal{F}$ mit $\Gamma, x : \sigma \vdash t_1 : \tau_1 \to \tau$ und $\Gamma, x : \sigma \vdash t_2 : \tau_1$. Nach Induktionsannahme gilt $\Gamma \vdash t_1[s/x] : \tau_1 \to \tau$ und $\Gamma \vdash t_2[s/x] : \tau_1$. Wegen $(t_1 t_2)[s/x] = (t_1[s/x])(t_2[s/x])$ folgt mit (App), daß $\Gamma \vdash (t_1 t_2)[s/x] : \tau_1$.

  - $t = \lambda x.t_1$:
    Da $\Gamma, x : \sigma \;\vdash\; \lambda x.t_1 : \tau$, gibt es wegen $(\lambda x.t_1)[s/x] = \lambda x.t_1$ auch eine Herleitung für $\Gamma, x : \sigma \vdash (\lambda x.t_1)[s/x] : \tau$. Nach dem Basislemma 2.12 ist damit auch $\Gamma \vdash (\lambda x.t_1)[s/x] : \tau$, da $x \notin \mathrm{FV}((\lambda x.t_1)[s/x])$.

  - $t = \lambda y.t_1$, $y \in X$, $y \neq x$:
    Nach dem Erzeugungslemma 2.11 gibt es $\tau_1, \tau_2 \in \mathcal{F}$ mit $\Gamma, x : \sigma, y : \tau_2 \vdash t_1 : \tau_1$ und $\tau \approx \tau_2 \to \tau_1$. Sei $z \in X$ eine Variable, die weder in Γ noch in s oder t enthalten ist mit $(\lambda y.t_1)[s/x] = \lambda z.(t_1[z/y][s/x])$. Dann gilt nach dem Basislemma 2.12 auch $\Gamma, x : \sigma, y : \tau_2, z : \tau_2 \;\vdash\; t_1 : \tau_1$. Wegen $\Gamma, x : \sigma, z : \tau_2 \;\vdash\; z : \tau_2$ folgt nach Induktionsannahme $\Gamma, x : \sigma, z : \tau_2 \;\vdash\; t_1[z/y] : \tau_1$. Da wegen des Basislemmas 2.12 mit $\Gamma \vdash s : \sigma$ auch $\Gamma, z : \tau_2 \;\vdash\; s : \sigma$ gilt, so folgt erneut aus der Induktionsannahme $\Gamma, z : \tau_2 \vdash t_1[z/y][s/x]$. Dieser Schluß ist möglich, da $\mathrm{len}(t_1[z/y]) = \mathrm{len}(t_1)$. Anwenden von (Abs) und (Kon) liefert wegen $\lambda z.(t[z/y][s/x]) = (\lambda y.t_1)[s/x]$ gerade $\Gamma \vdash (\lambda y.t_1)[s/x] : \tau$.

□

Folgendes Beispiel zeigt, daß aus $\Gamma \vdash s : \tau$ und $s \underset{\alpha}{=} t$ nicht $\Gamma \vdash t : \tau$ folgt:

Beispiel:

$$\dfrac{\dfrac{}{x : \sigma,\ y : \tau \quad \vdash \quad x : \sigma}\,(\mathrm{Var})}{y : \tau \quad \vdash \quad \lambda x.x \ : \ \sigma \to \sigma}\,(\mathrm{Abs})$$

ist eine korrekte Herleitung für $y : \tau \;\vdash\; \lambda x.x : \sigma \to \sigma$. Für $\sigma \neq \tau$ ist jedoch $y : \tau \not\vdash \lambda y.y : \sigma \to \sigma$, obwohl $\lambda x.x \underset{\alpha}{=} \lambda y.y$.

In etwas abgeschwächter Form gilt:

Lemma 2.15 *Sei Γ eine Basis, $\tau \in \mathcal{F}$ und $s, t \in \Lambda$ mit $s \underset{\alpha}{=} t$, wobei keine Variable sowohl in s als auch in t gebunden ist. Enthält Γ keine der in s gebundenen Variablen und gilt $\Gamma \vdash t : \tau$, dann auch $\Gamma \vdash s : \tau$.*

Beweis: Induktion über den Termaufbau von t:

- $t = x \in X$:
  Dieser Fall ist trivial, da t keine gebundenen Variablen enthält.

- $t = t_1 t_2$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\tau_1 \in \mathcal{F}$ mit $\Gamma \vdash t_1 : \tau_1 \to \tau$ und $\Gamma \vdash t_2 : \tau_1$. Wegen $t_1 t_2 \underset{\alpha}{=} s$ gibt es $s_1, s_2 \in \Lambda$ mit $s = s_1 s_2$, $t_1 \underset{\alpha}{=} s_1$ und $t_2 \underset{\alpha}{=} s_2$, wobei Γ nach Voraussetzung keine der in s_1 oder s_2 gebundenen Variablen enthält. Nach Induktionsannahme gilt also $\Gamma \vdash s_1 : \tau_1 \to \tau$ und $\Gamma \vdash s_2 : \tau_1$, mit (App) folgt $\Gamma \vdash s : \tau$.

- $t = \lambda x.t_1$:
  Nach dem Erzeugungslemma 2.11 gibt es $\tau_1, \tau_2 \in \mathcal{F}$ mit $\tau \approx \tau_2 \to \tau_1$ und $\Gamma, x : \tau_2 \vdash t_1 : \tau_1$. Wegen $s \underset{\alpha}{=} t$ gibt es ein $s_1 \in \Lambda$ mit $s = \lambda y.s_1$ und $s_1 \underset{\alpha}{=} t_1[y/x]$, wobei y weder in Γ noch in t_1 vorkommt (für $y = x$ folgt die Behauptung direkt aus der Induktionsannahme). Ersetzt man in der Herleitung von $\Gamma, x : \tau_2 \vdash t_1 : \tau_1$ jede vorkommende Basis Γ' durch $\Gamma' \cup \{(y : \tau_2)\}$, so erhält man $\Gamma, x : \tau_2, y : \tau_2 \vdash t_1 : \tau_1$. Nach dem Substitutionslemma 2.14 ist $\Gamma, y : \tau_2 \vdash t_1[y/x] : \tau_1$ und nach Induktionsannahme auch $\Gamma, y : \tau_2 \vdash s_1[y/x] : \tau_1$. Anwenden von (Abs) und (Kon) liefert $\Gamma \vdash s : \tau$.
  $\square$

Nicht alle Kongruenzen auf freien Gruppoiden eignen sich als Typsysteme. Im allgemeinen gilt nämlich nicht, daß aus $\Gamma \vdash (\lambda x.s)t : \tau$ auch schon $\Gamma \vdash s[t/x] : \tau$ folgt. Dies soll jetzt genauer untersucht werden.

Definition 2.16 *Der bezüglich $(\underline{\mathcal{F}}, \approx)$ getypte $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül heißt **$\beta$-typtreu**, falls für eine Basis Γ und für alle $\tau \in \mathcal{F}$ und $s, t \in \Lambda$ mit $s \xrightarrow[\beta]{} t$ gilt:*

$$\Gamma \vdash s : \tau \quad \Rightarrow \quad \Gamma \vdash t : \tau$$

Satz 2.17

a) Erfüllt $\underline{\mathcal{F}}$ die Kürzungsregel

(K) *Für alle $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{F}$ gilt:*
$$\sigma_1 \to \tau_1 \approx \sigma_2 \to \tau_2 \quad \Rightarrow \quad \sigma_1 \approx \sigma_2 \text{ und } \tau_1 \approx \tau_2,$$

so ist der $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül β-typtreu.

b) Ist der $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül β-typtreu, so erfüllt $(\underline{\mathcal{F}},\approx)$ die Regel:

(L) *Für alle $\sigma_1,\sigma_2,\tau_1,\tau_2 \in \mathcal{F}$ gilt:*
$\sigma_1 \to \tau_1 \approx \sigma_2 \to \tau_2 \quad \Rightarrow \quad \tau_1 \approx \tau_2.$

Beweis:

a) Wir zeigen: $\Gamma \vdash (\lambda x.t)s : \tau \;\Rightarrow\; \Gamma \vdash t[s/x] : \tau$. Daraus folgt die Behauptung. Nach dem Erzeugungslemma 2.11 folgt aus $\Gamma \vdash (\lambda x.t)s : \tau$, daß es ein $\sigma \in \mathcal{F}$ gibt mit $\Gamma \vdash \lambda x.t : \sigma \to \tau$ und $\Gamma \vdash s : \sigma$. Nochmalige Anwendung des Erzeugungslemmas 2.11 liefert $\sigma_1, \tau_1 \in \mathcal{F}$ mit $\sigma \to \tau \approx \sigma_1 \to \tau_1$ und $\Gamma, x : \sigma_1 \vdash t : \tau_1$. Nach Voraussetzung ist $\sigma \approx \sigma_1$ und $\tau \approx \tau_1$. Mit (Kon) gilt $\Gamma \vdash s : \sigma_1$ und $\Gamma, x : \sigma_1 \vdash t : \tau$ und mit dem Substitutionslemma 2.14 folgt $\Gamma \vdash t[s/x] : \tau$.

b) Sei $\sigma_1 \to \tau_1 \approx \sigma_2 \to \tau_2$ für $\sigma_1,\sigma_2,\tau_1,\tau_2 \in \mathcal{F}$. Dann gibt es mit $x,y,z \in X$ eine Herleitung:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\sigma_1,\ y:\tau_1,\ z:\sigma_2 \ \vdash\ y:\tau_1}\,(\mathrm{Var})}{y:\tau_1,\ z:\sigma_2 \ \vdash\ \lambda x.y:\sigma_1\to\tau_1}\,(\mathrm{Abs})}{y:\tau_1,\ z:\sigma_2 \ \vdash\ \lambda x.y:\sigma_2\to\tau_2}\,(\mathrm{Kon}) \qquad \dfrac{}{y:\tau_1,\ z:\sigma_2 \ \vdash\ z:\sigma_2}\,(\mathrm{Var})}{y:\tau_1,\ z:\sigma_2 \ \vdash\ (\lambda x.y)z:\tau_2}\,(\mathrm{App})$$

Da der $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül als β-typtreu angenommen wurde, gilt wegen $y = y[z/x]$ auch $y : \tau_1, z : \sigma_2 \vdash y : \tau_2$. Folglich ist $\tau_1 \approx \tau_2$.

□

Wir haben gezeigt, daß (K) ein hinreichendes und (L) ein notwendiges Kriterium für β-Typtreue sind. Daß (L) nicht hinreichend ist, zeigt folgendes Beispiel:

Beispiel: Sei $\mathcal{J} = \{\iota_n \mid n \in \mathbb{N}\}$ eine Menge von Grundtypen. Durch

$$\iota_m \to \iota_n \sim \begin{cases} \iota_n & \text{falls } n = 0 \\ \iota_{n+1} & \text{sonst.} \end{cases}$$

sei auf dem frei erzeugten Gruppoid $\underline{\mathcal{F}_\mathcal{J}}$ eine Relation definiert. Für $\sigma_1,\sigma_2,\tau_1,\tau_2 \in \mathcal{F}_\mathcal{J}$ folgt dann aus $\sigma_1 \to \tau_1 \approx \sigma_2 \to \tau_2$ auch $\tau_1 \approx \tau_2$. Folgendes ist eine korrekte Herleitung:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$x:\iota_0,\, y:\iota_1 \;\vdash\; x:\iota_0$}
\RightLabel{(Kon)}
\UnaryInfC{$x:\iota_0,\, y:\iota_1 \;\vdash\; x:\iota_0 \to \iota_0$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$x:\iota_0,\, y:\iota_1 \;\vdash\; x:\iota_0$}
\RightLabel{(App)}
\BinaryInfC{$x:\iota_0,\, y:\iota_1 \;\vdash\; xx:\iota_0$}
\RightLabel{(Abs)}
\UnaryInfC{$y:\iota_1 \;\vdash\; \lambda x.xx:\iota_0 \to \iota_0$}
\RightLabel{(Kon)}
\UnaryInfC{$y:\iota_1 \;\vdash\; \lambda x.xx:\iota_1 \to \iota_0$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$y:\iota_1 \;\vdash\; y:\iota_1$}
\RightLabel{(App)}
\BinaryInfC{$y:\iota_1 \;\vdash\; (\lambda x.xx)y:\iota_0$}
\end{prooftree}
$$

Wäre auch $y : \iota_1 \vdash yy : \iota_0$, so gäbe es nach dem Erzeugungslemma 2.11 ein $\tau \in \mathcal{F}_{\mathcal{J}}$ mit $y : \iota_1 \vdash y : \tau \to \iota_0$ und $y : \iota_1 \vdash y : \tau$. Damit wäre $\iota_1 \approx \tau \to \iota_0$ und $\iota_1 \approx \tau$, also auch $\iota_1 \approx \iota_1 \to \iota_0$. In $\underline{\mathcal{F}_{\mathcal{J}}}$ ist jedoch $\iota_1 \not\approx \iota_1 \to \iota_0$, der $\lambda_{(\underline{\mathcal{F}_{\mathcal{J}}},\approx)}$-Kalkül ist also nicht β-typtreu.

Keine Probleme gibt es für die η-Reduktion:

Definition 2.18 *Der bezüglich $(\underline{\mathcal{F}},\approx)$-getypte $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül heißt* ***$\eta$-typtreu****, falls für eine Basis Γ und für alle $\tau \in \mathcal{F}$ und $s, t \in \Lambda$ mit $s \xrightarrow[\eta]{} t$ gilt:*

$$\Gamma \vdash s : \tau \quad \Rightarrow \quad \Gamma \vdash t : \tau.$$

Satz 2.19 *Für jedes Typsystem $(\underline{\mathcal{F}},\approx)$ ist der $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül η-typtreu.*

Beweis: Gelte $\Gamma \vdash \lambda x.tx : \tau$ für ein $t \in \Lambda$ mit $x \notin \mathrm{FV}(t)$. Dann gibt es nach dem Erzeugungslemma 2.11 $\tau_1, \tau_2 \in \mathcal{F}$ mit $\tau \approx \tau_2 \to \tau_1$ und $\Gamma, x : \tau_2 \vdash tx : \tau_1$. Erneut nach dem Erzeugungslemma 2.11 gibt es ein $\sigma \in \mathcal{F}$ mit $\Gamma, x : \tau_2 \vdash t : \sigma \to \tau_1$ und $\Gamma, x : \tau_2 \vdash x : \sigma$, also $\sigma \approx \tau_2$. Nach Voraussetzung ist $x \notin \mathrm{FV}(t)$, es folgt mit dem Basislemma 2.12 $\Gamma \vdash t : \sigma \to \tau_1$ und wegen $\tau \approx \tau_2 \to \tau_1 \approx \sigma \to \tau_1$ mit (Kon) auch $\Gamma \vdash t : \tau$. $\square$

Da also jeder $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül η-typtreu ist, werden wir statt *β-typtreu* einfach nur *typtreu* schreiben. Typsysteme, die die Regel (K) erfüllen, nennen wir **K-Typsysteme**. Viele Eigenschaften des ungetypten λ-Kalkül übertragen sich auf den $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül. Insbesondere gilt:

Satz 2.20 (Church-Rosser) *Sei $(\underline{\mathcal{F}},\approx)$ ein K-Typsystem. Dann ist die β- bzw. $\beta\eta$- Reduktion im $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül konfluent, es gilt also:*

$$\begin{aligned} \Gamma \vdash s : \tau,\; s \xrightarrow[\beta\eta]{}\!\!\!\!\to t_1,\; s \xrightarrow[\beta\eta]{}\!\!\!\!\to t_2 \;\Rightarrow\; & \textit{es existiert ein } u \in \Lambda \textit{ mit } \Gamma \vdash u : t \\ & \textit{und } t_1 \xrightarrow[\beta\eta]{}\!\!\!\!\to u,\; t_2 \xrightarrow[\beta\eta]{}\!\!\!\!\to u. \end{aligned}$$

(Entsprechend für $\xrightarrow[\beta]{}\!\!\!\!\to$).

$$
\begin{tikzcd}
 & s : \tau \arrow[dl, two heads, "\beta\eta"'] \arrow[dr, two heads, "\beta\eta"] & \\
t_1 : \tau \arrow[dr, dotted, two heads, "\beta\eta"'] & & t_2 : \tau \arrow[dl, dotted, two heads, "\beta\eta"] \\
 & u : \tau &
\end{tikzcd}
$$

Beweis: Aus $\Gamma \vdash s : \tau$, $s \underset{\beta\eta}{\twoheadrightarrow} t_1$, $s \underset{\beta\eta}{\twoheadrightarrow} t_2$ folgt mit Satz 2.17 und Satz 2.19 bereits $\Gamma \vdash t_1 : \tau$ und $\Gamma \vdash t_2 : \tau$. Nach Satz 1.9 gibt es ein $u \in \Lambda$ mit $t_1 \underset{\beta\eta}{\twoheadrightarrow} u$ und $t_2 \underset{\beta\eta}{\twoheadrightarrow} u$, erneut aus Satz 2.17 und Satz 2.19 folgt $\Gamma \vdash u : \tau$. Für die β-Reduktion verläuft der Beweis analog. $\square$

Dieser Beweis funktioniert nicht für den Church-Style. Natürlich gilt auch hier der Satz von Church-Rosser, obige Argumentation scheitert jedoch, siehe [MS95].

Wir werden nun untersuchen, welche Typsysteme K-Typsysteme sind. Man sieht leicht ein, daß es keine nichttrivialen endlichen K-Typsysteme gibt:

Lemma 2.21 *Die Anzahl der Kongruenzklassen eines Typsystems* $(\underline{\mathcal{F}}, \approx)$ *ist 0, 1 oder unendlich.*

Beweis: Sei $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ mit $n > 1$ die Trägermenge eines Gruppoids $\underline{\mathcal{T}} = (\mathcal{T}, \rightarrow)$. Da es $n^2 > n$ Möglichkeiten gibt, zwei Elemente aus $\mathcal{T}$ mit $\rightarrow$ zu verknüpfen, gibt es $i_0, i_1, i_2, j_1, j_2 \in \{1, \ldots, n\}$, so daß $\tau_{i_0} = \tau_{i_1} \rightarrow \tau_{i_2} = \tau_{j_1} \rightarrow \tau_{j_2}$ mit $i_1 \neq j_1$ oder $i_2 \neq j_2$.

$\rightarrow$	τ_1	$\cdots$	τ_{i_2}	$\cdots$	τ_{j_2}	$\cdots$	τ_n
τ_1			$\vdots$		$\vdots$		
$\vdots$			$\vdots$		$\vdots$		
τ_{i_1}	$\cdots$	$\cdots$	τ_{i_0}	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\vdots$			$\vdots$		$\vdots$		
τ_{j_1}	$\cdots$	$\cdots$	$\cdots$	$\cdots$	τ_{i_0}	$\cdots$	$\cdots$
$\vdots$			$\vdots$		$\vdots$		
τ_n			$\vdots$		$\vdots$		

Betrachtet man das freie Gruppoid $\underline{\mathcal{F}}_{\mathcal{T}}$ mit der entsprechenden Kongruenz $\approx$, so erfüllt dieses die Kürzungsregel (K) nicht, $(\underline{\mathcal{F}}_{\mathcal{T}}, \approx)$ ist daher kein K-Typsystem. Daraus folgt die Behauptung. $\square$

Da das leere Gruppoid keine getypten λ-Terme erzeugt und der mit dem trivialen Typsystem $(\underline{\mathcal{F}}, \nabla)$ getypte λ-Kalkül gerade dem ungetypten λ-Kalkül entspricht, folgt aus Lemma 2.21, daß jedes nichttriviale K-Typsystem bereits unendlich viele Klassen von Typen besitzt. Wählt man das freie Gruppoid $\underline{\mathcal{F}}_{\mathcal{J}}$ mit der trivialen Kongruenz Δ als Typsystem für den λ-Kalkül, so erhält man den **einfach getypten λ-Kalkül** $\lambda_{\rightarrow}$ mit $\mathcal{J}$ als der Menge der Grundtypen. Da in diesem Fall die Kürzungsregel (K) trivialerweise erfüllt ist, ist $\lambda_{\rightarrow}$ nach Satz 2.17 typtreu.

Für eine Charakterisierung der K-Typsysteme ist folgendes Lemma aus der Allgemeinen Algebra hilfreich:

Lemma 2.22 *Sei $\underline{\mathcal{F}}_{\mathcal{J}}$ ein freies Gruppoid mit einer Relation $\sim$. Genau dann gilt $\sigma \approx \tau$, wenn man τ aus σ durch sukzessives Ersetzen einzelner Vorkommen von Teiltypen δ durch δ' mit $\delta \sim \delta'$ oder $\delta' \sim \delta$ erhält.*

Beweis: Sei τ' derjenige Typ, der durch Ersetzen eines Vorkommens eines Teiltyps δ durch δ' mit $\delta \sim \delta'$ oder $\delta' \sim \delta$ aus τ entsteht. Wir zeigen $\tau \approx \tau'$, daraus folgt mit der Transitivität (Regel (t)) die Behauptung.

- $\tau \in \mathcal{J}$:
  Da δ ein Teiltyp von τ ist, ist $\tau = \delta$. Dann gilt wegen $\delta \sim \delta'$ oder $\delta' \sim \delta$ mit Regel (c) und gegebenenfalls Regel (s) $\tau \approx \delta'$.

- $\tau = \tau_1 \rightarrow \tau_2$:
  Im Fall $\delta = \tau$ folgt die Behauptung wie oben. Andernfalls sei das Vorkommen von δ in τ, welches durch δ' ersetzt wird, o.B.d.A. ein Teiltyp von τ_1. Nach Induktionsannahme erhält man aus τ_1 dadurch ein σ_1 mit $\tau_1 \approx \sigma_1$. Mit den Regeln (r) und (v) gibt es eine Herleitung:

$$\dfrac{\begin{array}{c}\vdots\\ \tau_1 \approx \sigma_1\end{array} \qquad \dfrac{}{\tau_2 \approx \tau_2}\,(\mathrm{r})}{\tau_1 \rightarrow \tau_2 \;\approx\; \sigma_1 \rightarrow \tau_2}\,(\mathrm{v})$$

Sei umgekehrt $\sigma \approx \tau$. Wir beweisen die Aussage mit Induktion über den Aufbau der Herleitung von $\sigma \approx \tau$.

- (c) und (r): Trivial.
- (s): Gelte die Behauptung für $\tau \approx \sigma$. Nach Induktionsannahme erhält man σ durch eine Folge von Ersetzungen von Vorkommen von δ in τ durch δ' mit $\delta \sim \delta'$ oder $\delta' \sim \delta$. Führt man diese Ersetzungen in umgekehrter Reihenfolge mit σ beginnend durch, so erhält man τ.
- (t): Gelte die Behauptung für $\sigma \approx \rho$ und $\rho \approx \tau$. Nach Induktionsannahme erhält man ρ durch eine Folge von Ersetzungen aus σ und τ durch eine andere Ersetzungsfolge aus ρ. Führt man diese Ersetzungen nacheinander durch, so erhält τ aus σ.
- (v): Sei $\tau = \tau_1 \to \tau_2$, $\sigma = \sigma_1 \to \sigma_2$ und gelte die Behauptung für $\sigma_1 \approx \tau_1$ und $\sigma_2 \approx \tau_2$. Nach Induktionsannahme erhält man τ_1 durch eine Folge von Ersetzungen aus σ_1 und τ_2 aus σ_2. Ähnlich wie bei (t) erhält man $\tau_1 \to \tau_2$ aus $\sigma_1 \to \sigma_2$, indem man diese Ersetzungen nacheinander durchführt.

$\square$

Folgendes Beispiel veranschaulicht dieses Verfahren:

Beispiel: Sei $\mathcal{J} = \{\sigma, \tau, \rho\}$ mit $\sigma \sim \rho \to \sigma \to \tau$, $\tau \sim \sigma$ und $\rho \sim \tau \to \sigma$. Dann gibt es eine Herleitung für $(\sigma \to \rho \to \sigma \to \tau) \to \tau \sim \rho \to \sigma$:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(c)}
\UnaryInfC{$\rho \approx \tau \to \sigma$}
\AxiomC{}
\RightLabel{(c)}
\UnaryInfC{$\tau \approx \sigma$}
\AxiomC{}
\RightLabel{(c)}
\UnaryInfC{$\sigma \approx \rho \to \sigma \to \tau$}
\RightLabel{(v)}
\BinaryInfC{$\tau \to \sigma \approx \sigma \to \rho \to \sigma \to \tau$}
\RightLabel{(t)}
\BinaryInfC{$\rho \approx \sigma \to \rho \to \sigma \to \tau$}
\AxiomC{}
\RightLabel{(c)}
\UnaryInfC{$\tau \approx \sigma$}
\RightLabel{(s)}
\UnaryInfC{$\sigma \approx \tau$}
\RightLabel{(v)}
\BinaryInfC{$\rho \to \sigma \approx (\sigma \to \rho \to \sigma \to \tau) \to \tau$}
\RightLabel{(s)}
\UnaryInfC{$(\sigma \to \rho \to \sigma \to \tau) \to \tau \approx \rho \to \sigma$}
\end{prooftree}
$$

Über folgende Zwischenschritte erhält man $\rho \to \sigma$ aus $(\sigma \to \rho \to \sigma \to \tau) \to \tau$, die zu ersetzenden Teiltypen sind unterstrichen:

$$
\begin{aligned}
(\sigma \to \rho \to \sigma \to \tau) \to \underline{\tau} &\leadsto (\sigma \to \underline{\rho \to \sigma \to \tau)} \to \sigma \\
&\leadsto (\underline{\sigma} \to \sigma) \to \sigma \\
&\leadsto (\underline{\tau \to \sigma)} \to \sigma \\
&\leadsto \rho \to \sigma
\end{aligned}
$$

Satz 2.23 *Sei* $(\underline{\mathcal{F}}_\mathcal{J}, \approx)$ *ein von einer symmetrischen Relation* $\sim$ *erzeugtes Typsystem. Genau dann ist* $(\underline{\mathcal{F}}_\mathcal{J}, \approx)$ *ein K-Typsystem, wenn für alle* $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{F}_\mathcal{J}$ *und* $\iota_1, \ldots, \iota_n \in \mathcal{J}$ *mit* $\sigma_1 \to \tau_1 \sim \iota_1 \sim \ldots \sim \iota_n \sim \sigma_2 \to \tau_2$ *gilt:* $\sigma_1 \approx \sigma_2$ *und* $\tau_1 \approx \tau_2$.

Beweis: Für jedes K-Typsystem folgt aus $\sigma_1 \to \tau_1 \sim \iota_1 \sim \ldots \sim \iota_n \sim \sigma_2 \to \tau_2$ mit $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{F}_\mathcal{J}$ und $\iota_1, \ldots, \iota_n \in \mathcal{J}$ auch $\sigma_1 \approx \sigma_2$ und $\tau_1 \approx \tau_2$. Zum Beweis der Umkehrung seien wieder $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{F}_\mathcal{J}$ mit $\sigma_1 \to \tau_1 \approx \sigma_2 \to \tau_2$, wir zeigen $\sigma_1 \approx \sigma_2$ und $\tau_1 \approx \tau_2$. Nach Lemma 2.22 erhält man $\sigma_2 \to \tau_2$ durch sukzessives Ersetzen von Vorkommen von Teiltypen δ in $\sigma_1 \to \tau_1$ durch δ' mit $\delta \sim \delta'$. Ein Teiltyp δ von $\sigma_1 \to \tau_1$ ist entweder bereits ein Teiltyp von σ_1 oder τ_1 oder eben $\sigma_1 \to \tau_1$ selbst. Sei zunächst δ ein Teiltyp von σ_1. Durch Ersetzen erhält man ein $\sigma_1' \in \mathcal{F}_\mathcal{J}$ mit $\sigma_1 \approx \sigma_1'$. Ist δ ein Teiltyp von τ_1, so erhält man analog ein $\tau_1' \in \mathcal{F}_\mathcal{J}$ mit $\tau_1 \approx \tau_1'$. Erhält man $\sigma_2 \to \tau_2$ also nur durch Ersetzungen echter Teiltypen, so gilt $\sigma_1 \approx \sigma_2$ und $\tau_1 \approx \tau_2$. Da außerdem für jede Teilfolge $\sigma_1' \to \tau_1' \rightsquigarrow \iota_1 \rightsquigarrow \ldots \rightsquigarrow \iota_n \rightsquigarrow \sigma_2' \to \tau_2'$ der Ersetzungsfolge mit $\iota_1, \ldots, \iota_n \in \mathcal{J}$ nach Voraussetzung $\sigma_1' \approx \sigma_2'$ und $\tau_1' \approx \tau_2'$ gilt (wobei $\iota_1 \rightsquigarrow \ldots \rightsquigarrow \iota_n$ auch leer sein darf), folgt $\sigma_1 \approx \sigma_2$ und $\tau_1 \approx \tau_2$. $\square$

Eine besondere Rolle unter den Gruppoiden spielen die in gewisser Hinsicht endlich erzeugten. Diese werden in der folgenden Definition präzisiert:

Definition 2.24 *Ein Typsystem* $(\underline{\mathcal{F}}_\mathcal{J}, \approx)$ *heißt* ***endlich repräsentierbar****, falls* $\mathcal{J}$ *endlich ist und es eine endliche Relation* $\sim$ *gibt, so daß* $\approx$ *die von* $\sim$ *erzeugte Kongruenz ist.*

Für ein beliebiges endlich repräsentierbares Typsystem ist es entscheidbar, ob die Kürzungsregel (K) erfüllt ist. Dafür genügt es nach Satz 2.23, die (endliche) Erzeugendenrelation zu untersuchen. Das hierbei entstehende Problem, für zwei gegebene Typen σ und τ zu entscheiden, ob sie zueinander kongruent sind, wird in der Allgemeinen Algebra *Wortproblem* genannt. Folgender Satz besagt, daß dieses für endlich repräsentierbare Typsysteme entscheidbar ist.

Satz 2.25 *Sei* $\underline{\mathcal{F}}$ *ein frei erzeugtes Gruppoid mit einer endlichen binären Relation* $\sim$. *Dann gibt es ein Entscheidungsverfahren dafür, ob* $\sigma \approx \tau$ *für* $\sigma, \tau \in \mathcal{F}$ *gilt.*

Beweis: Sei $\mathcal{S} := \{\delta \in \mathcal{F} \mid \exists \delta' \in \mathcal{F} : \delta \sim \delta' \text{ oder } \delta' \sim \delta\} \cup \{\sigma, \tau\}$ und $\mathcal{T}$ die Menge aller Teiltypen von Elementen aus $\mathcal{S}$. Nach Voraussetzung ist $\mathcal{T}$ endlich. Sei $\approx'$ die kleinste reflexive, symmetrische und transitive Relation auf $\mathcal{T}$, die $\sim$ enthält und für die für alle $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathcal{T}$ gilt:

$$\sigma_1 \approx' \tau_1, \quad \sigma_2 \approx' \tau_2, \quad \sigma_1 \to \sigma_2,\ \tau_1 \to \tau_2 \in \mathcal{T} \qquad \Rightarrow \qquad \sigma_1 \to \sigma_2 \approx' \tau_1 \to \tau_2.$$

Mit $\mathcal{T}$ ist auch $\approx'$ endlich. Gelte $\sigma \approx \tau$. Nach Lemma 2.22 gibt es $\sigma_0, \sigma_1, \ldots, \sigma_{n+1} \in \mathcal{F}$ mit $\sigma = \sigma_0 \approx \sigma_1 \approx \ldots \approx \sigma_{n+1} = \tau$, wobei σ_{i+1} aus σ_i für alle $i = 0, \ldots, n$ durch Ersetzen eines Teiltyps δ durch δ' in σ_i mit $\delta \sim \delta'$ oder $\delta' \sim \delta$ entsteht, wir schreiben $\sigma_i \xleftrightarrow{1} \sigma_{i+1}$. Mit Induktion über $\sum_{i=0}^{n+1} \mathrm{len}(\sigma_i)$ zeigen wir, daß aus $\sigma_0 \xleftrightarrow{1} \sigma_1 \xleftrightarrow{1} \ldots \xleftrightarrow{1} \sigma_{n+1}$ mit $\sigma_0, \sigma_{n+1} \in \mathcal{T}$ auch $\sigma_0 \approx' \sigma_{n+1}$ folgt. Falls es ein $i = 0, \ldots, n$ mit $\sigma_i \sim \sigma_{i+1}$ oder $\sigma_{i+1} \sim \sigma_i$ gibt, so folgt nach Induktionsannahme $\sigma_0 \approx' \sigma_i$ und $\sigma_{i+1} \approx' \sigma_{n+1}$, also auch $\sigma_0 \approx' \sigma_{n+1}$. Andernfalls gibt es für jedes σ_i mit $i = 0, \ldots, n+1$ zwei Typen $\varphi_i, \psi_i \in \mathcal{F}$ mit $\sigma_i = \varphi_i \to \psi_i$. Es folgt $\varphi_0 \xleftrightarrow{1} \varphi_1 \xleftrightarrow{1} \ldots \xleftrightarrow{1} \varphi_{n+1}$ und $\psi_0 \xleftrightarrow{1} \psi_1 \xleftrightarrow{1} \ldots \xleftrightarrow{1} \psi_{n+1}$, nach Induktionsannahme gilt $\varphi_0 \approx' \varphi_{n+1}$ und $\psi_0 \approx' \psi_{n+1}$. Wegen der Verträglichkeit folgt $\sigma = \varphi_0 \to \psi_0 \approx' \varphi_{n+1} \to \psi_{n+1} = \tau$. Da trivialerweise mit $\sigma \approx' \tau$ auch $\sigma \approx \tau$ gilt, ist somit $\sigma \approx' \tau \;\Leftrightarrow\; \sigma \approx \tau$ gezeigt. Aus der Endlichkeit von $\approx'$ folgt die Behauptung.

$\square$

Folgendes Korollar beschreibt Typsysteme, die durch eine spezielle Relation erzeugt werden. Die Aussage folgt direkt aus Satz 2.23.

Korollar 2.26 *Sei $\underline{\mathcal{F}_\mathcal{J}}$ ein freies Gruppoid, zu jedem $\iota \in \mathcal{J}$ gebe es genau ein $\tau \in \mathcal{F}_\mathcal{J}$ mit $\iota \sim \tau$, wobei keine weiteren Elemente von $\mathcal{F}_\mathcal{J}$ in Relation stehen. Dann ist $(\underline{\mathcal{F}_\mathcal{J}}, \approx)$ ein K-Typsystem.*

Die in Korollar 2.26 betrachteten Typsysteme sind nach Satz 2.17 typtreu. Unter den K-Typsystemen spielen sie eine wichtige Rolle: Zu jedem endlich repräsentierbaren K-Typsystem $(\underline{\mathcal{F}_\mathcal{J}}, \approx)$ liefert folgendes Verfahren eine Relation $\sim'$ auf $\mathcal{J} \times \mathcal{F}_\mathcal{J}$, so daß $\mathcal{F}_\mathcal{J}$ faktorisiert nach der davon erzeugten Kongruenzrelation $\approx'$ isomorph zu $\mathcal{F}_\mathcal{J}/_\approx$ ist und es zu jedem $\iota_i \in \mathcal{J}$ genau ein $\tau_i \in \mathcal{F}$ gibt mit $\iota_i \sim' \tau_i$.

Sei $\mathcal{J} = \{\iota_1, \ldots, \iota_n\}$ eine Menge von Grundtypen und $\sim$ eine endliche binäre Relation auf $\mathcal{F}_\mathcal{J}$. Zu beachten ist, daß im Algorithmus $\sim$ nicht die starre Bezeichnung der Startrelation ist, sondern in jedem folgenden Schritt verändert werden kann.

Algorithmus 2.27

(1) *(a)* *Streiche jedes Paar $\sigma \sim \sigma$ mit $\sigma \in \mathcal{F}$.*

(b) *Ersetze jedes Paar der Form $\sigma \to \tau \sim \iota$ mit $\iota \in \mathcal{J}$, $\sigma, \tau \in \mathcal{F}$ durch $\iota \sim \sigma \to \tau$.*

(c) *Falls $\iota_{i_1} \sim \iota_{i_2} \sim \ldots \sim \iota_{i_n} \sim \iota_{i_1}$ mit $i_1 < i_2 < \ldots < i_n$, so streiche $\iota_{i_n} \sim \iota_{i_1}$.*

(d) *Ersetze im Fall $\iota_i \sim \iota_j$ mit $i, j = 1, \ldots, n$ jedes Auftreten von ι_i in anderen Paaren von $\sim$ durch ι_j.*

(e) *Streiche alle mehrfachen Vorkommen von Paaren* $\sigma \sim \tau$ *mit* $\sigma, \tau \in \mathcal{F}$.

(2) *Solange es noch Paare der Form* $\sigma_1 \to \sigma_2 \sim \tau_1 \to \tau_2$ *gibt, streiche* $\sigma_1 \to \sigma_2 \sim \tau_1 \to \tau_2$*, setze* $\sigma_1 \sim \tau_1$ *und* $\sigma_2 \sim \tau_2$ *und wiederhole Schritt (1).*

(3) *Sei* $F_i := |\{\tau \mid \iota_i \sim \tau\}|$. *Solange es noch ein* $i \in \{1, \ldots, n\}$ *gibt mit* $F_i > 1$:

   (a) *bestimme ein* $i \in \{1, \ldots, n\}$ *mit* $F_i > 1$ *und* $\max\{\mathrm{len}(\tau) \mid \iota_i \sim \tau\} \geq \max\{\mathrm{len}(\tau) \mid \iota_j \sim \tau,\ F_j > 1\}$;

   (b) *bestimme ein* $\varphi = \sigma_\varphi \to \tau_\varphi$ *mit* $\iota_i \sim \varphi$ *und* $\mathrm{len}(\varphi) \geq \mathrm{len}(\tau)$ *für alle* τ *mit* $\iota_i \sim \tau$;

   (c) *bestimme ein* $\psi = \sigma_\psi \to \tau_\psi$ *mit* $\iota_i \sim \psi$ *und* $\varphi \neq \psi$;

   (d) *streiche* $\iota_i \sim \varphi$ *und setze* $\varphi \sim \psi$;

   (e) *wiederhole Schritt (2).*

(4) *Für alle* $i \in \{1, \ldots, n\}$*, für die es keinen Typen* τ *mit* $\iota_j \sim \tau$ *gibt, setze* $\iota_i \sim \iota_i$. $\sim'$ *sei die zuletzt entstandene Relation* $\sim$.

Nach Voraussetzung von Schritt (3) existiert ein $i \in \{1, \ldots, n\}$, so daß $\mathrm{F}_i > 1$. Die vorkommenden Maxima existieren, da die Relation $\sim$ zu jedem Zeitpunkt des Ablaufs endlich ist. Hätte in Schritt (3b) φ nicht die Form $\varphi = \sigma_\varphi \to \tau_\varphi$, gäbe es wegen $\mathrm{F}_i > 1$ zwei Paare $\iota_i \sim \iota_j$ und $\iota_i \sim \iota_k$ ($j, k \in \{1, \ldots, n\}$, $j \neq k$). Dies kann aber nicht sein, da zuvor Schritt (1) durchgeführt worden ist. Aus einem ähnlichen Grund hat ψ die Form $\sigma_\psi \to \tau_\psi$, da es andernfalls Paare $\iota_i \sim \varphi$ und $\iota_i \sim \iota_j$ mit $j = 1, \ldots, n$ gäbe. Somit ist das Verfahren durchführbar.

Zum Nachweis der Korrektheit und der Terminierung von Algorithmus 2.27 brauchen wir noch einige Hilfsmittel:

Definition 2.28 *Sei* $(P, \leq)$ *eine geordnete Menge.* $(P, \leq)$ *erfüllt die* ***absteigende Kettenbedingung****, falls jede Kette* $p_0 \geq p_1 \geq p_2 \geq \ldots$ *mit* $p_n \in P$, $n \in \mathbb{N}$ *ein kleinstes Element besitzt. Auf dem Produkt* P^m *mit* $m \in M$ *ist die* ***lexikographische Ordnung*** $\sqsubseteq$ *wie folgt definiert:*

$$(p_1, \ldots, p_m) \sqsubseteq (q_1, \ldots, q_m) \quad :\Leftrightarrow \quad \begin{array}{l} p_1 = q_1, \ldots, p_m = q_m \qquad \textit{oder} \\ \exists i = 1, \ldots, m : p_1 = q_1, \ldots, p_{i-1} = q_{i-1},\ p_i < q_i \end{array}$$

Lemma 2.29 *Sei* $(P, \leq)$ *eine geordnete Menge, die die absteigende Kettenbedingung erfüllt. Für jede natürliche Zahl* $m \geq 1$ *erfüllt dann auch* $(P^m, \sqsubseteq)$ *die absteigende Kettenbedingung.*

Beweis: Wir beweisen die Behauptung mit Induktion über m. Der Fall $m = 1$ ist trivial. Angenommen P^m erfüllt die absteigende Kettenbedingung. Würde $P \times P^m$ eine unendlich absteigende Kette $(p_0, q_0) \sqsupset (p_1, q_1) \sqsupset (p_2, q_2) \ldots$ enthalten, so gäbe es nach Voraussetzung ein $i \in \mathbb{N}$ mit $p_j = p_i$ für alle $j \geq i$. Damit wäre $q_{i+1} > q_{i+2} > q_{i+3} > \ldots$ eine unendlich absteigende Kette in P^m, ein Widerspruch. Wegen $P^{m+1} \cong P \times P^m$ erfüllt also auch P^{m+1} die absteigende Kettenbedingung. □

Satz 2.30 *Algorithmus 2.27 terminiert und ist korrekt.*

Beweis: Wir zeigen zunächst die Terminierung. Da $\sim$ zu jedem Zeitpunkt aus endlich vielen Paaren von Typen endlicher Länge besteht, terminieren die Schritte (1) und (2). Sei $l := \max\{\text{len}(\tau) \mid \exists \sigma \in \mathcal{F}_{\mathcal{J}}\colon\ \sigma \sim \tau \text{ oder } \tau \sim \sigma\}$ die maximale Länge aller zu Beginn des Algorithmus in $\sim$ vorkommenden Typen. Für alle $k = 2, \ldots, l$ sei

$$m_k := |\{\tau \in \mathcal{F}_{\mathcal{J}} \mid \text{len}(\tau) \geq k \text{ und es ex. ein } i = 1, \ldots, n \text{ mit } \iota_i \sim \tau \text{ oder } \tau \sim \iota_i\}|.$$

Wir zeigen, daß $m := (m_l, m_{l-1}, \ldots, m_2) \in \mathbb{N}^{l-1}$ mit jedem Durchlaufen der Schritte (3a) bis (3e) bezüglich der lexikographischen Ordnung kleiner wird. Die Schritte (3a) bis (3c) terminieren immer, haben jedoch keinen Einfluß auf m. (3d) erhöht $m_{\text{len}(\psi)}$ um eins und läßt m ansonsten unverändert. In Schritt (3e) werden $m_{\text{len}(\varphi)}$ und $m_{\text{len}(\psi)}$ reduziert, damit wird insgesamt auch m kleiner. Nach Lemma 2.29 erfüllt $\mathbb{N}^{l-1}$ die absteigende Kettenbedingung, m kann also nicht unendlich oft vermindert werden. Da aber m bei jedem Durchlauf von (3a) bis (3e) verringert wird, terminiert Schritt (3). Es ist klar, daß auch Schritt (4) terminiert.

Nun kommen wir zum Beweis der Korrektheit. Wir zeigen zunächst, daß keiner der Schritte (1) bis (4) die von $\sim$ erzeugte Kongruenz $\approx$ verändert:

- Schritt (1): Wegen Regel (r) ist Teil (a) völlig unproblematisch. Das Ersetzen von Paaren der Gestalt $\sigma \to \tau \sim \iota$ durch $\iota \sim \sigma \to \tau$ in Schritt ändert wegen Regel (s) sicherlich auch nichts an der erzeugten Kongruenz. Wegen Regel (t) hat in Teil (c) das Streichen von $\iota_{i_n} \sim \iota_{i_1}$ keine Konsequenzen. Aus Regel (v) folgt mit Induktion über den Aufbau von τ, daß auch das Ersetzen von ι_i durch ι_j in τ mit $\iota_i \sim \iota_j$ die erzeugte Kongruenz nicht verändert.

- Schritt(2): Mit den Regeln (K) und (v) ist $\sigma_1 \to \sigma_2 \approx \tau_1 \to \tau_2$ genau dann, wenn $\sigma_1 \approx \tau_1$ und $\sigma_2 \approx \tau_2$ gelten. Daraus folgt die Behauptung.

- Schritt(3): Unter der Annahme $\iota \approx \psi$ gilt mit den Regeln (s) und (t) auch sicherlich $\iota \approx \varphi$ genau dann, wenn $\varphi \approx \psi$ gilt. Daraus folgt die Behauptung.

- Schritt (4): Wegen der Regel (r) macht auch dieser Fall keine Probleme.

Da nach Beendigung des Algorithmus $\mathrm{F}_i = 1$ für alle $i = 1, \ldots, n$ gilt, hat die Relation $\sim'$ gerade die gewünschte Form. Wegen obigen Überlegungen ist $\underline{\mathcal{F}}_{\mathcal{J}}/_{\approx} \cong \underline{\mathcal{F}}_{\mathcal{J}}/_{\approx'}$. $\square$

Diesen Algorithmus wollen wir an einem Beispiel veranschaulichen:

Beispiel: Sei $\mathcal{J} = \{\iota_1, \iota_2, \iota_3, \iota_4, \iota_5\}$ und $\sim$ wie folgt definiert:

$$\begin{array}{rcl}
\iota_2 & \sim & \iota_5 \\
\iota_1 \to \iota_2 & \sim & (\iota_2 \to \iota_1) \to \iota_3 \to \iota_5 \\
\iota_1 \to \iota_4 & \sim & \iota_4 \\
\iota_3 \to \iota_3 & \sim & \iota_3 \to \iota_3 \\
\iota_2 \to ((\iota_1 \to \iota_4) \to \iota_3) \to \iota_1 \to \iota_4 & \sim & \iota_2 \to \iota_3 \to \iota_4 \\
\iota_5 & \sim & \iota_2 \\
(\iota_2 \to \iota_4) \to \iota_1 & \sim & ((\iota_3 \to \iota_5) \to (\iota_2 \to \iota_1) \to \iota_4) \to \iota_5 \to \iota_1 \\
\iota_3 \to \iota_2 & \sim & (\iota_4 \to \iota_3) \to \iota_2
\end{array}$$

Schritt (1) von Algorithmus 2.27 formt diese Relation um in:

$$\begin{array}{rcl}
\iota_2 & \sim & \iota_5 \\
\iota_1 \to \iota_5 & \sim & (\iota_5 \to \iota_1) \to \iota_3 \to \iota_5 \\
\iota_4 & \sim & \iota_1 \to \iota_4 \\
\iota_5 \to ((\iota_1 \to \iota_4) \to \iota_3) \to \iota_1 \to \iota_4 & \sim & \iota_5 \to \iota_3 \to \iota_4 \\
(\iota_5 \to \iota_4) \to \iota_1 & \sim & ((\iota_3 \to \iota_5) \to (\iota_5 \to \iota_1) \to \iota_4) \to \iota_5 \to \iota_1 \\
\iota_3 \to \iota_5 & \sim & (\iota_4 \to \iota_3) \to \iota_5
\end{array}$$

Schritt (2) liefert:

$$\begin{array}{rcl}
\iota_2 & \sim & \iota_5 \\
\iota_1 & \sim & \iota_5 \to \iota_1 \\
\iota_5 & \sim & \iota_3 \to \iota_5 \\
\iota_4 & \sim & \iota_1 \to \iota_4 \\
\iota_3 & \sim & (\iota_1 \to \iota_4) \to \iota_3 \\
\iota_4 & \sim & (\iota_5 \to \iota_1) \to \iota_4 \\
\iota_3 & \sim & \iota_4 \to \iota_3
\end{array}$$

Die Schritte (a) bis (d) des ersten Durchlaufs von Schritt (3) bewirken lediglich, daß $\iota_3 \sim (\iota_1 \to \iota_4) \to \iota_3$ durch $(\iota_1 \to \iota_4) \to \iota_3 \sim \iota_4 \to \iota_3$ ersetzt wird. Nach Schritt (3e) erhält man:

$$\begin{array}{lcl} \iota_2 & \sim & \iota_5 \\ \iota_1 & \sim & \iota_5 \to \iota_1 \\ \iota_5 & \sim & \iota_3 \to \iota_5 \\ \iota_4 & \sim & \iota_1 \to \iota_4 \\ \iota_4 & \sim & (\iota_5 \to \iota_1) \to \iota_4 \\ \iota_3 & \sim & \iota_4 \to \iota_3 \end{array}$$

Nach einem zweiten Durchlauf endet der Algorithmus schließlich nach Schritt (4) mit:

$$\begin{array}{lcl} \iota_1 & \sim' & \iota_5 \to \iota_1 \\ \iota_2 & \sim' & \iota_5 \\ \iota_3 & \sim' & \iota_4 \to \iota_3 \\ \iota_4 & \sim' & \iota_1 \to \iota_4 \\ \iota_5 & \sim' & \iota_3 \to \iota_5 \end{array}$$

Algorithmus 2.27 liefert einen konstruktiven Beweis für die Umkehrung von Korollar 2.26 im Falle endlich repräsentierbarer Typsysteme. Dies wird durch folgendes Korollar ausgedrückt:

Korollar 2.31 *Jedes endlich repräsentierbare K-Typsystem ist zu einem von einer Relation* $\{\iota_i \sim \tau_i \mid i = 1, \ldots, n, \iota_i \in \mathcal{J}, \tau_i \in \mathcal{F}_{\mathcal{J}}\}$ *erzeugten Typsystem* $(\underline{\mathcal{F}}_{\mathcal{J}}, \approx)$ *isomorph.*

3 Starke Normalisierbarkeit monomorph getypter λ-Terme

In der *Funktionalen Programmierung* betrachtet man λ-Terme als Programme, die zu gewissen Eingaben entsprechende Ausgaben liefern. Die Berechnung der Ausgaben vollzieht sich im λ-Kalkül durch β- bzw. $\beta\eta$-Reduktionen. Dies legt es nahe, den Begriff der *Normalform* einzuführen, mit dem Endergebnisse von β- bzw. $\beta\eta$-Reduktionen bezeichnet werden. Aber auch aus mathematischer Sicht ist dieser Begriff von Interesse: Normalformen sind kanonische Repräsentanten der β- bzw. $\beta\eta$-Äquivalenzklassen von λ-Termen. In diesem Kapitel soll untersucht werden, bezüglich welcher monomorphen Typsysteme β- und $\beta\eta$-Reduktionen von λ-Termen stets terminieren.

Definition 3.1 *Ein λ-Term ist in **β-Normalform**, falls er keinen β-Redex enthält. Entsprechend ist ein λ-Term in **η-Normalform** bzw. in **$\beta\eta$-Normalform**, falls er keinen η- bzw. $\beta\eta$-Redex enthält.*

Jeder λ-Term läßt sich durch eine endliche Folge von η-Reduktionen in η-Normalform überführen, da jede η-Reduktion die Länge des Terms verkürzt. Somit hat jeder λ-Term, der eine β-Normalform besitzt, auch eine $\beta\eta$-Normalform. Im folgenden sei daher mit *Normalform* stets *$\beta\eta$-Normalform* gemeint.

Definition 3.2 *Ein λ-Term t heißt **normalisierbar**, falls es einen λ-Term s in Normalform gibt mit $t \xrightarrow[\beta\eta]{} \!\!\!\!\!\rightarrow s$. Falls jede Folge von Reduktionen in t terminiert, heißt t **stark normalisierend**. Der bezüglich $(\underline{\mathcal{F}}, \approx)$ getypte $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül heißt **stark normalisierend**, falls alle mit $(\underline{\mathcal{F}}, \approx)$ typisierbaren λ-Terme stark normalisierend sind.*

Natürlich ist jeder stark normalisierende λ-Term auch normalisierbar.

Beispiel:

a) $(\lambda xyz.xz(yz))(\lambda xy.x)$ ist stark normalisierend:

$$(\lambda xyz.xz(yz))(\lambda xy.x) \xrightarrow[\beta]{} \lambda yz.(\lambda xy.x)z(yz) \xrightarrow[\beta]{} \!\!\!\!\!\rightarrow \lambda yz.z$$

Dies ist die einzig mögliche Folge von Reduzierungen.

b) $\Omega := (\lambda x.xx)(\lambda x.xx)$ ist nicht normalisierbar:

$$\Omega \xrightarrow[\beta]{} \Omega \xrightarrow[\beta]{} \Omega \xrightarrow[\beta]{} \quad \cdots .$$

c) $(\lambda xy.y)\Omega(\lambda z.z)$ ist normalisierbar, aber nicht stark normalisierend:

$$\begin{tikzcd}
 & & (\lambda xy.y)\Omega(\lambda z.z) \arrow[dl, "\beta"] \arrow[dr, "\beta"'] & & & & \\
 & (\lambda y.y)(\lambda z.z) \arrow[dl, "\beta"] & & (\lambda xy.y)\Omega(\lambda z.z) \arrow[dl, "\beta"] \arrow[dr, "\beta"'] & & & \\
\lambda z.z & & (\lambda y.y)(\lambda z.z) \arrow[dl, "\beta"] & & (\lambda xy.y)\Omega(\lambda z.z) \arrow[dl, "\beta"] \arrow[dr, "\beta"'] & & \\
 & \lambda z.z & & (\lambda y.y)(\lambda z.z) \arrow[dl, dotted, no head] & & (\lambda xy.y)\Omega(\lambda z.z) \arrow[dl, dotted, no head] \arrow[dr, dotted, no head] & \\
 & & {} & & {} & & {}
\end{tikzcd}$$

Die Forderung, daß syntaktisch korrekte funktionale Programme stets terminieren, findet sich im λ-Kalkül in der starken Normalisierbarkeit wieder. Ziel ist es, eine möglichst allgemeine Charakterisierung derjenigen Typsysteme zu finden, die starke Normalisierbarkeit garantieren. Vorher benötigen wir jedoch noch einige Definitionen. Wie in Kapitel 2 betrachten wir Typsysteme der Form $(\underline{\mathcal{F}_{\mathcal{J}}}, \approx)$.

Definition 3.3 *Zu einem Typ $\tau \in \mathcal{F}$ betrachten wir die induktiv definierten Mengen $Pos(\tau)$ und $Neg(\tau)$. Für Grundtypen $\iota \in \mathcal{J}$ ist*

$$\begin{aligned} Pos(\iota) &= \{\iota\} \\ Neg(\iota) &= \phi \end{aligned}$$

und für Funktionentypen $\sigma \to \tau$ gilt:

$$\begin{aligned} Pos(\sigma \to \tau) &= \{\sigma \to \tau\} \cup Neg(\sigma) \cup Pos(\tau) \\ Neg(\sigma \to \tau) &= Pos(\sigma) \cup Neg(\tau) \end{aligned}$$

*σ kommt **positiv** in τ vor, falls $\sigma \in Pos(\tau)$; analog tritt σ **negativ** in τ auf, falls $\sigma \in Neg(\tau)$.*

Man kann Positivität auch anders verstehen, wenn man Typen als binäre Bäume betrachtet:

Definition 3.4 *Sei $\mathcal{J}$ die Erzeugendenmenge von $\underline{\mathcal{F}}$ und $\tau \in \mathcal{F}$. Der **Typbaum** $T(\tau)$ ist induktiv definiert durch:*

$$
\begin{array}{rcl}
T(\tau) & = & \tau \quad \textit{für } \tau \in \mathcal{J} \\
T(\tau) & = & \begin{array}{ccc} & \to & \\ / & & \backslash \\ T(\tau_1) & & T(\tau_2) \end{array} \qquad \textit{für } \tau = \tau_1 \to \tau_2
\end{array}
$$

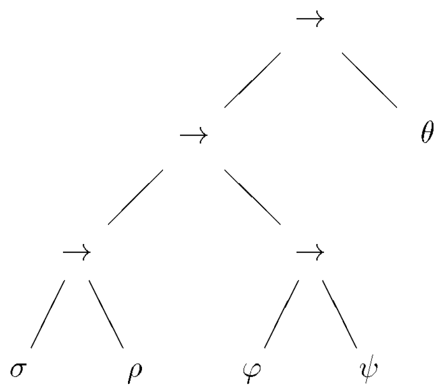
*Der oberste Pfeil eines Typbaums heißt **Wurzel**.*

Damit wird jedem Typ eineindeutig ein Typbaum zugeordnet.

Lemma 3.5 *Ein Vorkommen eines Teiltyps σ eines Typs $\tau \in \mathcal{F}$ ist genau dann positiv, falls im Kantenzug von* $\mathrm{T}(\sigma)$ *zur Wurzel im Typbaum* $\mathrm{T}(\tau)$ *die Anzahl der Kanten von links unten nach rechts oben gerade ist.*

Beweis: Der Beweis verläuft mit Induktion über den Typaufbau von τ. $\square$

Beispiel: Der Typbaum von $\tau = ((\sigma \to \rho) \to (\varphi \to \psi)) \to \theta$ ist:



Daher sind ρ, φ, θ und $\sigma \to \rho$ genau die positiv auftretenden Teiltypen von τ.

Definition 3.6 *Eine Relation $\sim$ heißt **negativ**, falls es $\sigma, \tau \in \mathcal{F}_{\mathcal{J}}$ mit $\sigma \approx \tau$ derart gibt, daß σ negativ in τ auftritt. Andernfalls heißt $\sim$ **positiv**.*

Nun sind wir in der Lage, ein notwendiges Kriterium für die starke Normalisierbarkeit des bezüglich $(\underline{\mathcal{F}}, \approx)$ getypten λ-Kalküls zu formulieren:

Satz 3.7 *Ist für ein Typsystem $(\underline{\mathcal{F}}, \approx)$ der $\lambda_{(\underline{\mathcal{F}},\approx)}$-Kalkül stark normalisierend, so ist die Relation $\approx$ positiv.*

Beweis: Wir nehmen an, es gibt $\sigma, \tau \in \mathcal{F}$ mit $\sigma \approx \tau$, wobei σ negativ in τ auftritt. Daraus konstruieren wir einen nicht stark normalisierenden $\lambda_{(\underline{\mathcal{F}},\approx)}$-Term.

Da σ negativ in τ auftritt, ist nach Lemma 3.5 im Typbaum $\mathrm{T}(\tau)$ die Anzahl der Kanten von links unten nach rechts oben im Kantenzug von $\mathrm{T}(\sigma)$ zur Wurzel ungerade. Damit steht rechts von σ eine ungerade Anzahl von Typen σ_i, die ihrerseits wieder in mehrere Teiltypen zerfallen können. Unter Berücksichtigung der Typen links von σ, läßt sich τ also darstellen als

$$\tau = \tau_{2k+1} = \rho^1_{2k+1} \to \ldots \to \rho^{n_{2k+1}}_{2k+1} \to \tau_{2k} \to \sigma_{2k+1}$$

mit

$$\begin{aligned}
\tau_{2k} &= \rho^1_{2k} \to \ldots \to \rho^{n_{2k}}_{2k} \to \tau_{2k-1} \to \sigma_{2k},\\
&\vdots\\
\tau_1 &= \rho^1_1 \to \ldots \to \rho^{n_1}_1 \to \tau_0 \to \sigma_1,\\
\tau_0 &= \rho^1_0 \to \ldots \to \rho^{n_0}_0 \to \sigma.
\end{aligned}$$

Zu jedem Typ τ_i mit $i = 0, \ldots, 2k+1$ geben wir einen λ-Term t_i an, so daß t_i bezüglich einer festen Basis Γ mit τ_i typisierbar ist. Daraus wird ein nicht normalisierbarer λ-Term t konstruiert. Setze

$$\begin{aligned}
t_0 &= \lambda x^1_0. \ldots x^{n_0}_0 . t_{2k+1}\\
t_1 &= \lambda x^1_1. \ldots x^{n_1}_1 y_0 . v(y_0 z^1_0 \ldots z^{n_0}_0 z^1_{2k+1} \ldots z^{n_{2k+1}}_{2k+1} y_{2k})
\end{aligned}$$

und für $k > 0$:

$$\begin{aligned}
t_2 &= \lambda x^1_2. \ldots x^{n_2}_2 y_1 . w_1(y_1 z^1_1 \ldots z^{n_1}_1 t_0)\\
&\vdots\\
t_{2k+1} &= \lambda x^1_{2k+1}. \ldots x^{n_{2k+1}}_{2k+1} y_{2k} . w_{2k}(y_{2k} z^1_{2k} \ldots z^{n_{2k}}_{2k} t_{2k-1}).
\end{aligned}$$

Dies ist keine zyklische Definition. Jeder λ-Term t_i mit $i > 1$ enthält nur t_{i-2} explizit als Teilterm. Die Definition von t_0 greift lediglich auf t_{2k+1} zurück, so daß alle oben definierten Terme nur aus Variablen und t_1 bestehen. Da t_1 seinerseits nur aus Variablen besteht, wird tatsächlich für jedes $i = 0, \ldots, 2k+1$ durch t_i ein λ-Term definiert. Setze

$$t = t_{2k+1} z^1_{2k+1} \ldots z^{n_{2k+1}}_{2k+1} t_{2k}.$$

Da keine der Variablen $y_0, \ldots, y_{2k-1}$ in einem der $t_0, \ldots, t_{2k+1}$ frei vorkommt, gilt $s[t_i/y_i] = s$ für jede Substitution $s[t_i/y_i]$ mit $i < 2k$. Somit ist t nicht stark normalisierend, denn es gilt:

$$
\begin{array}{lll}
t & \twoheadrightarrow_\beta & w_{2k}(t_{2k}z_{2k}^1 \ldots z_{2k}^{n_{2k}}(t_{2k-1}[t_{2k}/y_{2k}])) \\
 & \twoheadrightarrow_\beta & w_{2k}(w_{2k-1}(t_{2k-1}[t_{2k}/y_{2k}])z_{2k-1}^1 \ldots z_{2k-1}^{n_{2k-1}}t_{2k-2})) \\
 & \twoheadrightarrow_\beta & w_{2k}(w_{2k-1}(w_{2k-2}(t_{2k-2}z_{2k-2}^1 \ldots z_{2k-2}^{n_{2k-2}}(t_{2k-3}[t_{2k}/y_{2k}])))) \\
 & \twoheadrightarrow_\beta & \ldots \\
 & \twoheadrightarrow_\beta & w_{2k}(w_{2k-1} \ldots (w_2(w_1((t_1[t_{2k}/y_{2k}])z_1^1 \ldots z_1^{n_1}t_0))) \cdots) \\
 & \twoheadrightarrow_\beta & w_{2k}(w_{2k-1} \ldots (w_2(w_1(v(t_0z_0^1 \ldots z_0^{n_0}z_{2k+1}^1 \ldots z_{2k+1}^{n_2k+1}t_{2k})))) \cdots) \\
 & \twoheadrightarrow_\beta & w_{2k}(w_{2k-1} \ldots (w_2(w_1(v(t_{2k+1}z_{2k+1}^1 \ldots z_{2k+1}^{n_{2k+1}}t_{2k})))) \cdots),
\end{array}
$$

also

$$t \twoheadrightarrow_\beta w_{2k}(w_{2k-1} \ldots (w_2(w_1(vt))) \cdots).$$

Jeder der obigen reduzierten λ-Terme hat genau einen Redex, somit ist t nicht normalisierbar und damit auch nicht stark normalisierend. Nichtsdestotrotz ist typisierbar: Sei

$$
\begin{array}{rcl}
\Gamma & := & \{(v : \sigma_{2k+1} \to \sigma_1)\} \cup \{(w_i : \sigma_i \to \sigma_{i+1}) \mid i = 1, \ldots, 2k\} \\
 & & \cup \{(y_i : \tau_i) \mid i = 0, \ldots, 2k\} \cup \{(z_i^j : \rho_i^j) \mid i = 0, \ldots, 2k+1,\ j = 1, \ldots, n_i\}
\end{array}
$$

und

$$\Gamma^* := \Gamma \cup \{(x_i^j : \rho_i^j) \mid i = 0, \ldots, 2k+1,\ j = 1, \ldots, n_i\}.$$

Daraus folgt

$$\Gamma^* \vdash y_0 : \rho_0^1 \to \ldots \to \rho_0^{n_0} \to \sigma \quad \text{und} \quad \Gamma^* \vdash z_0^i : \rho_0^i$$

für $i = 1, \ldots, n_0$. Mehrfaches Anwenden von (App) liefert

$$\Gamma^* \vdash y_0z_0^1 \ldots z_0^{n_0} : \sigma$$

und mit (Kon) erhält man

$$\Gamma^* \vdash y_0z_0^1 \ldots z_0^{n_0} : \rho_{2k+1}^1 \to \ldots \to \rho_{2k+1}^{n_{2k+1}} \to \tau_{2k} \to \sigma_{2k+1}.$$

Wie oben folgt

$$\Gamma^* \vdash y_0z_0^1 \ldots z_0^{n_0}z_{2k+1}^1 \ldots z_{2k+1}^{n_{2k+1}}y_{2k} : \sigma_{2k+1}$$

und weiter

$$\Gamma^* \vdash v(y_0z_0^1 \ldots z_0^{n_0}z_{2k+1}^1 \ldots z_{2k+1}^{n_{2k+1}}y_{2k}) : \sigma_1.$$

Mehrfaches Anwenden von (Abs) liefert

$$\Gamma \vdash \lambda x_1^1 \dots x_1^{n_1} y_0.v(y_0 z_0^1 \dots z_0^{n_0} z_{2k+1}^1 \dots z_{2k+1}^{n_{2k+1}} y_{2k}) : \rho_1^1 \to \dots \to \rho_1^{n_1} \to \tau_0 \to \sigma_1,$$

also $\Gamma \vdash t_1 : \tau_1$. Sukzessive erhält man mit analoger Vorgehensweise $\Gamma \vdash t_3 : \tau_3$, $\Gamma \vdash t_5 : \tau_5, \dots, \Gamma \vdash t_{2k+1} : \tau_{2k+1}$, $\Gamma \vdash t_0 : \tau_0$, $\Gamma \vdash t_2 : \tau_2, \dots, \Gamma \vdash t_{2k} : \tau_{2k}$, also $\Gamma \vdash t_i : \tau_i$ für alle $i = 0, \dots, 2k+1$. Schließlich folgt $\Gamma \vdash t : \sigma_{2k+1}$. Damit ist t ein bezüglich $(\underline{\mathcal{F}}, \approx)$ typisierbarer λ-Term, der nicht normalisierbar ist; daraus folgt die Behauptung. □

Die Idee dieses Beweises stammt von N. Mendler, siehe [Men91]. Leider schränkt Satz 3.7 die Mächtigkeit monomorpher Typsysteme etwas ein:

Korollar 3.8 *Es gibt kein Typsystem $(\underline{\mathcal{F}}, \approx)$, so daß die $\lambda_{(\underline{\mathcal{F}}, \approx)}$-Terme genau den stark normalisierenden Termen entsprechen.*

Beweis: Sicherlich ist $t = xx$ mit $x \in X$ stark normalisierend, da t bereits in Normalform ist. Wäre t typisierbar, so gäbe es nach dem Erzeugungslemma 2.11 eine Basis Γ und $\tau_1, \tau_2 \in \mathcal{F}$ mit $\Gamma \vdash x : \tau_1 \to \tau_2$ und $\Gamma \vdash x : \tau_1$. Erneut nach 2.11 gibt es ein $\tau \in \mathcal{F}$ mit $\tau_1 \to \tau_2 \approx \tau \approx \tau_1$, also $\tau_1 \to \tau_2 \approx \tau_1$. Da τ_1 negativ in $\tau_1 \to \tau_2$ vorkommt, ist der $\lambda_{(\underline{\mathcal{F}}, \approx)}$-Kalkül nach Korollar 3.8 nicht stark normalisierend. □

Folgender Satz liefert ein Kriterium für starke Normalisierbarkeit. Nach Korollar 2.31 bedeutet die im Satz gemachte Voraussetzung für endlich repräsentierbare K-Typsysteme keine große Einschränkung.

Satz 3.9 *Sei $\underline{\mathcal{F}_\mathcal{J}}$ ein freies Gruppoid, zu jedem $\iota_i \in \mathcal{J} = \{\iota_i \mid i \in I\}$ gebe es genau ein $\tau_i \in \mathcal{F}_\mathcal{J}$ mit $\iota_i \sim \tau_i$. Ist $\sim$ positiv, so ist der $\lambda_{(\underline{\mathcal{F}}, \approx)}$-Kalkül stark normalisierend.*

Beweis: Siehe [Men91]. □

Leider ist es oft nicht offensichtlich, ob eine Relation der Form $\{\iota_i \sim \tau_i \mid i \in I\}$ auf einem freien Gruppoid $\underline{\mathcal{F}_\mathcal{J}}$ positiv ist. Um ein Kriterium für die Positivität der Relation $\sim$ zu geben, wollen wir zusätzlich $\tau_i \notin \mathcal{J}$ für alle $i \in I$ fordern. Dies ist keine wesentliche Einschränkung, da es für jedes Gruppoid $\underline{\mathcal{F}_\mathcal{J}}/_\approx$ eine Menge $\mathcal{J}'$ mit einer Relation $\sim'$ gibt, so daß $\underline{\mathcal{F}_\mathcal{J}}/_\approx$ und $\underline{\mathcal{F}_{\mathcal{J}'}}/_{\approx'}$ isomorph sind und $\sim'$ die obige Voraussetzung erfüllt.

Definition 3.10 *Sei $\sim$ eine Relation auf dem Gruppoid $\underline{\mathcal{F}}_{\mathcal{J}}$. Der* ***Typengraph*** *$\mathcal{G}_{\mathcal{J}}^{\sim}$ ist ein mit $\{-1,1\}$ bewerteter gerichteter Graph mit der Eckenmenge $\mathcal{F}_{\mathcal{J}}$ und der wie folgt definierten Kantenmenge:*

- $\sigma \xrightarrow{-1} \tau$, *falls σ negativ in τ vorkommt*
- $\sigma \xrightarrow{1} \tau$, *falls $\sigma \neq \tau$ und σ positiv in τ vorkommt*
- $\sigma \overset{1}{\longleftrightarrow} \tau$, *falls $\sigma \neq \tau$ und τ dadurch entsteht, daß in σ das Vorkommen eines Teiltyps φ durch ψ ersetzt wird, wobei $\varphi \sim \psi$ oder $\psi \sim \varphi$.*

Für $\sigma \underset{1}{\overset{1}{\rightleftarrows}} \tau$ schreiben wir einfach $\sigma \overset{1}{\longleftrightarrow} \tau$. Der Typengraph von $\underline{\mathcal{F}}_{\mathcal{J}}/_{\approx}$ enthält keine Schleifen $\tau \xrightarrow{x} \tau$ $(x \in \{1,-1\})$ und keine negativ bewerteten Doppelkanten $\sigma \underset{-1}{\overset{-1}{\rightleftarrows}} \tau$.

Sei $\underline{\mathcal{F}}_{\mathcal{J}}$ ein von $\mathcal{J} = \{\iota_i \,|\, i \in I\}$ frei erzeugtes Gruppoid und $\{\iota_i \sim \tau_i \,|\, i \in I,\, \iota_i \in \mathcal{J},\, \tau_i \in \mathcal{F}_{\mathcal{J}} \setminus \mathcal{J}\}$ eine Relation. Falls eines der ι_i negativ in τ_i für $i \in I$ vorkommt so ist $\sim$ offensichtlich negativ. Daher wollen wir voraussetzen, daß $\sim$ diese Eigenschaft nicht hat.

Sei σ^k die Abkürzung für

$$\sigma_{k,1} \overset{1}{\longleftrightarrow} \sigma_{k,2} \overset{1}{\longleftrightarrow} \ldots \overset{1}{\longleftrightarrow} \sigma_{k,n_k}$$

mit $k = 1,\ldots,m$. Folgendes Verfahren findet zu einem geschlossenen Kantenzug

$$\mathcal{K} \;=\; \sigma_{1,n_1} \xrightarrow{x_1} \sigma^2 \xrightarrow{x_2} \ldots \xrightarrow{x_{m-1}} \sigma^m \xrightarrow{x_m} \sigma^1$$

mit $x_k \in \{-1,1\}$, $k = 1,\ldots,m$, im Typengraph $\mathcal{G}_{\mathcal{J}}^{\sim}$ einen Kantenzug

$$\iota_{i_1} \xrightarrow{y_1} \tau_{i_2} \overset{1}{\longleftrightarrow} \iota_{i_2} \xrightarrow{y_2} \ldots \overset{1}{\longleftrightarrow} \iota_{i_n} \xrightarrow{y_n} \tau_{i_1} \overset{1}{\longleftrightarrow} \iota_{i_1}$$

mit $\iota_{i_j} \in \mathcal{J}$ und $\tau_{i_j} \in \mathcal{F}_{\mathcal{J}}$, $j = 1,\ldots,n$, so daß $x_1 \cdot \ldots \cdot x_m = y_1 \cdot \ldots \cdot y_n$. Die Tatsache, daß in diesem Kantenzug nur Doppelkanten der Form $\tau \overset{1}{\longleftrightarrow} \iota$ mit $\tau \in \mathcal{F}_{\mathcal{J}}$ und $\iota \in \mathcal{J}$ enthalten sind, wird für den Beweis der Gültigkeit des Kriteriums zur Charakterisierung der Positivität entscheidend sein.

Algorithmus 3.11

- *Solange $\mathcal{K}$ noch Kanten $\sigma \overset{1}{\longleftrightarrow} \sigma'$ mit $\sigma \not\sim \sigma'$ und $\sigma' \not\sim \sigma$ enthält:*
  - *(1) Ersetze alle Kanten der Form $\iota_i \overset{1}{\longleftrightarrow} \tau_i \overset{1}{\longleftrightarrow} \iota_i$ durch ι_i und alle Kanten der Form $\tau_i \overset{1}{\longleftrightarrow} \iota_i \overset{1}{\longleftrightarrow} \tau_i$ durch τ_i.*

(2) *Falls $\mathcal{K}$ einen Kantenzug $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{x} \sigma \xleftrightarrow{1} \sigma'$ mit $\sigma' \notin \mathcal{J}$ enthält, so gibt es vier mögliche Fälle:*

   (a) *$x = 1$ und ι_i kommt positiv in σ' vor:*
   ersetze $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{1} \sigma \xleftrightarrow{1} \sigma'$ durch $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{1} \sigma'$.

   (b) *$x = -1$ und ι_i kommt negativ in σ' vor:*
   ersetze $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{-1} \sigma \xleftrightarrow{1} \sigma'$ durch $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{-1} \sigma'$.

   (c) *σ' entsteht durch Ersetzen eines τ_j durch ι_j in σ, und ι_i ist für $x = 1$ nicht positiv und für $x = -1$ nicht negativ in σ' enthalten: ersetze $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{x} \sigma \xleftrightarrow{1} \sigma'$ durch $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{y} \tau_j \xleftrightarrow{1} \iota_j \xrightarrow{z} \sigma'$ mit $y = x$ und $z = 1$, falls das zu ersetzende Vorkommen von τ_j in σ positiv ist, andernfalls mit $y = -x$ und $z = -1$.*

   (d) *σ' entsteht durch Ersetzen von ι_i durch τ_i in σ, und ι_i ist kein Teiltyp von τ_i: ersetze $\tau_i \xleftrightarrow{1} \iota_i \xrightarrow{x} \sigma \xleftrightarrow{1} \sigma'$ durch $\tau_i \xrightarrow{x} \sigma'$.*

(3) *Falls $\mathcal{K}$ keine Ecke aus $\mathcal{J}$ enthält, so ist $\sigma_{k,i} = \varphi_{k,i} \to \psi_{k,i}$ für alle $k = 1, \ldots, m$, $i = 1, \ldots, n_k$.*

   (a) *Setze $k = 1$.*

   (b) *Solange $k \leq m$: ersetze*

$$\sigma^k \xrightarrow{x_k} \varphi_{k+1,1} \to \psi_{k+1,1} \xleftrightarrow{1} \ldots \xleftrightarrow{1} \varphi_{k+1,n_{k+1}} \to \psi_{k+1,n_{k+1}} \xrightarrow{x_{k+1}} \sigma^{k+2}$$

   durch

   - *$\sigma^k \xrightarrow{-x_k} \varphi_{k+1,1} \xleftrightarrow{1} \ldots \xleftrightarrow{1} \varphi_{k+1,n_{k+1}} \xrightarrow{-x_{k+1}} \sigma^{k+2}$, falls σ^k in $\varphi_{k+1,k}$ enthalten ist,*
   - *$\sigma^k \xrightarrow{x_k} \psi_{k+1,1} \xleftrightarrow{1} \ldots \xleftrightarrow{1} \psi_{k+1,n_{k+1}} \xrightarrow{x_{k+1}} \sigma^{k+2}$, falls σ^k in $\psi_{k+1,k}$ enthalten ist,*

   und erhöhe k um 1.

   (c) *Ersetze alle Ausdrücke der Form $\sigma \xleftrightarrow{1} \sigma$ durch σ.*

(4) *Ersetze alle aufeinanderfolgenden einfachen Kanten $\sigma_1 \xrightarrow{x} \sigma_2 \xrightarrow{y} \sigma_3$ durch $\sigma_1 \xrightarrow{x \cdot y} \sigma_3$.*

- *Ersetze alle Kanten $\iota_i \xleftrightarrow{1} \tau_i \xleftrightarrow{1} \iota_i$ durch ι_i und alle Kanten $\tau_i \xleftrightarrow{1} \iota_i \xleftrightarrow{1} \tau_i$ durch τ_i.*

Satz 3.12 *Algorithmus 3.11 terminiert und ist korrekt.*

Beweis: Es ist klar, daß jeder einzelne Schritt des Verfahrens terminiert. Zum Beweis der Terminierung des gesamten Algorithmus muß also gezeigt werden, daß $\mathcal{K}$ nach endlich vielen Schritten keine Kante $\sigma \xleftrightarrow{1} \sigma'$ mit $\sigma \not\rightsquigarrow \sigma'$ und $\sigma' \not\rightsquigarrow \sigma$ enthält. Die Schritte (1) und (4) beeinflussen diese Eigenschaft sicherlich nicht. In jedem der vier Fälle von Schritt (2) wird die Anzahl der Kanten $\sigma \xleftrightarrow{1} \sigma'$ mit $\sigma \not\rightsquigarrow \sigma'$ und $\sigma' \not\rightsquigarrow \sigma$ um eins reduziert. In Schritt (3) wird die Länge der durch $\xleftrightarrow{1}$ benachbarten Typen gekürzt, eventuell auch hier die Anzahl der Kanten reduziert. Da in jedem Durchlauf des Algorithmus einer der Fälle (2a), (2b), (2c), (2d) oder (3) eintritt, bricht die Schleife nach endlicher Zeit ab, und der Algorithmus terminiert.

Die strukturelle Eigenschaft des Kantenzugs, keine zwei aufeinanderfolgenden einfachen Pfeile zu beinhalten, bleibt in jedem Schleifendurchlauf erhalten. Am Ende wird sichergestellt, daß keine aufeinander folgenden Doppelpfeile im Kantenzug enthalten sind. Auch das Produkt der Bewertungen ist in jedem Schritt des Algorithmus invariant. Zu zeigen ist, daß zu jedem Zeitpunkt nur dann eine Kante $\sigma \xrightarrow{1} \tau$ in $\mathcal{K}$ enthalten ist, wenn σ positiv in τ vorkommt und $\sigma \xrightarrow{-1} \tau$ nur dann, wenn σ negativ in τ auftritt. Dies gilt zu Beginn des Ablaufs des Algorithmus und ändert sich sicherlich nicht in den Schritten (1), (2a), (2b) und (4). In Schritt (2c) unterscheiden wir zwei Fälle: Kommt das zu ersetzende τ_j positiv in σ vor, so ist das Vorkommen von ι_i in τ_j genauso wie das von ι_i in σ, und ι_j kommt auch positiv in σ' vor. Andernfalls ist das Vorkommen von ι_i in τ_i anders als das von ι_i in σ, und ι_j kommt negativ in σ' vor. Dies wird durch die Gleichungen $y = x$ und $z = 1$ bzw. $y = -x$ und $z = -1$ ausgedrückt. In Schritt (2d) kommt τ_j genauso in σ' wie ι_i in σ vor. Der für obige Frage einzig relevante Teilschritt in Schritt (3) ist Teil (b). Im ersten Fall ist das Vorkommen von σ_{k,n_k} in $\varphi_{k+1,1}$ bzw. von $\varphi_{k+1,n_{k+1}}$ in $\sigma_{k+2,1}$ anders als das von σ_{k,n_k} in $\varphi_{k+1,1} \to \psi_{k+1,1}$ bzw. von $\varphi_{k+1,n_{k+1}} \to \psi_{k+1,n_{k+1}}$ in $\sigma_{k+2,1}$. Im zweiten Fall ändert sich nichts. Damit ist auch die Korrektheit von Algorithmus 3.11 bewiesen. $\square$

Wir kommen nun zu dem zentralen Ergebnis, das die Positivitätseigenschaft eines Typsystems charakterisiert:

Satz 3.13 *Sei $\{\iota_i \sim \tau_i \mid i \in I,\ \iota_i \in \mathcal{J},\ \tau_i \in \mathcal{F}_{\mathcal{J}} \setminus \mathcal{J}\}$ eine Relation auf dem Gruppoid $\underline{\mathcal{F}_{\mathcal{J}}}$. Der **Erzeugendengraph** $\mathcal{E}_{\mathcal{J}}^{\sim}$ von $\mathcal{F}_{\mathcal{J}}$ besteht aus der Eckenmenge $\mathcal{J} \cup \{\tau \in \mathcal{F} \mid \exists \iota \in \mathcal{J} \colon \iota \sim \tau\}$ und den Kanten*

- *$\sigma \xrightarrow{-1} \tau$, falls σ negativ in τ vorkommt,*
- *$\sigma \xrightarrow{1} \tau$, falls $\sigma \neq \tau$ und σ positiv in τ vorkommt,*
- *$\sigma \xleftrightarrow{1} \tau$, falls $\sigma \neq \tau$ und $\sigma \sim \tau$ oder $\tau \sim \sigma$.*

(Wieder mit der Konvention: $\sigma \xleftrightarrow{1} \tau$ statt $\sigma \overset{1}{\underset{1}{\rightleftarrows}} \tau$). Dann ist $\sim$ genau dann positiv, wenn für jeden Kreis

$$\iota_{i_1} \xrightarrow{x_1} \tau_{i_2} \xleftrightarrow{1} \iota_{i_2} \xrightarrow{x_2} \ldots \xleftrightarrow{1} \iota_{i_n} \xrightarrow{x_n} \tau_{i_1} \xleftrightarrow{1} \iota_{i_1}$$

in $\mathcal{E}_{\mathcal{J}}^{\sim}$ gilt: $x_1 \cdot \ldots \cdot x_n = 1$.

Beweis: Bezeichne $\sigma[\varphi/\psi]$ den Typ, der entsteht, wenn in σ jedes Vorkommen von ψ durch φ ersetzt wird. Sei

$$\iota_{i_1} \xrightarrow{x_1} \tau_{i_2} \xleftrightarrow{1} \iota_{i_2} \xrightarrow{x_2} \ldots \xleftrightarrow{1} \iota_{i_n} \xrightarrow{x_n} \tau_{i_1} \xleftrightarrow{1} \iota_{i_1}$$

ein Kreis in $\mathcal{E}_{\mathcal{J}}^{\sim}$ für den $x_1 \cdot \ldots \cdot x_n = -1$ gilt. Dann kommt ι_{i_1} negativ in

$$\tau_{i_1}[\tau_{i_n}[\tau_{i_{n-1}} \ldots [\tau_{i_2}/\iota_{i_2}]/\ldots/\iota_{i_{n-1}}]/\iota_n]$$

vor.

Sei umgekehrt $\sim$ negativ. Es gibt also $\sigma_1, \sigma_2 \in \mathcal{F}_{\mathcal{J}}$ mit $\sigma_1 \approx \sigma_2$, so daß σ_1 negativ in σ_2 vorkommt. Da $\approx$ die von $\sim$ erzeugte Kongruenzrelation ist, gibt es im Typengraph $\mathcal{G}_{\mathcal{J}}^{\sim}$ einen Kantenzug

$$\sigma_1 \xrightarrow{-1} \sigma_2 \xleftrightarrow{1} \sigma_3 \ldots \xleftrightarrow{1} \sigma_n \xleftrightarrow{1} \sigma_1.$$

Algorithmus 3.11 liefert einen Kantenzug

$$\iota_{i_1} \xrightarrow{x_1} \tau_{i_2} \xleftrightarrow{1} \iota_{i_2} \xrightarrow{x_2} \ldots \xleftrightarrow{1} \iota_{i_n} \xrightarrow{x_n} \tau_{i_1} \xleftrightarrow{1} \iota_{i_1}$$

mit $x_1 \cdot \ldots \cdot x_n = -1$. □

Aus Satz 3.13 folgt direkt:

Korollar 3.14 *Es gibt ein Entscheidungsverfahren dafür, ob eine endliche Relation $\{\iota_i \sim \tau_i \,|\, i = 1, \ldots, n,\ \iota_i \in \mathcal{J},\ \tau_i \in \mathcal{F}_{\mathcal{J}}\}$ positiv oder negativ ist.*

Beweis: Transformiere $\mathcal{J}$ in $\mathcal{J}'$ und $\sim$ in eine Relation $\{\iota_i \sim' \tau_i \,|\, i = 1, \ldots, m,\ \iota_i \in \mathcal{J}',\ \tau_i \in \mathcal{F}_{\mathcal{J}'} \setminus \mathcal{J}'\}$, so daß $\mathcal{F}_{\mathcal{J}}/_{\approx} \cong \mathcal{F}_{\mathcal{J}'}/_{\approx'}$. Anschließend bestimme im Erzeugendengraph $\mathcal{E}_{\mathcal{J}}^{\sim'}$ alle Kreise der Gestalt

$$\iota_{i_1} \xrightarrow{x_1} \tau_{i_2} \xleftrightarrow{1} \iota_{i_2} \xrightarrow{x_2} \ldots \xleftrightarrow{1} \iota_{i_m} \xrightarrow{x_m} \tau_{i_1} \xleftrightarrow{1} \iota_{i_1},$$

berechne das Produkt $x_1 \cdot \ldots \cdot x_n$ der Bewertungen und wende Satz 3.13 an. $\square$

Folgendes Beispiel veranschaulicht Korollar 3.14 und die Wirkungsweise von Algorithmus 3.11:

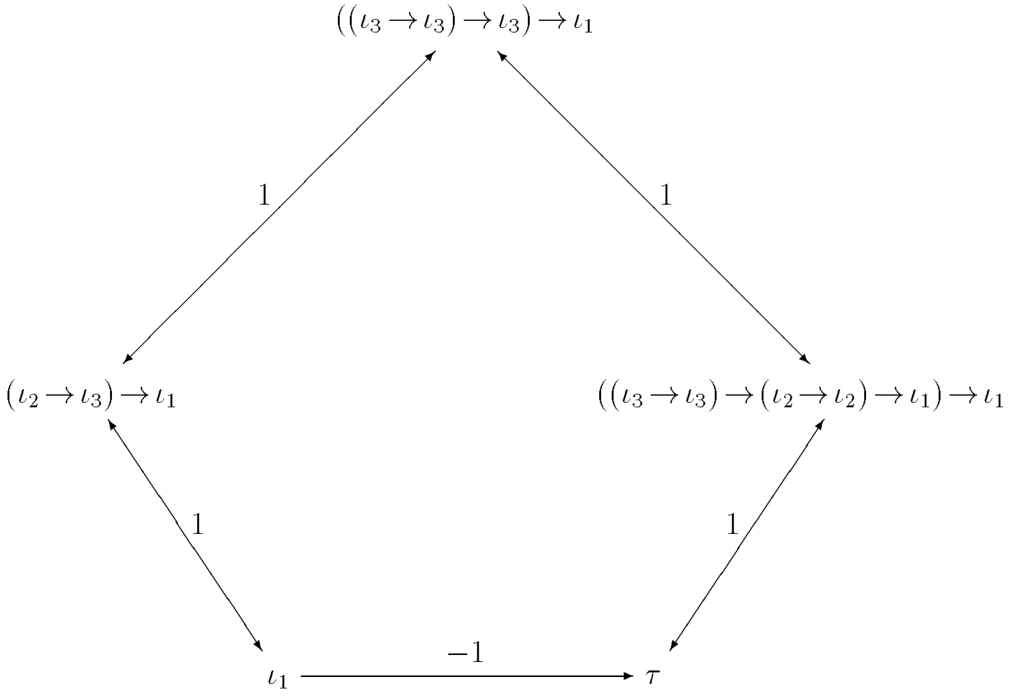
Beispiel: Sei $\mathcal{J} = \{\iota_1, \iota_2, \iota_3\}$ und $\sim$ wie folgt definiert:

$$\begin{array}{lcl}
\iota_1 & \sim & (\iota_2 \to \iota_3) \to \iota_1 \\
\iota_2 & \sim & \iota_3 \to \iota_3 \\
\iota_3 & \sim & (\iota_2 \to \iota_2) \to \iota_1
\end{array}$$

Obwohl keines der ι_i negativ in τ_i mit $i = 1, 2, 3$ vorkommt, ist ι_1 negativ in

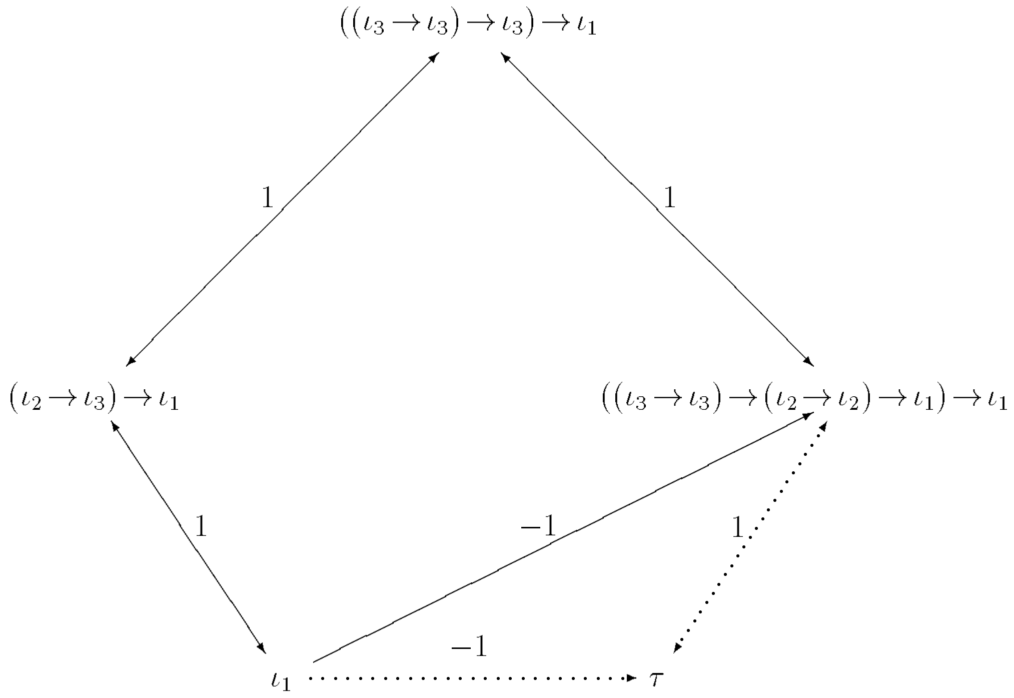
$$\tau := ((((\iota_2 \to \iota_2) \to \iota_1) \to \iota_3) \to (\iota_2 \to \iota_2) \to \iota_1) \to \iota_1 \approx \iota_1$$

enthalten. Der entsprechende Zyklus im Typengraph $\mathcal{G}_{\mathcal{J}}^{\sim}$ ist:

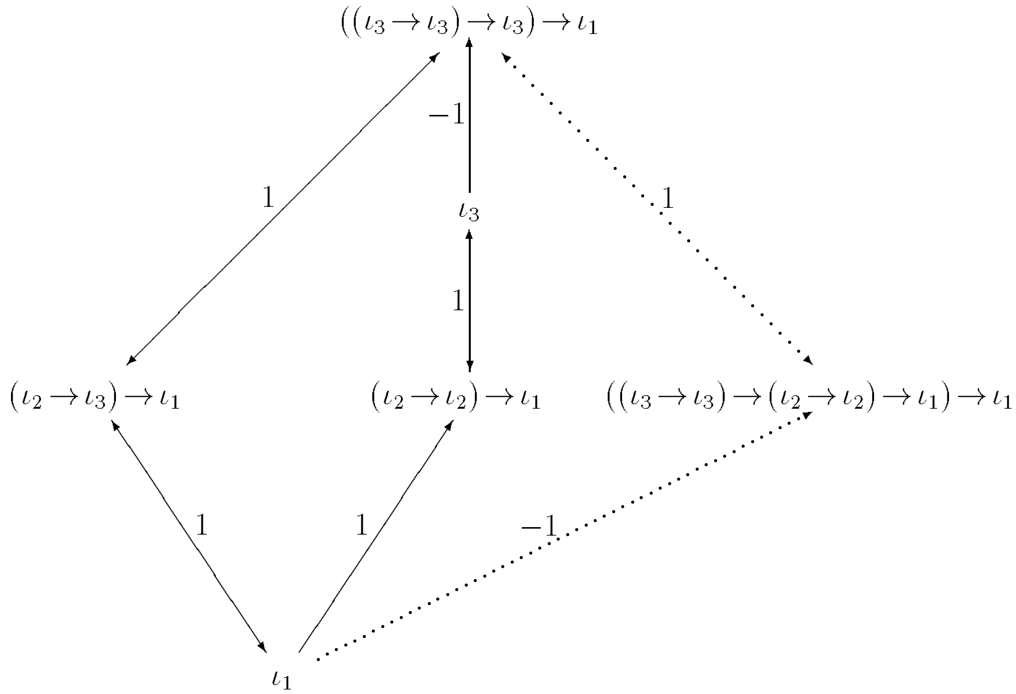


In den Schleifendurchläufen wird dieser Kantenzug wie folgt verändert:

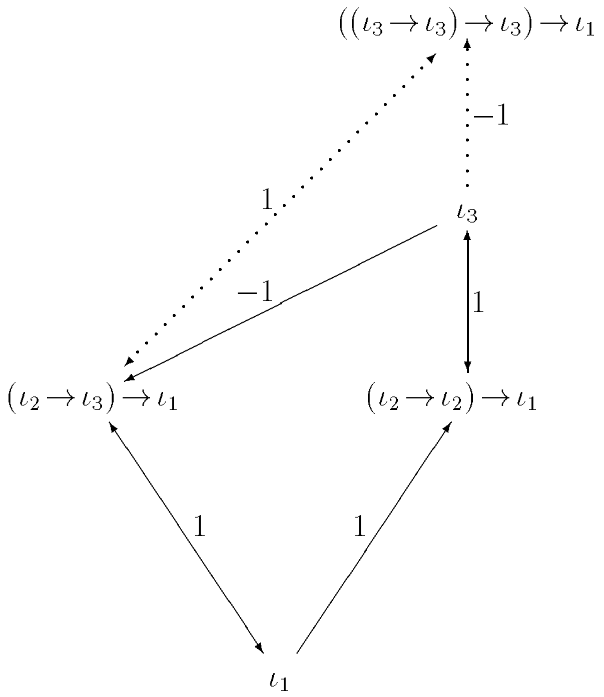
1. Durchlauf: Schritt (2b)



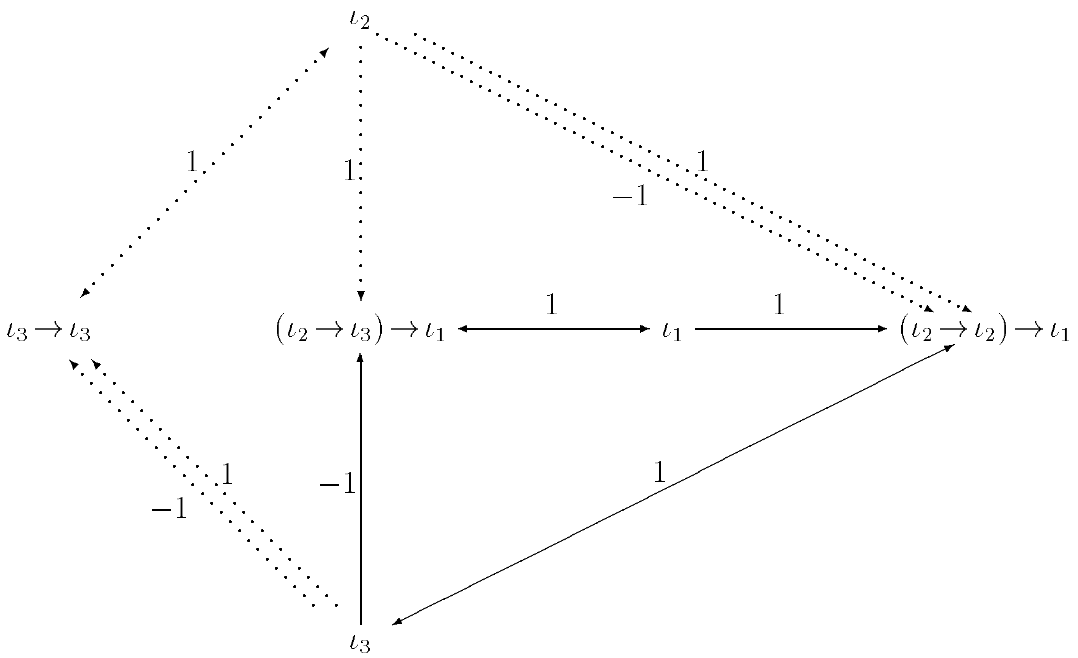
2. Durchlauf: Schritt (2c)



3. Durchlauf: Schritt (2b)



Nach drei Durchläufen endet die Schleife. Der entstandene Kantenzug ist auch ein Kreis im Erzeugendengraph $\mathcal{E}_{\mathcal{J}}^{\sim}$:



Wie im Ausgangsgraphen ist das Produkt der Bewertungen dieses Zyklus -1.

Zusammen mit Satz 2.30 und Satz 3.9 folgt aus Korollar 3.14:

Korollar 3.15 *Es gibt ein Entscheidungsverfahren dafür, ob der $\lambda_{(\underline{\mathcal{F}_\mathcal{J}},\approx)}$-Kalkül zu einem endlich repräsentierbaren K-Typsystem $(\underline{\mathcal{F}_\mathcal{J}},\approx)$ stark normalisierend ist.*

Beispiel: Der Erzeugendengraph $\mathcal{E}_\mathcal{J}^{\sim'}$ der Relation $\sim'$ aus dem Beispiel zu Algorithmus 2.27 ist:

$$\begin{tikzcd}
& & \iota_5\to\iota_1 \arrow[r, leftrightarrow, "1"] & \iota_1 \arrow[r, "-1"] & \iota_1\to\iota_4 \arrow[dr, leftrightarrow, "1"] & \\
\iota_2 \arrow[r, leftrightarrow, "1"] & \iota_5 \arrow[ur, "-1"] \arrow[dr, leftrightarrow, "1"'] & & & & \iota_4 \arrow[dl, "-1"] \\
& & \iota_3\to\iota_5 & \iota_3 \arrow[l, "-1"'] \arrow[r, leftrightarrow, "1"] & \iota_4\to\iota_3 &
\end{tikzcd}$$

Mit Satz 3.13 sieht man sehr leicht, daß $\sim'$ und damit auch die ursprüngliche Relation $\sim$ positiv sind. Satz 3.9 garantiert für dieses Typsystem starke Normalisierbarkeit.

Folgende Beispiele zeigen, daß durch die Einführung von positiven Typkongruenzen tatsächlich mehr λ-Terme typisiert werden können als im einfach getypten $\lambda_\to$-Kalkül:

Beispiel: Der λ-Term

$$t = \lambda xy.x(y(xy))$$

ist in $\lambda_\to$ nicht typisierbar. Wäre dies der Fall, so gäbe es eine Basis Γ und ein Typsystem $(\underline{\mathcal{F}},=)$ mit $\tau\in\mathcal{F}$ und $\Gamma\vdash x(y(xy)):\tau$. Nach dem Erzeugungslemma 2.11 gäbe es ein $\tau'\in\mathcal{F}$ mit $\Gamma\vdash x:\tau'\to\tau$ und $\Gamma\vdash y(xy):\tau'$. Erneut nach dem Erzeugungslemma 2.11 existierte ein $\sigma\in\mathcal{F}$ mit $\Gamma\vdash y:\sigma\to\tau'$ sowie $\Gamma\vdash xy:\sigma$. Schließlich folgt nochmals aus dem Erzeugungslemma 2.11 die Existenz eines σ' mit $\Gamma\vdash x:\sigma'\to\sigma$ und $\Gamma\vdash y:\sigma'$. Es gilt also $\tau'\to\tau=\sigma'\to\sigma$, also auch $\tau'=\sigma'$. Wegen $\sigma'=\sigma\to\tau'$ folgt $\tau'=\sigma\to\tau'$, ein Widerspruch.

Mit $\mathcal{F}:=\{\iota_1,\iota_2\}$, $\iota_1\sim\iota_2\to\iota_1$ und $\Gamma=\{(x:\iota_1\to\iota_2),(y:\iota_1)\}$ gibt es jedoch eine Herleitung

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma \vdash x : \iota_1 \to \iota_2$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma \vdash y : \iota_1$}
\RightLabel{(Kon)}
\UnaryInfC{$\Gamma \vdash y : \iota_2 \to \iota_1$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma \vdash x : \iota_1 \to \iota_2$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma \vdash y : \iota_1$}
\RightLabel{(App)}
\BinaryInfC{$\Gamma \vdash xy : \iota_2$}
\RightLabel{(App)}
\BinaryInfC{$\Gamma \vdash y(xy) : \iota_1$}
\RightLabel{(App)}
\BinaryInfC{$\Gamma \vdash x(y(xy)) : \iota_2$}
\RightLabel{(Abs)}
\UnaryInfC{$x : \iota_1 \to \iota_2 \vdash \lambda y.x(y(xy)) : \iota_1 \to \iota_2$}
\RightLabel{(Abs)}
\UnaryInfC{$\vdash \lambda xy.x(y(xy)) : (\iota_1 \to \iota_2) \to \iota_1 \to \iota_2$}
\end{prooftree}
$$

für $\vdash \lambda xy.x(y(xy)) : (\iota_1 \to \iota_2) \to \iota_1 \to \iota_2$. Mit Satz 3.13 sieht man anhand des Erzeugendengraphen $\mathcal{E}_{\mathcal{J}}^{\sim}$

$$
\begin{tikzcd}
 & \iota_2 \to \iota_1 \arrow[dl, "1"'] & \\
\iota_1 \arrow[ur] & & \iota_2 \arrow[ul, "-1"']
\end{tikzcd}
$$

schnell ein, daß $\sim$ positiv ist.

Beispiel: P. Urzyczyn hat in [Urz95] gezeigt, daß der λ-Term $\overline{2}(\overline{2}k)$ mit $\overline{2} := \lambda yx.y(yx)$ und $K := \lambda xy.x$ weder in $\lambda_{\to}$ noch in System F typisierbar ist. Mit $\mathcal{F}$ und $\sim$ wie im vorhergehenden Beispiel und $\Gamma' := \{(x : \iota_1), (y : \iota_1 \to \iota_1)\}$ gibt es jedoch Herleitungen:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$x : \iota_1, y : \iota_2 \vdash x : \iota_1$}
\RightLabel{(Abs)}
\UnaryInfC{$x : \iota_1 \vdash \lambda y.x : \iota_2 \to \iota_1$}
\RightLabel{(Kon)}
\UnaryInfC{$x : \iota_1 \vdash \lambda y.x : \iota_1$}
\RightLabel{(Abs)}
\UnaryInfC{$\vdash K : \iota_1 \to \iota_1$}
\end{prooftree}
$$

und

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma' \vdash y : \iota_1 \to \iota_1$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma' \vdash y : \iota_1 \to \iota_1$}
\AxiomC{}
\RightLabel{(Var)}
\UnaryInfC{$\Gamma' \vdash x : \iota_1$}
\RightLabel{(App)}
\BinaryInfC{$\Gamma' \vdash yx : \iota_1$}
\RightLabel{(App)}
\BinaryInfC{$\Gamma' \vdash y(yx) : \iota_1$}
\RightLabel{(Abs)}
\UnaryInfC{$y : \iota_1 \to \iota_1 \vdash \lambda x.y(yx) : \iota_1 \to \iota_1$}
\RightLabel{(Abs)}
\UnaryInfC{$\vdash \overline{2} : (\iota_1 \to \iota_1) \to \iota_1 \to \iota_1$}
\end{prooftree}
$$

Zweifaches Anwenden der Regel (App) liefert schließlich $\vdash \overline{2}(\overline{2}K) : \iota_1 \to \iota_1$.

J. B. Wells konnte in [Wel94] zeigen, daß das Typisierbarkeitsproblem in System F unentscheidbar ist. Im Kalkül der positiven Typkongruenzen ist dies anders, zum Beweis dafür benötigen wir folgendes Lemma:

Lemma 3.16 *Seien t ein λ-Term, Γ bezüglich eines K-Typsystems $(\underline{\mathcal{F}}, \approx)$ eine Basis, die alle Variablen von t enthält und $\Gamma \vdash t : \sigma$ und $\Gamma \vdash t : \tau$ Urteile mit $\sigma, \tau \in \mathcal{F}$. Dann gilt $\sigma \approx \tau$.*

Beweis: Induktion über den Aufbau von t:

- $t = x \in X$:
  Nach dem Erzeugungslemma 2.11 gibt es ein $\rho \in \mathcal{F}$ mit $(x : \rho) \in \Gamma$, $\sigma \approx \rho$ und $\tau \approx \rho$. Daraus folgt $\sigma \approx \tau$.

- $t = t_1 t_2$:
  Nach dem Erzeugungslemma 2.11 gibt es $\sigma_1, \tau_1 \in \mathcal{F}$ mit $\Gamma \vdash t_1 : \sigma_1 \to \sigma$ und $\Gamma \vdash t_1 : \tau_1 \to \tau$. Nach Induktionsannahme gilt $\sigma_1 \to \sigma \approx \tau_1 \to \tau$, mit der Kürzungsregel (K) folgt $\sigma \approx \tau$.

- $t = \lambda x.t_1$:
  Nach dem Erzeugungslemma 2.11 gibt es $\sigma_1, \tau_1, \rho \in \mathcal{F}$ mit $(x : \rho) \in \Gamma$, $\Gamma \vdash t_1 : \sigma_1$, $\Gamma \vdash t_1 : \tau_1$, $\sigma \approx \rho \to \sigma_1$ und $\tau \approx \rho \to \tau_1$. Nach Induktionsannahme gilt $\sigma_1 \approx \tau_1$, daraus folgt $\sigma \approx \tau$.

$\square$

Korollar 3.17 *Es gibt ein Entscheidungsverfahren dafür, ob es zu einem gegebenen ungetypten λ-Term t ein von einer positiven Relation $\sim$ erzeugtes K-Typsystem $(\underline{\mathcal{F}}, \approx)$ gibt, mit dem t typisierbar ist.*

Beweis: Sei $\{x_i \,|\, i \in I\}$ die Menge aller in t enthaltenen Variablen, wobei o.B.d.A. keine Variable x_i in t mehrfach gebunden oder sowohl frei als auch gebunden ist. Setze $\mathcal{J} := \{\iota_i \,|\, i \in I\}$ und $\Gamma := \{(x_i : \iota_i) \,|\, i \in I\}$. Für jeden Teilterm s von t führe man ausgehend von denjenigen mit minimaler Länge folgendes Verfahren durch. Dabei heißt eine Typvariable **frisch**, falls sie nicht in $\mathcal{J}$ enthalten ist und während des Ablaufs des Verfahrens noch nicht benutzt wurde.

- Falls s eine Variable ist mit $(s : \iota) \in \Gamma$, setze $\sigma_s := \iota$.

- Falls s eine Applikation $s_1 s_2$ ist, wähle eine frische Variable κ, setze $\sigma_s := \kappa$ und $\sigma_{s_1} \sim \sigma_{s_2} \to \kappa$.

- Falls s eine Abstraktion $\lambda x.s_1$ ist mit $(x : \iota) \in \Gamma$, setze $\sigma_s := \iota \to \sigma_{s_1}$.

K sei die Menge der in obigen Verfahren frisch eingeführten Variablen. Nach Konstruktion gilt für jeden Teilterm s von t auch $\Gamma \vdash s : \sigma_s$, insbesondere ist t also bezüglich des Typsystems $(\underline{\mathcal{F}}_{\mathcal{J}\cup\mathcal{K}}, \approx)$ typisierbar. Sei $(\underline{\mathcal{F}}_{\mathcal{L}}, \approx'_{\mathcal{L}})$ ein K-Typsystem, Γ' eine Basis und $\tau' \in \mathcal{F}_{\mathcal{L}}$ ein Typ, so daß $\Gamma' \vdash t : \tau'$ ein Urteil ist. O.B.d.A. enthalte Γ' genau die in t vorkommenden Variablen. Dann definiert φ: $\underline{\mathcal{F}}_{\mathcal{J}\cup\mathcal{K}} \to \mathcal{F}_{\mathcal{L}}$ mit

$$\varphi(\tau) = \begin{cases} \rho & \text{falls } (x : \tau) \in \Gamma \text{ und } (x : \rho) \in \Gamma' \\ \rho & \text{falls } \tau = \sigma_{s_1 s_2} \text{ für ein beliebiges } \rho \in \mathcal{F}_{\mathcal{L}} \text{ mit } \Gamma' \vdash s_1 s_2 : \rho \\ \varphi(\tau_1) \to \varphi(\tau_2) & \text{falls } \tau = \tau_1 \to \tau_2 \end{cases}$$

einen Gruppoidhomomorphismus. Für jeden Teilterm s von t wird $\Gamma' \vdash s : \varphi(\sigma_s)$ mit Induktion über den Termaufbau von s gezeigt:

- $s = x \in X$: Wegen $(x : \sigma_x) \in \Gamma$ folgt $(x : \varphi(\sigma_x)) \in \Gamma'$ und somit $\Gamma' \vdash x : \varphi(\sigma_x)$.

- $s = s_1 s_1$: Nach Definition von φ gilt $\Gamma' \vdash s_1 s_2 : \varphi(\sigma_{s_1 s_2})$.

- $s = \lambda x.s_1$: Da x in Γ' enthalten ist, gilt nach Induktionsannahme $\Gamma' \vdash s_1 : \varphi(\sigma_{s_1})$ und $(x : \varphi(\sigma_x)) \in \Gamma'$. Daraus folgt $\Gamma' \vdash \lambda x.s_1 : \varphi(\sigma_x) \to \varphi(\sigma_{s_1})$, also mit $\sigma_s = \sigma_x \to \sigma_{s_1}$ und $\varphi(\sigma_x) \to \varphi(\sigma_{s_1}) = \varphi(\sigma_x \to \sigma_{s_1})$ auch $\Gamma \vdash \lambda x.s_1 : \varphi(\sigma_s)$.

Wir zeigen nun, daß aus $\sigma \sim \sigma'$ mit $\sigma, \sigma' \in \mathcal{F}_{\mathcal{J}\cup\mathcal{K}}$, auch $\varphi(\sigma) \approx' \varphi(\sigma')$ folgt. Nach Definition von $\sim$ gibt es einen Teilterm $s = s_1 s_2$ von t und ein $\kappa \in \mathcal{K}$ mit $\sigma_{s_1} = \sigma$, $\sigma' = \sigma_{s_2} \to \kappa$ und $\sigma_s = \kappa$. Wie oben gezeigt, gilt $\Gamma' \vdash s_1 : \varphi(\sigma)$, $\Gamma' \vdash s_2 : \varphi(\sigma_{s_2})$ und $\Gamma' \vdash s_1 s_2 : \varphi(\kappa)$. Nach dem Erzeugungslemma 2.11 gibt es ein $\rho \in \mathcal{F}_{\mathcal{L}}$ mit $\Gamma' \vdash s_1 : \rho \to \varphi(\kappa)$ und $\Gamma' \vdash s_2 : \rho$. Wegen Lemma 3.16 gilt daher $\varphi(\sigma) \approx' \rho \to \varphi(\kappa)$ und $\varphi(\sigma_{s_2}) \approx' \rho$, es folgt $\varphi(\sigma) \approx' \varphi(\sigma_{s_2}) \to \varphi(\kappa) = \varphi(\sigma_{s_2} \to \kappa) = \varphi(\sigma')$.

Sei $\smile$ die symmetrische, transitive K-Hülle von $\sim$, also die kleinste symmetrische und transitive Relation, die die Kürzungsregel (K) erfüllt. Nach Satz 2.23 ist $(\underline{\mathcal{F}}_{\mathcal{J}\cup\mathcal{K}}, \asymp)$ mit der von $\smile$ erzeugten Kongruenz $\asymp$ wieder ein K-Typsystem. Da auch $(\underline{\mathcal{F}}_{\mathcal{L}}, \asymp)$ ein K-Typsystem ist, folgt aus $\sigma \sim \sigma' \Rightarrow \varphi(\sigma) \approx' \varphi(\sigma')$ auch $\sigma \asymp \sigma' \Rightarrow \varphi(\sigma) \approx' \varphi(\sigma')$. Falls für $\sigma, \tau \in \mathcal{F}_{\mathcal{J}\cup\mathcal{K}}$ gilt, daß σ negativ in τ vorkommt, kommt auch $\varphi(\sigma)$ negativ in $\varphi(\tau)$ vor. Ist also $\asymp$ negativ, so auch die jedem anderen t typisierenden Typsystem zugrunde liegende Kongruenzrelation. Da mit $\sim$ auch $\smile$ endlich ist, ist nach Korollar 3.14 entscheidbar, ob $\asymp$ positiv oder negativ ist. Mit Satz 3.7 und Satz 3.9 folgt die Behauptung. $\square$

4 Zusammenfassung

In dieser Arbeit wird untersucht, welche Typsysteme Typtreue bewahren. Satz 2.17 besagt, daß die Bedingung (K) hierfür ein hinreichendes Kriterium ist, wobei offen bleibt, ob sie auch notwendig ist. In Kapitel 3 wird diskutiert, wann alle Terme eines mit einem K-Typsystem $(\underline{\mathcal{F}_\mathcal{J}}, \approx)$ getypten $\lambda_{(\underline{\mathcal{F}_\mathcal{J}}, \approx)}$-Kalküls stark normalisierend sind. Dies ist genau dann der Fall, wenn die Erzeugendenrelation $\sim$ positiv ist. Korollar 3.14 liefert ein graphentheoretisches notwendiges und hinreichendes Kriterium zum Nachweis der Positivität von endlichen Erzeugendenrelationen der Form $\{\iota_i \sim \tau_i \,|\, i = 1, \ldots, n,\ \iota_i \in \mathcal{J},\ \tau_i \in \mathcal{F}_\mathcal{J}\}$. Mit Algorithmus 2.27 ist ein Verfahren angegeben, das beliebige endlich repräsentierbare K-Typsysteme in Typsysteme der obigen Form übersetzt, so daß auch dann die Positivitätsbedingung nachgewiesen oder widerlegt werden kann. Dadurch wird eine Klasse von Typsystemen spezifiziert, mit denen neben den einfach typisierbaren λ-Termen noch weitere Terme typisiert werden können und dennoch starke Normalisierbarkeit garantiert ist. Nach Korollar 3.17 ist die Menge dieser Terme entscheidbar.

Es soll noch erwähnt werden, daß es auch andere Formen monomorpher Typsysteme gibt. Eng verwandt mit dem Begriff der Typkongruenzen und K-Typsystemen ist der der *rekursiven Typen*. Dabei verzichtet man auf Typkongruenzen und führt stattdessen *Typvariablen* und Ausdrücke der Form $\mu\alpha.\tau$ ein, wobei α eine Typvariable und τ ein Typ ist, in dem α nicht negativ vorkommt. Anstelle der Regel (Kon) gibt es zwei Regeln:

$$\frac{\Gamma \vdash t : \tau[\mu\alpha.\tau/\alpha]}{\Gamma \vdash t : \mu\alpha.\tau}\,(\text{Fold}) \qquad \frac{\Gamma \vdash t : \mu\alpha.\tau}{\Gamma \vdash t : \tau[\mu\alpha.\tau/\alpha]}\,(\text{Unfold})$$

P. Urzyczyn konnte in [Urz95] zeigen, daß dieser Kalkül im folgenden Sinn äquivalent zum Kalkül der Typkongruenzen ist: Ein ungetypter λ-Term t läßt sich mit rekursiven Typen genau dann typisieren, wenn es eine positive Relation $\sim$ auf einem freien Gruppoid $\underline{\mathcal{F}}$ gibt, so daß t bezüglich $(\underline{\mathcal{F}}, \approx)$ typisierbar ist.

Literatur

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, revised edition, 1984.

[Bar92] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 118–309. Clarendon Press, 1992.

[HS86] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and λ-Calculus.* Cambridge University Press, 1986.

[Ihr88] Th. Ihringer. *Allgemeine Algebra.* Teubner-Verlag, 1988.

[Men91] N. P. Mendler. Inductive types and type constraints in the second-order lambda-calculus. *Annals of Pure and Applied Logic*, 51:159–172, 1991.

[MS95] M. Marz and P. Selinger. A very well known theorem. *Semsoc Newsletter*, 4:7–8, 1995.

[Tak92] M. Takahashi. Parallel reductions in λ-calculus. Technical report, Tokyo Institute of Technology, 1992.

[Urz95] P. Urzyczyn. Positive recursive type assignment. Technical Report 95-03(203), Instytut Informatyki, Uniwersytet Warszawski, 1995.

[Wel94] J. B. Wells. Typability and type checking in the second order λ-calculus are equivalent and undecidable. In *9th LICS conference*, pages 176–185. IEEE Computer Society Press, 1994.