λ-Kalkül und Typen
1.Teil

Thorsten Altenkirch
email: `alti@informatik.uni-muenchen.de`

13. November 1996

1 Der einfach getypte λ-Kalkül ($\lambda^{\rightarrow}$)

Wir können λ-Terme und Typen auf verschiedenen Abstraktionsebenen betrachten:

1. Als *strings*, d.h. als Folge von Zeichen, die einen λ-Term darstellen.
2. Als Termbäume.
3. Als Kongruenzklassen von Termbäumen, die sich nur in gebundenen Variablen unterscheiden.

Die ersten beiden Stufen enthalten für unsere Zwecke zu viel unwesentliche Details. Das *parsing* von λ-Termen kann in einer Vorlesung über formale Sprachen oder Compilerbau behandelt werden. Wir werden jedoch λ-Terme als Strings notieren, und vereinbaren, daß diese für bestimmte Termbäume stehen.

Auch die zweite Stufe enthält unwesentliche Details. In einem λ-Term wie $\lambda x : o.x$ wird der Name x nur verwendet um eine Verbindung zu dem λ (den sogenannten Binder herzustellen). In dieser Sicht sollte es keinen Unterschied zwischen $\lambda x : o.x$ und $\lambda y : o.y$ geben.

Als λ-Terme wollen wir die Klassen der dritten Stufe bezeichnen. Wir führen jedoch zunächst Termbäume ein und definieren dann eine Äquivalenzrelation $\equiv$, die äquivalente Termbäume darstellen sollen. Diese Relation wird α-Äquivalenz genannt. Später werden wir eine Alternative kennenlernen, die diesen Umweg vermeidet (*Terme mit de-Bruijn-Indizes*,[dB72]).

Da die Typen des einfach getypten λ-Kalküls keine Variablen enthalten, gibt es hier keinen Unterschied zwischen den Stufen zwei und drei. Wir werden jedoch später Kalküle kennenlernen, in denen es auch Typvariablen gibt. Daher verwenden wir auch die Notation $\equiv$ um äquivalente, d.h. gleiche Typen zu kennzeichnen.

1.1 Typen und Termbäume

Wir führen zunächst eine Version des getypten λ-Kalküls mit einer minimalen Signatur ein, d.h. es gibt eine Typkonstante o und keine Termkonstanten. Später werden wir allgemeine Signaturen zulassen.

Grundtyp $o \in \mathrm{Ty}$

Funktionstyp Wenn $\sigma, \tau \in \mathrm{Ty}$ dann ist auch $\sigma \to \tau \in \mathrm{Ty}$.

Dies sind alle Typen.

Wir bemerken, daß Ausdrücke, die nicht vollständig geklammert sind, mehrdeutig sind. So kann $o \to o \to o$ als $(o \to o) \to o$ oder als $o \to (o \to o)$ gelesen werden.

Um Klammern zu sparen, vereinbaren wir, daß $\to$ rechtsassoziativ ist, also wird $o \to o \to o$ als $o \to (o \to o)$ interpretiert.

Wie bereits besprochen unterscheiden wir zwischen Termbäumen Tm_0 und Termen Tm. Wir definieren zunächst die Termbäume, dabei verwenden wir die (unendliche) Menge der Variablennamen V. Wir werden in Beispielen Variablen wie $x, y, z, x', y', z', \ldots$ verwenden.

Wir definieren die Menge der Termbäume Tm_0 induktiv:

Variablen $x \in \mathrm{Tm}_0$ für alle $x \in \mathrm{V}$.

Applikation Wenn $M, N \in \mathrm{Tm}_0$ dann ist auch $MN \in \mathrm{Tm}_0$.

Abstraktion Wenn $M \in \mathrm{Tm}_0, x \in V, \sigma \in \mathrm{Ty}$ dann ist auch $\lambda x : \sigma.M \in \mathrm{Tm}_0$.

Dies sind alle Termbäume.

Ähnlich wie schon bei den Typen, ist die lineare Notation von Termbäumen nicht immer eindeutig. Wir vereinbaren folgendes:

1. Applikation ist linksassoziativ, d.h. xyz wird wie $(xy)z$ interpretiert.
2. λ bindet schwächer als Applikation, d.h. $\lambda x : o \to o.xy$ wird wie $\lambda x : o \to o.(xy)$ interpretiert.

Aufgabe 1.1 *Mit dem folgenden Datentypen können wir in ML λ-Terme darstellen:*

```
datatype Ty = t_o | t_arr of Ty*Ty;
datatype Tm = var of string | lam of string*Ty*Tm | app of Tm*Tm;
```

1. *Implementiere Funktionen*

   ```
   parse_ty : string -> Ty;
   parse_tm : string -> Tm;
   ```

   in ML, die einen als string repräsentierten Typen, b.z.w. λ-term parsen. Dabei beachte folgende Konventionen:

   - $\to$ *wird als* `->` *dargestellt.*
   - λ *wird als* `\` *dargestellt.*
   - *Variablennamen sind Folgen von Buchstaben, Ziffern oder* `'`*, die mit einem Buchstaben anfangen.*
   - *Leerzeichen, Tabulatorzeichen und Zeilenwechsel sind Zwischenräume.*
   - *Klammern können verwendet werden.*

2. *Implementiere Funktionen:*

   ```
   printTy : Ty -> string;
   printTm : Tm -> string;
   ```

   die in ML repräsentierte Terme und Typen so ausgeben, daß die Anzahl der verwendeten Klammern minimal ist.

Wir sagen, eine Variable ist frei, wenn sie nicht durch λ gebunden wird. Präziser definieren wir die Menge der freien Variablen $\mathrm{FV}(M)$ eines Termbaums M wie folgt:

Definition 1.1 (Freie Variablen) *Wir definieren* $\mathrm{FV} \in \mathrm{Tm}_0 \to \mathcal{P}(\mathrm{V})$ *durch primitive Rekursion:*

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{x\} \\ \mathrm{FV}(MN) & = & \mathrm{FV}(M) \cup \mathrm{FV}(N) \\ \mathrm{FV}(\lambda x:\sigma.M) & = & \mathrm{FV}(M) \setminus \{x\} \end{array}$$

Wir definieren eine Substitutionsoperation auf Bäumen: Wenn M, N Termbäume sind und x eine Variable, dann ist $M\{x := N\}$ der Termbaum, bei dem alle freie Vorkommnisse von x in M durch N ersetzt worden sind.

Definition 1.2 (Substitution auf Bäumen) *Wir definieren* $-\{- := -\} \in \mathrm{Tm}_0 \times \mathrm{V} \times \mathrm{Tm}_0 \to \mathrm{Tm}_0$ *durch primitive Rekursion:*

$$\begin{array}{lcl} y\{x := N\} & = & \begin{cases} N & \textit{wenn } x = y \\ y & \textit{sonst} \end{cases} \\ (MM')\{x := N\} & = & (M\{x := N\})(M'\{x := N\}) \\ (\lambda y:\sigma.M)\{x := N\} & = & \begin{cases} \lambda y:\sigma.M & \textit{wenn } x = y \\ \lambda y:\sigma.(M\{x := N\}) & \textit{sonst} \end{cases} \end{array}$$

1.2 α-Kongruenz und Substitution auf Termen

Wir wollen jetzt präzise machen, daß Termbäume, die sich nur durch die Namen gebundener Variablen unterscheiden, dieselben Terme darstellen.

Zur Erinnerung: Gegeben zwei Mengen A, B eine Relation R ist eine Teilmenge des kartesischen Produktes $R \subseteq A \times B$. Wir schreiben aRb, wenn gilt $(a, b) \in R$. Sei $R \subseteq A \times A$ eine Relation, dann nennen wir R eine *Äquivalenzrelation*, wenn gilt:

- R ist *reflexiv*, d.h.
$$\forall_{x \in A} xRx$$
- R ist symmetrisch, d.h.
$$\forall_{x,y \in A} xRy \Rightarrow yRx$$
- R ist transitiv, d.h.
$$\forall_{x,y,z \in A} xRy \wedge yRz \Rightarrow xRz$$

Beispiel: Die Gleichheit $- = -$ ist eine Äquivalenzrelation.

Unter einer *Äquivalenzklasse* verstehen wir eine nichtleere Teilmenge $P \subseteq A$, so daß gilt:

- $\forall_{x,y \in P} xRy$
- $\forall_{x \in P, y \in A} xRy \Rightarrow y \in P$

Die Menge der Äquivalenzklassen A/R wird als *Quotientenmenge* bezeichnet. Zu jedem Element $a \in A$ können wir eine Äquivalenzklasse

$$[a] = \{x \in A | xRa\} \in A/R$$

zuordnen.

Wir definieren jetzt eine Äquivalenzrelation, die Termbäume, die sich nur in den Namen gebundener Variablen unterscheidet, identifiziert.

Definition 1.3 $\equiv \subseteq \mathrm{Tm}_0 \times \mathrm{Tm}_0$ *ist die kleinste Relation, so daß gilt:*

1. $x \equiv x$
2. $MN \equiv M'N'$ *wenn gilt* $M \equiv M'$ *und* $N \equiv N'$.

3. *$\lambda x : \sigma.M \equiv \lambda y : \sigma.N$ wenn $M\{x := z\} \equiv N\{y := z\}$, wobei z nicht in M oder N vorkommt.*

Lemma 1.4 *$\equiv$ ist eine Äquivalenzrelation.*

Beweis: Wir müssen zeigen, daß $\equiv$ reflexiv, symmetrisch und transitiv ist.

reflexiv Durch Induktion über die Struktur des Termbaums:

1. Für Variablen x gilt $x \equiv x$ durch 1.
2. Um zu zeigen, daß $MN \equiv MN$ können wir als Induktionsannahmen verwenden, daß $M \equiv M$ und $N \equiv N$. Die Behauptung folgt durch 2.
3. Um zu zeigen, daß $\lambda x : \sigma.M \equiv \lambda x : \sigma.M$ verwenden wir als Induktionsannahme, daß $M \equiv M$ ist. Es folgt, daß $M\{x := z\} \equiv M\{x := z\}$, da $-\{- := -\}$ eine totale Funktion ist. Also folgt mit 3: $\lambda x : \sigma.M \equiv \lambda x : \sigma.M$.

symmetrisch Wir nützen wir aus, daß $\equiv$ als die kleinste Relation definiert ist, die die Eigenschaften 1,2,3 hat, d.h. daß $M \equiv N$ genau dann gilt, wenn es eine Herleitung gibt, die nur 1,2,3 verwendet. Wir nehmen an $M \equiv N$ und wir wollen $N \equiv M$ durch Induktion über die Herleitung von $M \equiv N$ zeigen:

1. Wenn die letzte Regel, die angewendet wurde 1. war, haben wir $M = x$ und $N = x$, dann gilt mit 1. aber auch $N \equiv M$.
2. Wenn die letzte Regel, die angewendet wurde 2. war, haben wir $M = M_1 M_2, N = N_1 N_2$ und $M_1 \equiv M_2, N_1 \equiv N_2$. Als Induktionsannahmen können wir verwenden, daß Symmetrie schon für die Herleitungen von $M_1 \equiv N_1, M_2 \equiv N_2$ gilt, also haben wir $N_1 \equiv M_1, N_2 \equiv M_2$, wir können 2. anwenden und daher gilt $N_1 N_2 \equiv M_1 M_2$ also $N \equiv M$.
3. Der Fall, daß die letzte Regel, die angewendet wurde 3. war, läßt sich analog dem vorigen Fall behandeln.

transitiv Übung.
Hinweis: Wir setzen voraus, daß $M \equiv N$ und $N \equiv O$, der Beweis wird durch Induktion über die Struktur der Herleitung von $M \equiv N$ durch geführt. Dabei brauchen wir eine Hilfsinduktion (eigentlich nur eine Fallunterscheidung) über die Herleitung von $N \equiv O$.

□

Definition 1.5 (Terme) *Wir definieren die Menge der Terme (*Tm*) als die Menge der α-Äquivalenzklassen von Termbäumen, d.h.*

$$\mathrm{Tm} = \mathrm{Tm}_0 / \equiv$$

Wir notieren Terme durch Angabe eines Termbaumes, d.h. wir schrieben M anstelle von $[M]$, wobei $M \in \mathrm{Tm}_0$.

Wir wollen die Substitution jetzt auf Terme erweitern. Hier tritt jedoch ein Problem auf: die Substitutionsoperation auf Termbäumen ist nicht unter $\equiv$ abgeschlossen. Betrachten wir $M = \lambda x.y$ und $N = \lambda z.y$. Es ist leicht zu sehen, daß gilt $M \equiv N$. Aber $M\{y := z\} = \lambda x.z$ und $N\{y := z\} = \lambda z.z$ und $\lambda x.z \not\equiv \lambda z.z$!

Das Problem ist, daß alleine durch die Wahl eines Namens im zweiten Fall $N\{y := z\}$ eine neue Bindung erzeugt wird (*name capture*). Dies widerspricht der Intuition, daß Variablen nur Zeiger darstellen sollen. Wir lösen das Problem, indem wir eine Substitutionsoperation auf Termen definieren, die dies vermeidet (*capture-free substitution*).

Definition 1.6 (Substitution auf Termen) *Wir definieren die Operation* $-[- := -] \in \mathrm{Tm} \times \mathrm{V} \times \mathrm{Tm} \to \mathrm{Tm}$ *wie folgt. Seien* M, N *Termbäume, und* $M \equiv M'$ *wobei alle gebundenen Variablen in* M' *sich von den freien Variablen in* N *unterscheiden, dann ist das*

$$M[x := N] = M'\{x := N\}$$

d.h. das Ergebnis von $M[x := N]$ *ist die Äquivalenzklasse von* $M'\{x := N\}$*.*

Dies löst das oben beschriebene Problem, denn $M' = \lambda z.y \equiv \lambda z'.y$, was keine Variablen bindet, die in z frei sind. Daher ist:

$$\begin{aligned}(\lambda z.y)[y := z] &= (\lambda z'.y)\{y := z\} \\ &= \lambda z'.z\end{aligned}$$

Und:

$$\begin{aligned}(\lambda x.y)[y := z] &= (\lambda x.y)\{y := z\} \\ &= \lambda x.z\end{aligned}$$

In der Tat erhalten wir zwei äquivalente Repräsentanten

$$\lambda z'.z \equiv \lambda x.z$$

Wir können in der Praxis die Umbenennung vermeiden, indem wir der sogenannten Barendregt-Konvention folgen:

Konvention 1.7 (Barendregt) *Wir wählen die Termbäume, die Terme repräsentieren so, daß die Namen von gebundenen Variablen, sich von den Namen freier Variablen, die gleichzeitig diskutiert werden, unterscheiden.*

Lemma 1.8 (Substitutionslemmas)

1. $\dfrac{x \notin \mathrm{FV}(M)}{M[x := N] \equiv M}$
2. $M[x := x] \equiv M$
3. $\dfrac{x \neq y, x \notin \mathrm{FV}(L)}{M[x := N][y := L] \equiv M[y := L][x := N[y := L]]}$

Beweis: Übung
Hinweis: Durch Induktion über die Struktur der Termbäume. □

Aufgabe 1.2 *Implementiere* capture free substitution *für den Datentyp* `Tm` *aus Übung 1.1. D.h. implementiere einen Funktion:*

```
subst : Tm -> Var -> Tm -> Tm
```

die unter $\equiv$ *abgeschlossen ist. Dabei ist es sinnvoll die folgende Operation zu verwenden:*

```
fun variant x ys =
      let fun var x [] = x
            | var x (y::ys') =
                 if x=y then var (x^"'") ys
                 else var x ys'
      in var x ys
      end;
```

1.3 Typisierungsregeln

Wir definieren eine Relation $- : -$, die Terme und Typen in Beziehung setzt. Da wir auch Terme mit freien Variablen in Betracht ziehen müssen, führen wir Kontexte $\Gamma \in \mathrm{Con}$ als eine Folge von Paaren von Variablen und Typen ein, also z.B.

$$x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n$$

wobei alle Variablen verschieden sind, d.h. $x_i = x_j$ impliziert $i = j$.

Definition 1.9 *Wir definieren die Relation* $- \vdash - : - \subseteq \mathrm{Con} \times \mathrm{Tm} \times \mathrm{Ty}$ *als die kleinste Relation, die unter den folgenden Regeln abgeschlossen ist:*

$$\frac{}{\Gamma, x : \sigma, \Gamma' \vdash x : \sigma}\ \mathbf{var} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}\ \mathbf{lam}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \mathbf{app}$$

Es ist wichtig zu realisieren, daß die Regeln über Termen und nicht Termbäume definiert sind. Die Regel **lam** verlangt implizit, daß die Variable x noch nicht in Γ vorkommt. Es scheint, daß es nicht möglich ist, nachzuweisen, daß der Term $\lambda x : o.\lambda x : o.x$ den Typ $o \to o \to o$ hat. Die Ursache ist allerdings nur eine schlechte Wahl des Repräsentanten[1]. Der α-äquivalente Termbaum $\lambda x : o.\lambda y : o.y$ verursacht keine Probleme.

Als Beispiele für Herleitungen leiten wir die Typen für die *Kombinatoren* (geschlossene λ-Terme) I, K und S her. Dazu nehmen wir an, daß σ, τ, ρ beliebige Typen sind.

I

$$\dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\ \mathbf{var}}{\vdash \lambda x : \sigma . x : \sigma \to \sigma}\ \mathbf{lam}$$

K

$$\dfrac{\dfrac{\dfrac{}{x : \sigma, y : \tau \vdash x : \sigma}\ \mathbf{var}}{x : \sigma \vdash \lambda y : \tau . x : \tau \to \sigma}\ \mathbf{lam}}{\vdash \lambda x : \sigma . \lambda y : \tau . x : \sigma \to \tau \to \sigma}\ \mathbf{lam}$$

S

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_0 \vdash x : \sigma \to \tau \to \rho}\ \mathbf{var} \quad \dfrac{}{\Gamma_0 \vdash z : \sigma}\ \mathbf{var}}{\Gamma_0 \vdash xz : \tau \to \rho}\ \mathbf{app} \quad \dfrac{\dfrac{}{\Gamma_0 \vdash y : \sigma \to \tau}\ \mathbf{var} \quad \dfrac{}{\Gamma_0 \vdash z : \sigma}\ \mathbf{var}}{\Gamma_0 \vdash yz : \tau}\ \mathbf{app}}{\Gamma_0 \vdash (xz)yz : \rho}\ \mathbf{app}}{x : \sigma \to \tau \to \rho, y : \sigma \to \tau \vdash \lambda z : \sigma.(xz)yz : \sigma \to \rho}\ \mathbf{lam}}{x : \sigma \to \tau \to \rho \vdash \lambda y : \sigma \to \tau . \lambda z : \sigma.(xz)yz : (\sigma \to \tau) \to \sigma \to \rho}\ \mathbf{lam}}{\vdash \lambda x : \sigma \to \tau \to \rho . \lambda y : \sigma \to \tau . \lambda z : \sigma.(xz)yz : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho}$$

wobei $\Gamma_0 = x : \sigma \to \tau \to \rho, y : \sigma \to \tau, z : \sigma$.

[1] Bei der Konstruktion des Terms haben wir uns nicht an die Barendregt-Konvention gehalten.

Aufgabe 1.3 (Church Numerale I) *Wir können im einfach getypten λ-Kalkül natürliche Zahlen und einige Operationen auf Ihnen als sogenannte* Church Numerale *darstellen. Wir definieren $\overline{\mathrm{Nat}} = (o \to o) \to (o \to o)$. Zeige (durch Angabe von Herleitungsbäumen), daß die folgenden Typisierungen herleitbar sind:*

$$
\begin{array}{rcccl}
\bar{0} & = & \lambda f : o \to o.\lambda x : o.x & : & \overline{\mathrm{Nat}} \\
\bar{\mathrm{s}} & = & \lambda n : \overline{\mathrm{Nat}}.\lambda f : o \to o.\lambda x.f(nfx) & : & \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}} \\
\bar{+} & = & \lambda m : \overline{\mathrm{Nat}}.\lambda n : \overline{\mathrm{Nat}}.\lambda x : 0.mf(nfx) & : & \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}} \\
\bar{\times} & = & \lambda m : \overline{\mathrm{Nat}}.\lambda n : \overline{\mathrm{Nat}}.m(nf) & : & \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}}
\end{array}
$$

$$
\bar{+} \quad = \quad \lambda m : \overline{\mathrm{Nat}}.\lambda n : \overline{\mathrm{Nat}}.\lambda x : o.mf(nfx) \quad : \quad \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}} \to \overline{\mathrm{Nat}}
$$

In der Darstellung der λ-Terme mit Typen in den λ-Abstraktionen ist die Typisierung eindeutig:

Satz 1.10 (Eindeutigkeit der Typisierung) *Wenn $\Gamma \vdash M : \sigma$ und $\Gamma \vdash M : \sigma'$ dann ist $\sigma \equiv \sigma'$*

Beweis: Übung.
Hinweis: Der Beweis erfolgt durch Induktion über die Struktur der ersten Herleitung. Als Hilfssatz etablieren wir, daß sich alle Regeln umkehren lassen:

1. $\dfrac{\Gamma \vdash x : \sigma}{\Gamma = \Gamma_1, x : \sigma, \Gamma_2}$

2. $\dfrac{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}{\Gamma, x : \sigma \vdash M : \tau}$

3. $\dfrac{\Gamma \vdash MN : \tau}{\exists_{\sigma \in \mathrm{Ty}} \Gamma \vdash M : \sigma \to \tau \wedge \Gamma \vdash N : \sigma}$

□

Der Beweis dieses Lemma benötigt die Typannotationen der λ-Abstraktionen. Wenn wir die Regel für die Termbäume modifizieren:

Abstraktion Wenn $M \in \mathrm{Tm}'_0, x \in V$ dann ist auch $\lambda x.M \in \mathrm{Tm}'_0$.

und die entsprechende Typisierungsregel abändern

$$\frac{\Gamma, x : \sigma \vdash' M : \tau}{\Gamma \vdash' \lambda x.M : \sigma \to \tau}\,\mathbf{lam}'$$

erhalten wir ein System [2] für das Satz 1.10 nicht gilt. Z.B. haben wir

$$\vdash' \lambda x.x : o \to o$$

und

$$\vdash' \lambda x.x : (o \to o) \to (o \to o)$$

Aufgabe 1.4 *Implementiere Kontexte als*

```
type Con = (string*Ty) list
```

Zeige, daß die Typisierung entscheidbar ist, indem Du folgende Funktionen in L implementierst:

[2] Barendregt [Bar92] nennt ein derartiges System, ein *Curry* System und das System, das wir vorher eingeführt haben ein *Church-System*.

```
checkTy : Con -> Tm -> Ty -> bool
infTy : Con -> Tm -> (Ty option)
```

wobei `checkTy` $\Gamma\ M\ \sigma = \text{true}$ *und* `infTy` $\Gamma\ M\ = \texttt{SOME}\ \sigma$ *gdw.(genau dann wenn)* $\Gamma \vdash M : \sigma$.

Wir erwarten, daß die Typisierung eines Terms erhalten bleibt, wenn wir im Kontext

1. neue Annahmen hinzufügen,
2. die Reihenfolge von Typannahmen vertauschen,
3. unbenötigte Annahmen entfernen.

1.und 2. zeigen, daß wir Kontexte auch als Mengen von Annahmen behandeln können.

Lemma 1.11 (Abschwächen) *Wenn* $\Gamma, \Gamma' \vdash M : \tau$ *und* x *kommt nicht in* Γ, Γ' *vor, dann ist auch* $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$.

Beweis: Übung
Hinweis: Induktion über die Herleitung von $\Gamma, \Gamma' \vdash M : \tau$.
Zusatzfrage : Können wir den Spezialfall, wenn $\Gamma \vdash M : \tau$ dann $\Gamma, x : \sigma \vdash M : \tau$ direkt zeigen? □

Lemma 1.12 (Vertauschen) *Wenn* $\Gamma, x : \sigma, \Gamma', y : \sigma', \Gamma'' \vdash M : \tau$ *dann auch* $\Gamma, y : \sigma', \Gamma', x : \sigma, \Gamma'' \vdash M : \tau$.

Beweis: Übung
Hinweis: Induktion über die Herleitung von $\Gamma, x : \sigma, \Gamma', y : \sigma', \Gamma'' \vdash M : \tau$. □

Lemma 1.13 (Verstärken) *Wenn* $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$ *und* $x \notin \mathrm{FV}(M)$ *dann* $\Gamma, \Gamma' \vdash M : \tau$.

Beweis: Übung
Hinweis: Induktion über die Herleitung von $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$. □
Eine weitere wichtige Eigenschaft der Typisierungsrelation ist, daß sie unter Substitution abgeschlossen ist:

Lemma 1.14 (Substitution) *Wenn* $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$ *und* $\Gamma, \Gamma' \vdash N : \sigma$ *dann* $\Gamma, \Gamma' \vdash M[x := N] : \tau$.

Beweis: Übung
Hinweis: Induktion über die Herleitung von $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$. □

1.4 Gleichungstheorien

Gleichungstheorien sind Mengen wohlgetypter Gleichungen über λ-Termen. Wir werden eine Anzahl von Axiomen angeben, die jede Gleichungstheorie zu erfüllen hat. Im Unterschied zur universellen Algebra ist die *freie Theorie*, also gewissermaßen die minimale Theorie, die die Axiome erfüllt, nicht trivial. Wir nennen diese Theorie $\beta\eta$-Gleichheit und wir werden im nächsten Abschnitt nachweisen, daß diese Theorie entscheidbar ist. Wir werden auch Variationen betrachten, bei denen wir gewisse Axiome weglassen. Die jeweiligen freien Theorien (die β-Gleichheit und die schwache Gleichheit) sind von unabhängigen Interesse. Insbesondere werden wir später sehen, daß die schwache Gleichheit genau der Gleichheit der kombinatorischen Logik entspricht - einem variablenfreien Kalkül.

Eine Gleichung des λ-Kalküls besteht aus einem Kontext Γ, einem Typen σ und zwei Termen M, N so daß $\Gamma \vdash M, N : \sigma$. Wir notieren diese Gleichung als

$$\Gamma \rhd M = N : \sigma$$

Als erstes wollen wir verlangen, daß die Gleichheit für jedes Paar von Kontext und Typ eine Äquivalenzrelation darstellt. Wir formulieren die folgenden Axiome:

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \rhd M = M : \sigma}\,\mathbf{Refl} \qquad \frac{\Gamma \rhd M = N : \sigma}{\Gamma \rhd N = M : \sigma}\,\mathbf{Sym}$$

$$\frac{\Gamma \rhd M = N : \sigma \qquad \Gamma \rhd N = O : \sigma}{\Gamma \rhd M = O : \sigma}\,\mathbf{Trans}$$

Die Gleichungen sollen erhalten bleiben, wenn wir Variablen hinzufügen oder vertauschen und die Theorie soll unter Substitution abgeschlossen sein. Wir nehmen also die folgenden Axiome an:

$$\frac{\Gamma \rhd M = N : \sigma \qquad x \text{ erscheint nicht in } \Gamma}{\Gamma, x : \sigma \rhd M = N : \sigma}\,\mathbf{Weak}$$

$$\frac{\Gamma, x : \sigma, \Gamma', y : \sigma', \Gamma'' \rhd M = N : \sigma}{\Gamma, y : \sigma', \Gamma', x : \sigma, \Gamma'' \rhd M = N : \sigma}\,\mathbf{Swap}$$

$$\frac{\Gamma, x : \sigma, \Gamma' \rhd M = M' : \tau \qquad \Gamma, \Gamma' \rhd N = N' : \sigma}{\Gamma, \Gamma' \rhd M[x := N] = M'[x := N'] : \tau}\,\mathbf{Subst}$$

Durch die entsprechenden Lemmas 1.11, 1.12 und 1.14 wird sichergestellt, daß wohlgetypte Gleichungen in den Prämissen immer zu wohlgetypten Konklusionen führen.

Die Verstärkungsregel für Gleichungen:

$$\frac{\Gamma, x : \sigma, \Gamma' \rhd M = M' : \tau \qquad x \notin \mathrm{FV}(M) \cup \mathrm{FV}(M')}{\Gamma, \Gamma' \rhd M = M' : \tau}\,\mathbf{Str}$$

spielt eine besondere Rolle. Wir werden später sehen, daß die Verstärkungsregel die Interpretation von Typen durch leere Mengen ausschließt. Daher betrachten wir die Axiome mit und ohne die Verstärkungsregel.

Weitere Axiome legen fest, daß Applikation und λ-Abstraktion die Gleichheit erhalten:

$$\frac{\Gamma \rhd M = M' : \sigma \to \tau \qquad \Gamma \rhd N = N' : \sigma}{\Gamma \rhd MN = M'N' : \tau}\,\mathbf{App}$$

$$\frac{\Gamma, x : \sigma \rhd M = M' : \tau}{\Gamma \rhd \lambda x : \sigma.M = \lambda x : \sigma.M' : \sigma \to \tau}\,\xi$$

Für die schwache Gleichheit wird ξ nicht verlangt.

Das zentrale Axiom ist das β-Axiom:

$$\frac{\Gamma, x : \sigma \vdash M : \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \rhd (\lambda x : \sigma.M)N = M[x := N] : \tau}\,\beta$$

Das β-Axiom sagt, daß wir eine Funktionsanwendung ausrechnen können, indem wir die formalen Parameter der Funktionsabstraktion durch die aktuellen Argumente ersetzen.

Das letzte Axiom η ist etwas schwerer zu verstehen. Wie wir sehen werden, läßt sich aus ξ und η eine Extensionalitätsregel herleiten.

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad x \notin \mathrm{FV}(M)}{\Gamma \vdash \lambda x : \sigma . Mx = M : \sigma \to \tau}\,\eta$$

Die Axiome

$$\mathrm{Ax}_{\mathrm{W}} = \{\mathbf{Refl}, \mathbf{Sym}, \mathbf{Trans}, \mathbf{Weak}, \mathbf{Swap}, \mathbf{Subst}, \mathbf{App}, \beta\}$$

definieren die *schwache Gleichheit*; die Axiome

$$\mathrm{Ax}_{\beta} = \mathrm{Ax}_{\mathrm{W}} \cup \{\xi\}$$

die β-Gleichheit und

$$\mathrm{Ax}_{\beta\eta} = \mathrm{Ax}_{\beta} \cup \{\eta\}$$

die $\beta\eta$-Gleichheit.

Sei $\mathcal{E}$ eine Menge wohlgetypter Gleichungen, dann schreiben wir

$$\mathcal{E} \vdash_s \Gamma \triangleright M = N : \sigma$$

wenn sich $\Gamma \triangleright M = N : \sigma$ aus $\mathcal{E}$ mit den Axiomen aus Ax_s herleiten läßt (wobei $s \in \{\mathrm{W}, \beta, \beta\eta\}$). Wenn $\mathcal{E}$ leer ist, lassen wir es weg.

Wir sagen, daß sich eine Regel $\dfrac{E}{E'}$ in s herleiten läßt, wenn gilt $\{E\} \vdash_s E'$.

Das Axiom **Subst** wird häufig weggelassen, da es sich in Ax_β und daher auch in $\mathrm{Ax}_{\beta\eta}$ herleiten läßt:

Lemma 1.15 *In* $\mathrm{Ax}_\beta \setminus \{\mathbf{Subst}\}$ *läßt sich folgende Regel herleiten:*

$$\frac{\Gamma, x : \sigma \triangleright M = M' : \tau \qquad \Gamma \triangleright N = N' : \sigma}{\Gamma \triangleright M[x := N] = M'[x := N'] : \tau}$$

Beweis:

$$\cfrac{\cfrac{\cfrac{\Gamma, x : \sigma \triangleright M = M' : \tau}{\Gamma \triangleright \lambda x : \sigma . M = \lambda x : \sigma . M' : \sigma \to \tau}\,\xi \qquad \Gamma \triangleright N = N' : \sigma}{\Gamma \triangleright (\lambda x : \sigma . M)N = (\lambda x : \sigma . M')N' : \tau}\,\mathbf{App}}{\Gamma \triangleright M[x := N] = M'[x := N'] : \tau}\,\beta, \mathbf{Trans}$$

□

Da diese Herleitung die Regel ξ benötigt, läßt sie sich nicht für W durchführen. Aus diesem Grund führen wir **Subst** explizit als Axiom ein.

Aufgabe 1.5 (Church Numerale II) *Wir wollen jetzt nachweisen, daß die in Aufgabe 1.3 definierten λ-Terme bestimmte Gleichungen erfüllen. Gebe Herleitungen in $\mathrm{Ax}_{\beta\eta}$ für folgende Gleichheiten an:*

$$\begin{array}{rcccccc}
n : \overline{\mathrm{Nat}} & \triangleright & \bar{+}\bar{0}n & = & n & : & \overline{\mathrm{Nat}} \\
n : \overline{\mathrm{Nat}} & \triangleright & \bar{+}n\bar{0} & = & n & : & \overline{\mathrm{Nat}} \\
m, n : \overline{\mathrm{Nat}} & \triangleright & \bar{+}(\bar{s}m)n & = & \bar{s}(\bar{+}mn) & : & \overline{\mathrm{Nat}} \\
n : \overline{\mathrm{Nat}} & \triangleright & \bar{\times}\bar{0}n & = & \bar{0} & : & \overline{\mathrm{Nat}} \\
m, n : \overline{\mathrm{Nat}} & \triangleright & \bar{\times}(\bar{s}m)n & = & \bar{+}n(\bar{\times}mn) & : & \overline{\mathrm{Nat}}
\end{array}$$

1.5 Extensionalität

Wie bereits erwähnt, läßt sich aus den Regeln η und ξ ein Extensionalitätsprinzip herleiten. Wir betrachten zwei alternative Formulierungen:

$$\frac{\Gamma \vdash M, M' : \sigma \to \tau \qquad \forall_{\Gamma''=\Gamma,\Gamma' \in \mathrm{Con}, N \in \mathrm{Tm}} (\Gamma'' \vdash N : \sigma) \Rightarrow (\Gamma'' \rhd MN = M'N : \tau)}{\Gamma \rhd M = M' : \sigma \to \tau}\,\mathbf{ext}$$

$$\frac{\Gamma \vdash M, M' : \sigma \to \tau \qquad \Gamma, x : \sigma \rhd Mx = M'x : \tau \qquad \text{mit } x \notin \mathrm{FV}(M) \cup \mathrm{FV}(M')}{\Gamma \rhd M = M' : \sigma \to \tau}\,\mathbf{ext'}$$

Wir nennen zwei Axiomensysteme (regel-)äquivalent, wenn sich alle Axiome des einen in dem anderen herleiten lassen.

Satz 1.16 *Die folgenden Systeme sind regeläquivalent:*

1. $\mathrm{Ax}_{\beta\eta} = \mathrm{Ax}_{\mathrm{W}} \cup \{\xi, \eta\}$
2. $\mathrm{Ax}_{\mathrm{W}} \cup \{\mathbf{ext'}\}$
3. $\mathrm{Ax}_{\mathrm{W}} \cup \{\mathbf{ext}\}$

Beweis: Wir zeigen, daß $1 \vdash 2 \vdash 3 \vdash 1$:

$\{\xi, \eta\} \vdash_{\mathrm{W}} \mathbf{ext'}$

$$\frac{\dfrac{}{\Gamma \rhd M =}\,\eta \quad \dfrac{\Gamma, x : \sigma \rhd Mx = Nx : \tau}{\lambda x : \sigma.Mx = \lambda x : \sigma.Nx}\,\xi \quad \dfrac{}{= N : \sigma \to \tau}\,\eta, \mathbf{Sym}}{\Gamma \rhd M = N : \sigma \to \tau}\,\mathbf{Trans}$$

$\{\mathbf{ext'}\} \vdash_{\mathrm{W}} \mathbf{ext}$

$$\frac{\Gamma, x : \sigma \rhd Mx = M'x : \tau}{\Gamma \rhd M = M' : \sigma \to \tau}\,\mathbf{ext'} \text{ mit } \Gamma'' = \Gamma, x : \sigma; N = x$$

$\{\mathbf{ext}\} \vdash_{\mathrm{W}} \xi$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \rhd (\lambda x : \sigma.M)N =}\,\beta, \mathbf{Weak} \quad \dfrac{\Gamma, x : \sigma \rhd M = M' : \tau}{M[x := N] = M'[x := N]}\,\mathbf{Subst} \quad \dfrac{}{= (\lambda x : \sigma.M')N}\,\beta, \mathbf{Weak}}{\Gamma \rhd (\lambda x : \sigma.M)N = (\lambda x : \sigma.M')N}\,\mathbf{Trans}}{\Gamma \rhd \lambda x : \sigma.M = \lambda x : \sigma.M' : \sigma \to \tau}\,\mathbf{ext}$$

$\{\mathbf{ext}\} \vdash_{\mathrm{W}} \eta$

$$\frac{\dfrac{}{\Gamma \rhd MN = (Mx)[x := N]}\, x \notin \mathrm{FV}(M) \quad \dfrac{}{= (\lambda x : \sigma.M)N : \tau}\,\beta, \mathbf{Sym}}{\Gamma \rhd M = \lambda x : \sigma.M : \sigma \to \tau}\,\mathbf{ext}$$

□

Es ist wichtig zu bemerken, daß die Verwendung von verschiedenen Kontexten in **ext** absolut essentiell ist. Die folgende Regel ist nicht herleitbar:

$$\frac{\Gamma \vdash M, M' : \sigma \to \tau \qquad \forall_{N \in \mathrm{Tm}} \Gamma \vdash N : \sigma \Rightarrow \Gamma \rhd MN = M'N : \tau}{\Gamma \rhd M = M' : \sigma \to \tau}\,\omega$$

Wir werden später unter Verwendung der Normalisierung zeigen, daß diese Regel nicht gilt, d.h. die $\beta\eta$-Gleichheit ist ω-unvollständig.

1.6 Mengentheoretische Modelle

Die einfachste Semantik des einfach getypten λ-Kalkül erhält man, wenn man Typen als Mengen interpretiert und $\rightarrow$ als den mengentheoretischen Funktionsraum.

Zur Erinnerung: Gegeben Mengen A, B wir definieren die Menge der Funktionen von A nach B: $A \rightarrow B$ als die Menge der Relationen $R_f \subseteq A \times B$, die die folgenden Eigenschaften haben:

linkstotal $\forall_{x \in A} \exists_{y \in B} x R_f y$

rechtseindeutig $\forall_{x \in A, y, y' \in B} (x R_f y) \wedge (x R_f y') \Rightarrow y = y'$

Gegeben $x \in A$, schreiben wir $f(x)$ für das eindeutig bestimmte y, so daß $(x, y) \in R_f$. Sei $\phi(x)$ ein Term in B mit einer freien Variable aus A, dann beschreibt $x \in A \mapsto \phi(x)$ die Funktion $\{(x, \phi(x)) | x \in A\}$.

Wir werden später allgemeinere Modelle betrachten (Henkin-Modelle und kategorielle Modelle), aber diese einfache Semantik reicht schon aus, um nachzuweisen, daß das Axiomensystem $\mathrm{Ax}_{\beta\eta}$ konsistent ist, d.h. daß sich aus einer leeren Menge von Annahmen nicht jede wohlgetypte Gleichung herleiten läßt.

Eine mengentheoretisches Modell ist gegeben durch die Angabe einer Menge X. Wir definieren die Auswertung von Typen:

1. $[\![o]\!] = X$
2. $[\![\sigma \rightarrow \tau]\!] = [\![\sigma]\!] \rightarrow [\![\tau]\!]$

Wir interpretieren Kontexte

$$x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n$$

als eine Zuordnung ρ (genannt *Umgebung*), die jeder Variable x_i einen Wert $\rho(x_i) \in [\![\sigma_i]\!]$ zuordnet. Wir schreiben $\rho \in [\![\Gamma]\!]$. Mit $\rho, x = v$ notieren wir die Umgebung, die folgende Gleichung erfüllt:

$$(\rho, x = v)(y) = \begin{cases} v & \text{wenn } x = y \\ \rho(y) & \text{sonst} \end{cases}$$

Gegeben ein $\Gamma \vdash M : \sigma$ und eine Umgebung $\rho \in [\![\Gamma]\!]$, definieren wir die Auswertung von Typen $[\![M]\!]\rho \in [\![\sigma]\!]$ wie folgt:

$$\begin{aligned} [\![x]\!]\rho &= \rho(x) \\ [\![\lambda x : \sigma . M]\!]\rho &= v \in \sigma \mapsto [\![M]\!](\rho, x = v) \\ [\![MN]\!]\rho &= [\![M]\!]\rho([\![N]\!]\rho) \end{aligned}$$

Wir haben bei dieser Definition die Typkorrektheit $[\![M]\!]\rho \in [\![\sigma]\!]$ gleichzeitig bewiesen. Bei komplizierteren Systemen mit abhängigen Typen ist das in der Regel nicht möglich und wir müssen zunächst eine partielle Bewertung definieren (siehe [Str91]).

Wir definieren, daß eine Gleichung gültig in einem Modell ist

$$X \models \Gamma \rhd M = N : \sigma$$

wenn gilt, daß für alle $\rho \in [\![\Gamma]\!]$ gilt: $[\![M]\!]\rho = [\![N]\!]\rho$. Wir erweitern dies zu Theorien, indem wir sagen, daß $X \models \mathcal{E}$, wenn gilt, daß jede Gleichung aus $\mathcal{E}$ in X gültig ist.

Satz 1.17 (Korrektheit für Gleichungen) *Gegeben eine Menge X und*

$$\mathcal{E} \vdash \Gamma \rhd M = N : \sigma$$

Wenn $X \models \mathcal{E}$ dann $X \models \Gamma \rhd M = N : \sigma$.

Beweis: Übung

Hinweis: Der Beweis wird durch Induktion über die Struktur der Herleitung geführt. Der wesentliche Inhalt ist, daß die mengentheoretischen Funktionsräume, die Axiome erfüllen. □

Korollar 1.18 (Konsistenz) *Unter Konsistenz verstehen wir, daß es eine Gleichung*

$$\Gamma \rhd M = N : \sigma$$

gibt, die in der leeren Theorie nicht herleitbar ist. Wir zeigen, daß das System $\mathrm{Ax}_{\beta\eta}$ *konsistent ist.*

Beweis: Wir wählen $X = \{\mathrm{tt}, \mathrm{ff}\}$ (das ist die Menge der Wahrheitswerte). Nun betrachten wir

$$\vdash M = \lambda x : o.\lambda y : o.x : o \to o \to o$$

$$\vdash M' = \lambda x : o.\lambda y : o.y : o \to o \to o$$

Wenn wir annehmen

$$\rhd M = M' : o \to o \to o$$

dann muß wegen der Korrektheit gelten:

$$[\![M]\!] = [\![M']\!]$$

Es ist aber leicht zu sehen, daß die beiden Funktionen verschieden sind. □

Aufgabe 1.6 *Gebe die Werte von* $[\![M]\!], [\![M']\!]$ *im Beweis von Lemma 1.18 explizit an.*

Literatur

[Bar92] Henk P. Barendregt. Lambda calculi with types. In Samson Abramsky, Dov Gabbay, and Tom S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, chapter 2.2. Oxford University Press, 1992.

[dB72] N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indag. Math.*, 34, 1972.

[Str91] Thomas Streicher. *Semantics of Type Theory.* Birkhäuser, 1991.