Übungen zur "Theorie der Programmierung" - SoSe2015

Freitag 12:15-13:45, Martenstr. 3, Raum 02.133-113

Daniel Hausmann

daniel.hausmann@fau.de

Friedrich-Alexander-Universität Erlangen
Department Informatik

Lehrstuhl 8

June 29, 2015

System F (à la Curry)

Es seien V und $\underline{V}$ Mengen von Term- bzw. Typ-Variablen.

Typen, Terme

Typen: $\alpha, \beta ::= a \mid \alpha \to \beta \mid \forall a.\alpha \qquad a \in \underline{V}$

Terme: $s, t ::= x \mid s\ t \mid \lambda x.s \qquad x \in V$

Typisierungsregeln:

$$(\mathrm{Ax})\ \frac{}{\Gamma, x : \alpha \vdash x : \alpha}$$

$$(\to_i)\ \frac{\Gamma, x : \alpha \vdash s : \beta}{\Gamma \vdash \lambda x.s : \alpha \to \beta} \qquad (\to_e)\ \frac{\Gamma \vdash s : \alpha \to \beta \quad \Gamma \vdash t : \alpha}{\Gamma \vdash s\ t : \beta}$$

$$(\forall_i)\ \frac{\Gamma \vdash s : \alpha \qquad a \notin FV(\Gamma)}{\Gamma \vdash s : \forall a.\alpha} \qquad (\forall_e)\ \frac{\Gamma \vdash s : \forall a.\alpha}{\Gamma \vdash s : (\alpha[a := \beta])}$$

System F (à la **Curry**)

Seien s und t λ-Terme, Γ Kontext und α Typ.

Satz: (Subjekt-Reduktion / preservation)

Wenn $\Gamma \vdash s : \alpha$ und $s \to_\beta t$, dann $\Gamma \vdash t : \alpha$

Satz (Normalisierung - Girard 1972)

Wenn $\Gamma \vdash s : \alpha$, dann ist s stark normalisierend.

Übung 1 - Produkte in System F (à la Curry)

Wir kodieren das **kartesische Produkt** der Typen a und b in System F unter Verwendung von $(a \times b) := \forall r . (a \to b \to r) \to r$.

1. Wir definieren $pair = \lambda x\, y . (\lambda f . f\, x\, y)$. Zeigen Sie, dass in System F gilt: $\vdash pair : \forall a\, b . a \to b \to (a \times b)$.
2. Geben Sie zu jeder der folgenden Funktionssignaturen einen λ-Term an und zeigen Sie jeweils, dass er den benötigten Typ hat:
   1. $fst : \forall a\, b . (a \times b) \to a$
   2. $snd : \forall a\, b . (a \times b) \to b$
3. Schreiben Sie unter Verwendung der obigen Funktionen eine weitere Funktion $swap : \forall a\, b . (a \times b) \to (b \times a)$ und zeigen Sie, dass sie den korrekten Typ hat. Das heißt, finden Sie einen λ-Term s, so dass $\Gamma \vdash s : \forall ab.(a \times b) \to (b \times a)$ wobei

   $\Gamma := \{ pair : \forall a\, b.a \to b \to (a \times b), fst : \forall a\, b.(a \times b) \to a,$
   $snd : \forall a\, b.(a \times b) \to b\}$

Übung 2 - Koprodukte in System F (à la Curry)

Wir kodieren das Koprodukt der Typen a und b unter Verwendung von $(a + b) := \forall\, r.(a \to r) \to (b \to r) \to r$. Für diese Kodierung sind die "Konstruktions-Funktionen" *cLeft* und *cRight* wie folgt definiert:

$$\begin{aligned} cLeft &= \lambda x.\lambda f\, g.f\ x \\ cRight &= \lambda y.\lambda f\, g.g\ y \end{aligned}$$

1. Zeigen Sie, dass $\vdash cLeft : \forall ab.a \to (a + b)$ und $\vdash cRight : \forall ab.b \to (a + b)$.
2. Schreiben Sie eine Funktion des Typs $\forall\ a\ b\ c.((a \times b) + c) \to (a \to c) \to c$ und zeigen Sie, dass Ihre Funktion den verlangten Typ hat (d.h., geben Sie eine entsprechende Typinferenz an). Wie viele **unterschiedliche** Funktionen diesen Typs gibt es? ("unterschiedlich" im Sinne der berechneten Funktion, nicht "unterschiedlich" im Sinne der Struktur des λ-Terms!)

Übung 3 - Listen in System F (à la Curry)

Listen können in System F unter Verwendung des Typs
$\textbf{List}\ a := \forall r.\ r \to (a \to r \to r) \to r$ kodiert werden. In diesem Fall ergeben sich die folgenden "Konstruktor-Funktionen":

$$
\begin{aligned}
nil &= \lambda u\, f.u \\
cons &= \lambda x\, l.\lambda u\, f.\ f\ x\ (l\ u\ f)
\end{aligned}
$$

Für einen gegebenen Term t des Typs $\textbf{List}\ a$ verhält sich der Term $t\ u\ (\lambda\ x\ l.s)$ also wie eine Funktion f mit:

$$
\begin{aligned}
f\ nil &\to^*_{\beta\delta} u \\
f\ (cons\ x\ l) &\to^*_{\beta\delta} s[l \mapsto f\ l]
\end{aligned}
$$

1. Zeigen Sie, dass $\vdash nil : \forall a.\textbf{List}\ a$ und $\vdash cons : \forall a.a \to \textbf{List}\ a \to \textbf{List}\ a$.

Übung 3 - Listen in System F (à la Curry)

2. Schreiben Sie eine Funktion *length*, welche die Länge einer Liste berechnet. Das heißt:

$$\begin{aligned} \mathit{length}\ \ \mathit{nil} & \to^*_{\beta\delta} \mathit{zero} \\ \mathit{length}\ \ (\mathit{cons}\ x\ \ l) & \to^*_{\beta\delta} \mathit{succ}\ (\mathit{length}\ l) \end{aligned}$$

   Zeigen Sie, dass $\Gamma_0 \vdash \mathit{length} : \forall a.\mathbf{List}\ \ a \to \mathbb{N}$, wobei $\Gamma_0 = \{\mathit{zero} : \mathbb{N}, \mathit{succ} : \mathbb{N} \to \mathbb{N}\}$.

3. Schreiben Sie eine Funktion zur Listenkonkatention. Das heißt, schreiben Sie eine Funkton *append*, so dass:

$$\begin{aligned} \mathit{append}\ \ \mathit{nil}\ \ r & \to^*_{\beta\delta} r \\ \mathit{append}\ (\mathit{cons}\ x\ \ l)\ \ r & \to^*_{\beta\delta} \mathit{cons}\ x\ (\mathit{append}\ l) \end{aligned}$$

   Zeigen Sie außerdem, dass $\Gamma_0 \vdash \mathit{append} : \forall a.\mathbf{List}\ \ a \to \mathbf{List}\ \ a \to \mathbf{List}\ \ a$, wobei:

$$\Gamma_0 = \{\ \mathit{nil}\ : \forall a.\mathbf{List}\ \ a, \mathit{cons} : \forall a.a \to \mathbf{List}\ \ a \to \mathbf{List}\ \ a\}$$

Übung 3 - Listen in System F (à la Curry)

4. Schreiben Sie eine Funktion *join*, die eine Liste von Listen als Argument erwartet und die durch (von links nach rechts erfolgende) Konkatenation der einzelnen Listen entstehende Gesamtliste berechnet. Das heißt, für Ihre Funktion soll gelten:

$$\begin{array}{lll} join\ \ nil & \to^*_{\beta\delta} & nil \\ join\ \ (Cons\ x\ l) & \to^*_{\beta\delta} & append\ x\ (join\ l) \end{array}$$

Zeigen Sie, dass $\Gamma_0 \vdash join : \forall a.\mathbf{List}\ (\mathbf{List}\ a) \to \mathbf{List}\ a$, wobei

$$\Gamma_0 = \{ nil : \forall a.\mathbf{List}\ a, append : \forall a.\mathbf{List}\ a \to \mathbf{List}\ a \to \mathbf{List}\ a\}$$

System F (**à la Church**) (λ_2-Church)

Terme

Terme: $s, t ::= x \mid s\ t \mid \lambda x : \alpha.s \mid \Lambda a.s \mid s\ \alpha \qquad\qquad x \in V$

β-Reduktion im System F à la Church

$(\lambda x : \alpha.s)\ t \to_\beta s[x \mapsto t]$, $(\Lambda a.t)\alpha \to_\beta t[a \mapsto \alpha]$

Typisierungsregeln:

$$(\mathrm{Ax})\ \frac{}{\Gamma, x : \alpha \vdash x : \alpha}$$

$$(\to_i)\ \frac{\Gamma, x : \alpha \vdash s : \beta}{\Gamma \vdash \lambda x : \alpha.s : \alpha \to \beta} \qquad (\to_e)\ \frac{\Gamma \vdash s : \alpha \to \beta \quad \Gamma \vdash t : \alpha}{\Gamma \vdash s\ t : \beta}$$

$$(\forall_i)\ \frac{\Gamma \vdash s : \alpha \quad a \notin FV(\Gamma)}{\Gamma \vdash \Lambda a.s : \forall a.\alpha} \qquad (\forall_e)\ \frac{\Gamma \vdash s : \forall a.\alpha}{\Gamma \vdash s\ \beta : (\alpha[a := \beta])}$$

Übung 4 - System F à la Church

1. Definieren Sie die Konstruktor-Funktionen *nil* und *cons* aus der vorherigen Übung als äquivalente Terme des jeweils gleichen Typs in System F à la Church.
2. Implementieren Sie die Funktionen *length*, *append* und *join* als äquivalente Terme des jeweils gleichen Typs in System F à la Church.