29.11.05

Theorie der Programmierung I

(Mitschrift von Lars Friedrich email@lars-friedrich-home.de)

$\alpha_0 = (\alpha_3 \to \alpha_3) \to (\alpha_3 \to \alpha_3)$

Also: Jeder mögliche Typ von $\lambda f.\ \lambda x.\ f(fx)$ ist von der Form $(\tau \to \tau) \to (\tau \to \tau)$

Umgekehrt: Jeder Typ dieser Form kommt als Typ des Ausdrucks in Frage (z.B. durch Typherleitung zeigen).

Jetzt: Systematisches Verfahren zum Aufstellen und Lösen dieser Typgleichungen

Zunächst: Aufstellen der Gleichungen

Vorgegeben: Typumgebung Γ; Ausdruck e (der Sprache L_2, d.h. **ohne** Typinformation); Typvariable α

(**Beachte:** Da wir die Menge Type aller Typen erweitert haben, gibt es jetzt auch neuartige Typumgebungen Γ, nämlich: Γ (id) darf Typvariablen enthalten)

Definition: Die Menge tequs (Γ, e, α) wird durch Induktion über die Größe von e definiert durch:

- tequs $(\Gamma, e, \alpha) = \{ \alpha = \tau \}$
- tequs $(\Gamma, id, \alpha) = \begin{cases} \alpha = \Gamma(id), \text{ falls } id \in dom(\Gamma) \\ \text{„Fehler“, falls } id \notin dom(\Gamma) \end{cases}$
- tequs $(\Gamma, e_1e_2, \alpha) =$ tequs $(\Gamma, e_1, \alpha_1) \cup$ tequs $(\Gamma, e_2, \alpha_2) \cup \{ \alpha_1 = \alpha_2 \to \alpha \}$
  (wobei α_1, α_2 neue Typvariablen sind)
- tequs $(\Gamma, \lambda id.\ e, \alpha) =$ tequs $(\Gamma\ [\alpha_0/id], e, \alpha_1) \cup \{ \alpha = \alpha_1 \to \alpha_0 \}$
  (wobei α_0, α_1 neue Typvariablen sind)
- tequs $(\Gamma, rec\ id\ e, \alpha) =$ tequs $(\Gamma\ [\alpha_0/id], e, \alpha_1) \cup \{ \alpha = \alpha_0, \alpha = \alpha_1 \}$
  (wobei α_0, α_1 neue Typvariablen sind)
- tequs $(\Gamma, \text{if } e_0 \text{ then } e_1 \text{ else } e_2, \alpha) =$ tequs $(\Gamma, e_0, \alpha_0) \cup$ tequs $(\Gamma, e_1, \alpha_1) \cup$ tequs $(\Gamma, e_2, \alpha_2) \cup \{ \alpha_0 = bool, \alpha = \alpha_1, \alpha = \alpha_2 \}$
  (wobei $\alpha_0, \alpha_1, \alpha_2$ neue Typvariablen sind)

(let-Ausdrücke bei **diesem** Algorithmus nicht möglich) **) Nachtrag am 06.12.05*

Beispiel:

tequs([], $\lambda f.\ \lambda x.\ f(fx)$ α_0) =

tequs([f: α_1], $\lambda x.\ f(fx)$, α_2) $\cup$ { $\alpha_0 = \alpha_1 \to \alpha_2$}

└→tequs([f: α_1, x: α_3], f(fx), α_4) $\cup$ { $\alpha_2 = \alpha_3 \to \alpha_4$}

  └→tequs([f: α_1, x: α_3], f, α_5) $\cup$ teques([f: α_1, x: α_3], fx, α_6) $\cup$ { $\alpha_5 = \alpha_6 \to \alpha_4$}

    └→{ $\alpha_1 = \alpha_5$}   └→tequs([f: α_1, x: α_3], f, α_7) $\cup$ teques([f: α_1, x: α_3], x, α_3) $\cup$ { $\alpha_7 = \alpha_8 \to \alpha_6$}

      └→{ $\alpha_1 = \alpha_7$}   └→{ $\alpha_3 = \alpha_8$}

Wie löst man eine Gleichungsmenge? Was versteht man unter einer Lösung?

Definition: **(a)** Eine **(Typ-)Substitution** ist eine Funktion s = TVar, für die gilt: Es gibt nur endlich viele $\alpha \in$ TVar mit $s(\alpha) \neq \alpha$.

Schreibweise: $[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n]$ bezeichnet die Substitution s mit

- $s(\alpha_i) = \tau_i$ für i = 1, …, n
- $s(\alpha) = \alpha$ für alle $\alpha \notin \{\alpha_1, \ldots, \alpha_n\}$

(b) Die Anwendung einer Substitution s auf einen Typ τ liefert einen Typ τs, der so definiert ist:

- $\beta s = \beta$ falls $\beta \in$ {int, bool, unit}
- $\alpha s = s(\alpha)$
- $\tau_1 \rightarrow \tau_2 = \tau_1 s \rightarrow \tau_2 s$

In Worten: τs ergibt sich aus τ, indem man jede Typvariable α durch ihr Bild $s(\alpha)$ ersetzt, und zwar alle Typvariablen „gleichzeitig“ z.B.:

$$\left.\begin{array}{l}\tau=\alpha_1\rightarrow\alpha_2\\ s=[\alpha_2\rightarrow\alpha_2/\alpha_1, int/\alpha_2]\end{array}\right\} \tau s=(\alpha_2\rightarrow\alpha_2)\rightarrow int$$

(c) Die Komposition zweier Substitutionen τ_1 und τ_2 ist die Substitution s_1s_2, die so definiert ist: $\underbrace{(s_1s_2)}_{\mid \alpha s_1s_2}(\alpha) = \underbrace{(s_1(\alpha))}_{\mid \in \text{Type}}s_2 = (\alpha s_1)s_2$

Bsp.: $s_1 = [\alpha_2\rightarrow\alpha_2/\alpha_1]$, $s_2 = [int/\alpha_2]$

Dann gilt: $(s_1s_2)(\alpha_1) = (s_1\alpha_1)s_2 = (\alpha_2\rightarrow\alpha_2)s_2 = int\rightarrow int$

$(s_1s_2)(\alpha_2) = (s_1\alpha_2)s_2 = \alpha_2 s_2 = int$

$(s_1s_2)(\alpha) = (s_1\alpha)s_2 = \alpha s_2 = \alpha$ für alle $\alpha \notin \{\alpha_1, \alpha_2\}$

Also: $s_1s_2 = [int\rightarrow int/\alpha_1, int/\alpha_2]$

Bemerkung: Die Komposition ist gerade so definiert, dass für alle Typen τ gilt:
$\tau(s_1s_2) = (\tau s_1)s_2$

Definition: (a) eine Substitution s heißt **Lösung** der Gleichung $\tau_1=\tau_2$, wenn $\tau_1 s = \tau_2 s$ (syntaktisch gleich)

(b) s heißt Lösung einer Gleichungsmenge, wenn s Lösung jedes Elements dieser Menge.

(Alternativer Begriff: Eine Lösung s einer Gleichung $\tau_1=\tau_2$ wird auch **Unifikator** von $\{\tau_1\tau_2\}$ genannt. Man sagt dann, s unifiziert die Menge $\{\tau_1\tau_2\}$.)

Beispiel:
- Die Gleichung $\alpha = \alpha \rightarrow int$ hat <u>keine</u> Lösung, da es kein $\tau \in$ Type gibt mit $\tau = \tau \rightarrow int$ (syntaktisch gleich)
- Die Gleichungsmenge $\{\alpha = \alpha \rightarrow int, \alpha = int \rightarrow \alpha\}$ hat keine Lösung.
- $\{\alpha \rightarrow \alpha' = \alpha' \rightarrow \alpha\}$ hat unendlich viele Lösungen, nämlich alle Substitutionen s mit $s(\alpha) = s(\alpha')$

Definition: Seien s, s' Substitutionen. s heißt **allgemeiner** als s', wenn es eine Substitution s'' gibt mit $s' \subset ss''$

Beispiel: $s = [\alpha/\alpha']$, $s' = [int/\alpha, int/\alpha']$
s ist allgemeiner als s', denn S' = ss'' mit $s''= [int/\alpha]$
sei $s_1= [\alpha_1/\alpha']$, dann ist s allgemeiner als s_1 <u>und</u> s_1 allgemeiner als s.

Definition: Eine Substitution s heißt **allgemeinste Lösung** einer Gleichung(smenge), wenn gilt:
- s ist Lösung
- wenn s' Lösung dieser Gleichung ist, dann muss $s'(\alpha) = \underbrace{s'(\alpha')}_{\mid \tau}$ sein, also gilt
  $s' = ss''$ mit s'': TVar $\rightarrow$ Type, $\alpha\rightarrow\tau$, $\alpha''\rightarrow s'(\alpha'')$ für alle $\alpha''\in \{\alpha, \alpha'\}$