

Typinferenz für die Sprachen \mathcal{L}_i^{ML}

Vorbereitungen

Um Konstanten und Namen einheitlich behandeln zu können, setzen wir voraus, dass die Axiome für Konstanten jetzt von der Form $c :: \pi$ sind. Für die ‘alten’ Konstanten (aus \mathcal{L}_1^{ML}) sei π einfach der bisherige Typ τ , für die ‘neuen’ Konstanten (aus \mathcal{L}_3^{ML}) sei es jeweils der ‘richtige’ polymorphe Typ, d.h.

$$\begin{aligned}
 [] &:: \forall \alpha. \alpha \mathbf{list} \\
 cons &:: \forall \alpha. \alpha * \alpha \mathbf{list} \rightarrow \alpha \mathbf{list} \\
 hd &:: \forall \alpha. \alpha \mathbf{list} \rightarrow \alpha \\
 tl &:: \forall \alpha. \alpha \mathbf{list} \rightarrow \alpha \mathbf{list} \\
 is_empty &:: \forall \alpha. \alpha \mathbf{list} \rightarrow \mathbf{bool} \\
 fst &:: \forall \alpha_1, \alpha_2. \alpha_1 * \alpha_2 \rightarrow \alpha_1 \\
 snd &:: \forall \alpha_1, \alpha_2. \alpha_1 * \alpha_2 \rightarrow \alpha_2
 \end{aligned}$$

Dann muss natürlich die Regel (CONST) abgeändert werden zu

$$(P\text{-CONST}) \frac{c :: \pi}{\Gamma \triangleright c :: \tau} \text{ falls } \tau \text{ Instanz von } \pi$$

Zur Formulierung des Algorithmus benötigen wir dann den folgenden Begriff:

τ' heißt *neue Instanz* des polymorphen Typs $\pi = \forall \alpha_1, \dots, \alpha_n. \tau$ wenn τ' von der Form $\tau[\alpha'_1/\alpha_1, \dots, \alpha'_n/\alpha_n]$ ist, wobei $\alpha'_1, \dots, \alpha'_n$ neue Typvariablen sind.

Außerdem muss die Definition von Γs angepasst werden, weil Γ jetzt *polymorphe* Typen liefert: Für einen polymorphen Typ $\pi = \forall \alpha_1, \dots, \alpha_n. \tau$ sei

$$\pi s = \forall \alpha_1, \dots, \alpha_n. \tau (s|_{free(\pi)})$$

d.h. bei der Anwendung einer Substitution s auf den polymorphen Typ $\pi = \forall \alpha_1, \dots, \alpha_n. \tau$ werden in τ nur die Typvariablen ersetzt, die *nicht* durch einen \forall -Quantor gebunden sind, z.B.

$$(\forall \alpha_1. \alpha_2 \rightarrow \alpha_1) [\mathbf{int}/\alpha_1, \mathbf{int}/\alpha_2] = \forall \alpha_1. \mathbf{int} \rightarrow \alpha_1$$

Γs ist dann (wieder) definiert durch $dom(\Gamma s) = dom(\Gamma)$ und $\Gamma s(id) = \Gamma(id)s$ für alle $id \in dom(\Gamma)$.

Der Typinferenz-Algorithmus

... wird in Form einer Funktion $solution(\Gamma, e, \tau)$ dargestellt, die durch Induktion über die Größe von e definiert ist.

- (a) $solution(\Gamma, c, \tau) = unify(\{\}\tau = \tau')$
falls $c :: \pi$ und τ' neue Instanz von π
- (b) $solution(\Gamma, id, \tau) = unify(\{\}\tau = \tau')$
falls $id \in dom(\Gamma)$ und τ' neue Instanz von $\Gamma(id)$
- (c) $solution(\Gamma, e_1 e_2, \tau) = s_1 s_2$ wobei
 - $s_1 = solution(\Gamma, e_1, \alpha \rightarrow \tau)$ mit einer neuen Typvariablen α
 - $s_2 = solution(\Gamma s_1, e_2, \alpha s_1)$
- (d) $solution(\Gamma, \mathbf{if} e_0 \mathbf{then} e_1 \mathbf{else} e_2, \tau) = s_0 s_1 s_2$ wobei
 - $s_0 = solution(\Gamma, e_0, \mathbf{bool})$
 - $s_1 = solution(\Gamma s_0, e_1, \tau s_0)$
 - $s_2 = solution(\Gamma s_0 s_1, e_2, \tau s_0 s_1)$
- (e) $solution(\Gamma, (e_1, e_2), \tau) = s_1 s_2 s_3$ wobei
 - $s_1 = solution(\Gamma, e_1, \alpha_1)$ mit einer neuen Typvariablen α_1
 - $s_2 = solution(\Gamma s_1, e_2, \alpha_2 s_1)$ mit einer neuen Typvariablen α_2
 - $s_3 = unify(\{\}\tau s_1 s_2 = \alpha_1 s_1 s_2 * \alpha_2 s_1 s_2)$
- (f) $solution(\Gamma, \lambda id. e, \tau) = s_1 s_2$ wobei
 - $s_1 = solution(\Gamma[\alpha_1/id], e, \alpha_2)$ mit neuen Typvariablen α_1, α_2
 - $s_2 = unify(\{\}\tau s_1 = \alpha_1 s_1 \rightarrow \alpha_2 s_1)$
- (g) $solution(\Gamma, \mathbf{rec} id. e, \tau) = solution(\Gamma[\tau/id], e, \tau)$
- (h) $solution(\Gamma, \mathbf{let} id = e_1 \mathbf{in} e_2, \tau) = s_1 s_2$ wobei
 - $s_1 = solution(\Gamma, e_1, \alpha)$ mit einer neuen Typvariablen α
 - $s_2 = solution(\Gamma s_1[Closure_{\Gamma s_1}(\alpha s_1)/id], e_2, \tau s_1)$