Types 1999 p7 q12 PMS

Solution to Q1

Implicit typing reduces the amount of routine verbiage in programs; explicit typing provides useful documentation and is necessary for the interfaces of module systems. Implicit typing requires a usable type inference algorithm and so cannot be used for very expressive type systems.

The judgement is $\Gamma \vdash M : \tau$ where Γ is a pair of a set of type variables and an assignment of type schemes to identifiers.

$$\Gamma, x : \sigma \vdash x : \tau \quad \text{if well-formed and } \sigma \succ \tau \qquad (\text{var} \succ)$$

$$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \mathbf{fn}\, x \Rightarrow M : \tau_1 \to \tau_2} \qquad (\text{fn})$$

$$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\, M_2 : \tau_2} \qquad (\text{app})$$

$$\frac{\begin{array}{c} A, \Gamma \vdash M_1 : \tau_1 \\ \Gamma, x : \forall A\,(\tau_1) \vdash M_2 : \tau_2 \end{array}}{\Gamma \vdash \mathbf{let\,val}\, x = M_1\, \mathbf{in}\, M_2\, \mathbf{end} : \tau_2} \qquad (\text{let})$$

Take

$$\begin{array}{rcl} M_1 & = & \mathbf{fn}\, x \Rightarrow x \\ M_2 & = & (f\, \mathbf{true}) :: (f\, \mathbf{nil}) \end{array}$$

Use

$$\begin{array}{rcl} \forall\, \alpha\, (\alpha \to \alpha) & \succ & \mathtt{bool} \to \mathtt{bool} \\ \forall\, \alpha\, (\alpha \to \alpha) & \succ & \mathtt{bool\,list} \to \mathtt{bool\,list} \end{array}$$

A proof of $\emptyset, \emptyset \vdash N_2 : \tau$ would have to be of the form

$$\cfrac{\cfrac{\cfrac{\nabla_1 \quad \nabla_2}{\emptyset, f : \tau_1 \vdash M_2 : \tau}\,(\text{cons})}{\emptyset, \emptyset \vdash \mathbf{fn}\, f \Rightarrow M_2 : \tau_1 \to \tau}\,(\text{fn}) \qquad \cfrac{\cdots}{\emptyset, \emptyset \vdash M_1 : \tau_1}\,(\text{fn})}{\emptyset, \emptyset \vdash N_2 : \tau}\,(\text{app})$$

where ∇_1 is

$$\cfrac{\cfrac{}{\emptyset, f : \tau_1 \vdash f : \mathtt{bool} \to \tau_2}\,\text{var} \qquad \cfrac{}{\emptyset, f : \tau_1 \vdash \mathbf{true} : \mathtt{bool}}\,\text{bool}}{\emptyset, f : \tau_1 \vdash f\, \mathbf{true} : \tau_2}\,\text{app}$$

and ∇_2 is

$$\cfrac{\cfrac{}{\emptyset, f : \tau_1 \vdash f : \tau_3\, \mathtt{list} \to \tau_2\, \mathtt{list}}\,\text{var} \qquad \cfrac{}{\emptyset, f : \tau_1 \vdash \mathbf{nil} : \tau_3\, \mathtt{list}}\,\text{nil}}{\emptyset, f : \tau_1 \vdash f\, \mathbf{nil} : \tau_2\, \mathtt{list}}\,\text{app}$$

and

$$\begin{array}{rcl} \tau & = & \tau_2\,\texttt{list} \\ \tau_1 & \succ & \texttt{bool} \rightarrow \tau_2 \\ \tau_1 & \succ & \tau_3\,\texttt{list} \rightarrow \tau_2\,\texttt{list} \end{array}$$

but this leads to a contradiction, as by well-formedness τ_1 contains no free type variables so there is no τ_1 satisfying the two generalisations.

A closed type scheme σ is principal for a closed term M if if

(a) $\vdash_{ML} M : \sigma$

(b) for all closed σ', if $\vdash_{ML} M : \sigma'$ then $\sigma \succ \sigma'$

$\texttt{bool list}$ is the principal type scheme for N_1.