Abstract syntax for maybe types and lists (LLMH)

Types:

$$Type \ ::= \ \ldots \ \mid \ \texttt{Maybe}\ Type \ \mid \ [\,Type\,]$$

Expressions:

$$\begin{aligned} Exp \ ::= \ & \ldots \quad \text{(all existing constructs)} \\ & \mid \ \texttt{Nothing} \ \mid \ \texttt{Just} \\ & \mid \ \texttt{case}\ Exp\ \texttt{of}\ \texttt{Nothing} \to Exp ; \texttt{Just}\ Var \to Exp \\ & \mid \ [\,] \ \mid \ Exp : Exp \\ & \mid \ \texttt{case}\ Exp\ \texttt{of}\ [\,] \to Exp ; Var{:}Var \to Exp \end{aligned}$$

New typing rules for LLMH

$$\frac{}{\Gamma \vdash \texttt{Nothing} : \texttt{Maybe}\ \tau} \qquad \frac{}{\Gamma \vdash \texttt{Just} : \tau \to \texttt{Maybe}\ \tau}$$

$$\frac{\Gamma \vdash e_0 : \texttt{Maybe}\ \tau \qquad \Gamma \vdash e_1 : \tau' \qquad \Gamma, x : \tau \vdash e_2 : \tau'}{\Gamma \vdash \texttt{case}\ e_0\ \texttt{of Nothing} \to e_1\ ;\ \texttt{Just}\, x \to e_2\ :\ \tau'}$$

$$\frac{}{\Gamma \vdash [\,] : [\,\tau\,]} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : [\,\tau\,]}{\Gamma \vdash e_1{:}e_2\ :\ [\,\tau\,]}$$

$$\frac{\Gamma \vdash e_0 : [\tau] \qquad \Gamma \vdash e_1 : \tau' \qquad \Gamma,\ x : \tau,\ y : [\tau] \vdash e_2 : \tau'}{\Gamma \vdash \texttt{case}\ e_0\ \texttt{of}\ [\,] \to e_1\ ;\ x{:}y \to e_2\ :\ \tau'}$$

New operational rules (big step)

$$\frac{}{\texttt{Nothing} \;\Rightarrow\; \texttt{Nothing}}$$

$$\frac{}{\texttt{Just} \;\Rightarrow\; \texttt{Just}} \qquad \frac{}{\texttt{Just}\, e \;\Rightarrow\; \texttt{Just}\, e}$$

$$\frac{e \;\Rightarrow\; \texttt{Nothing} \qquad e_1 \;\Rightarrow\; v}{\texttt{case}\ e\ \texttt{of}\ \texttt{Nothing} \to e_1\ ;\ \texttt{Just}\, x \to e_2 \;\Rightarrow\; v}$$

$$\frac{e \;\Rightarrow\; \texttt{Just}\, e_0 \qquad e_2[x := e_0] \;\Rightarrow\; v}{\texttt{case}\ e\ \texttt{of}\ \texttt{Nothing} \to e_1\ ;\ \texttt{Just}\, x \to e_2 \;\Rightarrow\; v}$$

The new values are:

$$\texttt{Nothing} \qquad \texttt{Just} \qquad \texttt{Just}\ e$$

For lazy lists (as in Haskell)

$$\frac{}{[\,] \;\Rightarrow\; [\,]} \qquad\qquad \frac{}{e_1{:}e_2 \;\Rightarrow\; e_1{:}e_2}$$

$$\frac{e \;\Rightarrow\; [\,] \qquad e_1 \;\Rightarrow\; v}{\texttt{case}\ e\ \texttt{of}\ [\,] \to e_1\ ;\ x{:}y \to e_2 \;\Rightarrow\; v}$$

$$\frac{e \;\Rightarrow\; e_0{:}\,e_0' \qquad\qquad e_2[x, y := e_0, e_0'] \;\Rightarrow\; v}{\texttt{case}\ e\ \texttt{of}\ [\,] \to e_1\ ;\ x{:}y \to e_2 \;\Rightarrow\; v}$$

The new values are:

$$[\,] \qquad\qquad e_1 : e_2$$

Alternative for eager lists

Replace the rule

$$\frac{}{e_1{:}e_2 \;\Rightarrow\; e_1{:}e_2}$$

with the rule

$$\frac{e_1 \;\Rightarrow\; v_1 \qquad e_2 \;\Rightarrow\; v_2}{e_1{:}e_2 \;\Rightarrow\; v_1{:}v_2}$$

And the values are:

$$[\,] \qquad\qquad v_1 : v_2$$

Type inference algorithm for lists

Given (Γ, e) as input, the algorithm returns (θ_0, τ_0) as output.

$[\,]$: Let a be a fresh type variable.
Return $([], \; [\,a\,])$.

$e_1 : e_2$: Compute (θ_1, τ_1) for (Γ, e_1).
Compute (θ_2, τ_2) for (Γ, e_2).
Let θ_2' be the mgu for $[\,\tau_1\,]$ and τ_2.
Return $(\theta_1\,\theta_2\,\theta_2' \restriction_\Gamma, \;\; \tau_2\,\theta_2')$.

$\texttt{case}\ e_0\ \texttt{of}\ [\,] \to e_1\,;\ x{:}y \to e_2$: Compute (θ_0, τ_0) for (Γ, e_0).
Let a be a fresh type variable.
Let θ_0' be the mgu of τ_0 and $[\,a\,]$.
Compute (θ_1, τ_1) for $(\Gamma\theta_0\,\theta_0', e_1)$.
Define Γ_2 to be $\Gamma\,\theta_0\,\theta_0'\,\theta_1, \;\; x : a\,\theta_0'\,\theta_1, \;\; y : [\,a\,]\,\theta_0'\,\theta_1$.
Compute (θ_2, τ_2) for (Γ_2, e_2).
Let θ_2' be the mgu for $\tau_1\,\theta_2$ and τ_2.
Return $(\theta_0\,\theta_0'\,\theta_1\,\theta_2\,\theta_2' \restriction_\Gamma, \;\; \tau_2\,\theta_2')$.

Milner's "let polymorphism"

The context Γ maps variables to *polymorphic types*

$$\forall a_1, \ldots, a_k\ \tau$$

with explicit quantification over $k \geq 0$ type variables.

$$\frac{}{\Gamma \vdash x : \tau[a_1, \ldots, a_k := \tau_1, \ldots, \tau_k]}\ \Gamma(x) = \forall a_1, \ldots, a_k\ \tau$$

$$\frac{\Gamma, x : \tau_1 \vdash e_1 : \tau_1 \qquad \Gamma, x : \forall a_1, \ldots, a_k\ \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \mathtt{let}\ x\ \mathtt{=}\ e_1\ \mathtt{in}\ e_2 : \tau_2}\ (\star)$$

($\star$) $a_1, \ldots, a_k$ must not occur as *free type variables* in Γ (i.e., every occurrence of any of $a_1, \ldots, a_k$ in Γ must be bound by some $\forall$).