(Recap) 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.

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

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

($\star$) $a_1, \ldots, a_k$ 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 $\forall$).

Polymorphic type inference algorithm

We separate type variables into *polymorphic type variables* $a_0, a_1, a_2, \ldots$ and *monomorphic type variables* $b_0, b_1, b_2, \ldots$.

Given (Γ, e) as input, the algorithm returns (θ_0, τ_0) as output such that $\Gamma\theta_0 \vdash e : \tau_0$. (Here Γ may contain both polymorphic and monomorphic type variables, but θ_0 and τ_0 contain only monomorphic type variables.)

The algorithm proceeds recursively on the structure of e:

$\underline{x}$: Let $\tau = \Gamma(x)$.

Let τ^m be obtained from τ by replacing every polymorphic type variable in τ with a fresh monomorphic type variable.

Return $([], \tau^m)$.

$\texttt{let}\ x = e_1\ \texttt{in}\ e_2$: Let b be a fresh monomorphic type variable

Compute (θ_1, τ_1) for $((\Gamma, x : b), e_1)$.

Let θ_1' be the most general unifier of τ_1 and $b\theta_1$.

Consider $\tau_1' = \tau_1\theta_1'$ and $\Gamma' = \Gamma\theta_1\theta_1'$

Let τ_1^p be obtained from τ_1' by replacing every monomorphic type variable in τ_1' that does not occur in Γ' into a distinct polymorphic type variable.

Compute (θ_2, τ_2) for $((\Gamma\theta_1\theta_1',\ x : \tau_1^p),\ e_2)$.

Return $(\theta_1\theta_1'\theta_2 \restriction_\Gamma,\ \tau_2)$.

$\underline{e_1\, e_2}$: Compute (θ_1, τ_1) for (Γ, e_1).

Next compute (θ_2, τ_2) for $(\Gamma\,\theta_1, e_2)$.

Let b be a fresh monomorphic type variable.

Let θ_3 be the most general unifier of $\tau_1\,\theta_2$ and $\tau_2 \to b$.

Return $(\theta_1\,\theta_2\,\theta_3 \restriction_\Gamma,\ b\,\theta_3)$.

$\underline{\lambda\, x\, .\, e_0}$: Let b be a fresh monomorphic type variable.

Compute (θ, τ) for $(\,(\Gamma,\ x\colon b),\ e_0)$.

Return $(\theta \restriction_\Gamma,\ (b\,\theta) \to \tau)$.

Example data type declarations

```
data Days = Mon| Tue | Wed | Thu | Fri | Sat | Sun ;

data Pair a b = Pair a b ;

data Maybe a = Just a | Nothing ;

data List a = Nil | Cons a (List a) ;

data Tree a = Leaf | Node a (Tree a) (Tree a) ;

data BinTree2 a = ValLeaf a | EmptyNode (BinTree2 a) (BinTree2 a) ;

data WideTree a = ValNode a (List (WideTree a)) ;

data TwoThreeTree a b = DataLeaf a b
  | TwoNode a (TwoThreeTree a b) (TwoThreeTree a b)
  | ThreeNode a a (TwoThreeTree a b) (TwoThreeTree a b) (TwoThreeTree a b)

data LambdaCalc = Num Integer | Fun (LambdaCalc -> LambdaCalc) ;
```

Expressions associated with data types

Data type declaration:

$$\texttt{data}\ F\ a_1\ \ldots a_k\ \ \texttt{=}\ \ C_1\ \tau_{11}\ \ldots \tau_{1n_1}\ \mid\ \ldots\ \mid\ C_m\ \tau_{m1}\ \ldots \tau_{mn_m}$$

Constructor expressions:

$$C_1 \quad \ldots \quad C_m$$

Associated case expression:

$$\texttt{case}\ e\ \texttt{of}\ \ C_1\ x_{11}\ \ldots\ x_{1n_1} \to e_1\ \ \texttt{;}\ \ \ldots\ \ \texttt{;}\ \ C_m\ x_{m1}\ \ldots\ x_{mn_m} \to e_m$$

Big-step operational rules

$$(\texttt{data}\ F\ a_1\ \dots a_k \ \ \texttt{=}\ \ C_1\ \tau_{11}\ \dots \tau_{1n_1} \ \mid\ \dots\ \mid\ C_m\ \tau_{m1}\ \dots \tau_{mn_m})$$

$$\frac{}{C_i \ \Rightarrow\ C_i} \qquad\qquad \frac{e \Rightarrow C_i\, e_1 \dots e_{n-1}}{e\, e_n \ \Rightarrow\ C_i\, e_1 \dots e_n}\ (n \leq n_i)$$

$$\frac{e \Rightarrow C_i\, e'_1 \dots e'_{n_i} \qquad e_i[x_{i1}, \dots, x_{in_i} := e'_1, \dots, e'_{n_i}] \Rightarrow v}{\texttt{case}\ e\ \texttt{of}\ C_1\, x_{11} \dots x_{1n_1} \to e_1\ ;\ \dots\ ;\ C_m\, x_{m1} \dots x_{mn_m} \to e_m \ \Rightarrow\ v}$$

Typing rules

$$(\texttt{data}\ F\ a_1\ \ldots a_k\ \ \texttt{=}\ \ C_1\ \tau_{11}\ \ldots \tau_{1n_1}\ \mid\ \ldots\ \mid\ C_m\ \tau_{m1}\ \ldots \tau_{mn_m}\ )$$

$$\frac{}{\Gamma \vdash C_i : \tau_{i1}\,\theta \to \cdots \to \tau_{in_i}\,\theta \to F\sigma_1 \ldots \sigma_k}$$

$$\frac{\Gamma \vdash e : F\sigma_1 \ldots \sigma_k \qquad \Gamma, x_{11} : \tau_{11}\,\theta, \ldots, x_{1n_1} : \tau_{1n_1}\,\theta \vdash e_1 : \tau \quad \ldots \quad \Gamma, x_{m1} : \tau_{m1}\,\theta, \ldots, x_{mn_m} : \tau_{mn_m}\,\theta \vdash e_m : \tau}{\Gamma \vdash (\texttt{case}\ e\ \texttt{of}\ C_1\ x_{11}\ \ldots\ x_{1n_1}\ \texttt{->}\ e_1\ \texttt{;}\ \ldots\ \texttt{;}\ C_m\ x_{m1}\ \ldots\ x_{mn_m}\ \texttt{->}\ e_m) : \tau}$$

In both rules θ is the substitution $[a_1, \ldots, a_k := \sigma_1 \ldots \sigma_k]$.