M Algorithm

in simple(monomorphic) type system

1 M Algorithm

$$M : TyEnv \times Exp \times Type \to Subst$$

$$
\begin{array}{rcll}
M(\Gamma, n, \tau) & = & \text{unify}(\iota, \tau) & \\
M(\Gamma, x, \tau) & = & \text{unify}(\tau, \tau') & \text{if } x : \tau' \in \Gamma \\
M(\Gamma, \lambda x.e, \tau) & = & \text{let } S = \text{unify}(\alpha_1 \to \alpha_2, \tau) & \text{new } \alpha_1, \alpha_2 \\
 & & \quad S' = M(S\Gamma + x : S\alpha_1, e, S\alpha_2) & \\
 & & \text{in } S'S & \\
M(\Gamma, e\ e', \tau) & = & \text{let } S = M(\Gamma, e, \alpha \to \tau) & \text{new } \alpha \\
 & & \quad S' = M(S\Gamma, e', S\alpha) & \\
 & & \text{in } S'S & \\
M(\Gamma, e + e', \tau) & = & \text{let } S = \text{unify}(\iota, \tau) & \\
 & & \quad S' = M(S\Gamma, e, \iota) & \\
 & & \quad S'' = M(S'S\Gamma, e', \iota) & \\
 & & \text{in } S''S'S & \\
M(\Gamma, e \text{ and } e', \tau) & = & \text{let } S = \text{unify}(Bool, \tau) & \\
 & & \quad S' = M(S\Gamma, e, Bool) & \\
 & & \quad S'' = M(S'S\Gamma, e', Bool) & \\
 & & \text{in } S''S'S & \\
M(\Gamma, e = e', \tau) & = & \text{let } S = \text{unify}(Bool, \tau) & \text{new comparable } \alpha \\
 & & \quad S' = M(S\Gamma, e, S\alpha) & \\
 & & \quad S'' = M(S'S\Gamma, e', S'S\alpha) & \\
 & & \text{in } S''S'S & \\
M(\Gamma, \text{let } x = e_1 \ in \ e_2, \tau) & = & \text{let } S = M(\Gamma, e_1, \alpha) & \text{new } \alpha \\
 & & \quad S' = M(S\Gamma + x : S\alpha, e_2, S\tau) & \\
 & & \text{in } S'S & \\
M(\Gamma, \text{let rec } f = \text{fn } x \ \Rightarrow e_1 \ in \ e_2, \tau) & = & \text{let } S = M(\Gamma + f : \alpha, \text{FN}(x, e'), \alpha) & \text{new } \alpha \\
 & & \quad S' = M(S\Gamma + f : S\alpha, e_2, S\tau) & \\
 & & \text{in } S'S &
\end{array}
$$

$$
\begin{array}{rcll}
M(\Gamma, \text{if } e_1 \text{then } e_2 \text{ else} e_3) & = & \text{let } S = M(\Gamma, e_1, \mathit{Bool}) & \\
 & & \quad S' = M(S\Gamma, e_2, S\tau) & \\
 & & \quad S'' = M(S'S\Gamma, e_3, S'S\tau) & \\
 & & \text{in } S''S'S & \\
M(\Gamma, \text{read}, \tau) & = & \text{unify}(\iota, \tau) & \\
M(\Gamma, \text{write } e, \tau) & = & \text{let } S = \text{unify}(\alpha, \tau) & \text{new writable } \alpha \\
 & & \quad S' = M(S\Gamma, e, S\tau) & \\
 & & \text{in } S'S & \\
M(\Gamma, \text{malloc } e, \tau) & = & \text{let } S = M(\Gamma, e, \alpha) & \text{new } \alpha \\
 & & \quad S' = \text{unify}(\mathit{Loc}(S\alpha), S\tau) & \\
 & & \text{in } S'S & \\
M(\Gamma, e_1 := e_2, \tau) & = & \text{let } S = M(\Gamma, e_1, \mathit{Loc}(\tau)) & \\
 & & \quad S' = M(S\Gamma, e_2, S\tau) & \\
 & & \text{in } S'S & \\
M(\Gamma, !e, \tau) & = & M(\Gamma, e, \mathit{Loc}(\tau)) & \\
M(\Gamma, e_1;e_2, \tau) & = & \text{let } S = M(\Gamma, e, \alpha) & \text{new } \alpha \\
 & & \quad S' = M(S\Gamma, e_2, S\tau) & \\
 & & \text{in } S'S & \\
M(\Gamma, (e_1,e_2), \tau) & = & \text{let } S = \text{unify}(\mathit{Pair}(\alpha_1, \alpha_2), \tau) & \text{new } \alpha_1, \alpha_2 \\
 & & \quad S' = M(S\Gamma, e_2, S\alpha_1) & \\
 & & \quad S'' = M(S'S\Gamma, e_2, S'S\alpha_2) & \\
 & & \text{in } S''S'S & \\
M(\Gamma, e.1, \tau) & = & M(\Gamma, e, \mathit{Pair}(\tau, \alpha)) & \text{new } \alpha \\
M(\Gamma, e.2, \tau) & = & M(\Gamma, e, \mathit{Pair}(\alpha, \tau)) & \text{new } \alpha
\end{array}
$$