Mini-ML type-inference problem:

(b) $\boxed{\text{given } \Gamma \text{ \& } M, \text{ does there exist } \sigma \text{ such that } \Gamma \vdash M : \sigma \text{ holds?}}$

Two examples involving self-application

$$M \triangleq \mathtt{let}\ f = \lambda x_1\,(\lambda x_2\,(x_1))\ \mathtt{in}\ f\ f$$

$$M' \triangleq (\lambda f\,(f\ f))\ \lambda x_1\,(\lambda x_2\,(x_1))$$

Are M and M' typeable in the Mini-ML type system?

(in the empty typing environment)

Figure 1 [p15]

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 : \quad , x_2 : \quad \vdash x_1 :}\,(C3)}{x_1 : \quad \vdash \lambda x_2(x_1) :}\,(C2)}{\{\} \vdash \lambda x_1(\lambda x_2(x_1)) :}\,(C1) \qquad \dfrac{\dfrac{}{f : \quad \vdash f :}\,(C5) \quad \dfrac{}{f : \quad \vdash f :}\,(C6)}{f : \quad \vdash ff :}\,(C4)}{\{\} \vdash \text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } ff \quad :}\,(C0)$$

Figure 1 [p 19]

$$\forall\{\}\tau_3 \qquad \forall\{\}\tau_5$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1:\tau_3,\ x_2:\tau_5 \vdash x_1:\tau_6}\,(C3)}{x_1:\tau_3 \vdash \lambda x_2(x_1):\tau_4}\,(C2)}{\{\} \vdash \lambda x_1(\lambda x_2(x_1)):\tau_2}\,(C1) \qquad \dfrac{\dfrac{}{f:\forall A(\tau_2) \vdash f:\tau_7}\,(C5) \quad \dfrac{}{f:\forall A(\tau_2) \vdash f:\tau_8}\,(C6)}{f:\forall A(\tau_2) \vdash f\,f:\tau_1}\,(C4)}{\{\} \vdash \mathrm{let}\ f = \lambda x_1(\lambda x_2(x_1))\ \mathrm{in}\ f\,f \ :\ \tau_1}\,(C0)$$

Constraints generated while inferring a type for $\texttt{let}\ f = \lambda x_1\,(\lambda x_2\,(x_1))\ \texttt{in}\ f\ f$

$$A = ftv(\tau_2) \tag{C0}$$

$$\tau_2 = \tau_3 \to \tau_4 \tag{C1}$$

$$\tau_4 = \tau_5 \to \tau_6 \tag{C2}$$

$$\forall\{\,\}\,(\tau_3) \succ \tau_6, \text{ i.e. } \tau_3 = \tau_6 \tag{C3}$$

$$\tau_7 = \tau_8 \to \tau_1 \tag{C4}$$

$$\forall A\,(\tau_2) \succ \tau_7 \tag{C5}$$

$$\forall A\,(\tau_2) \succ \tau_8 \tag{C6}$$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{array}{l} \tau_6 = \alpha_1 \\ \tau_5 = \alpha_2 \end{array}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{aligned} \tau_6 &= \alpha_1 \\ \tau_5 &= \alpha_2 \end{aligned}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

(C5) : $\forall\{\alpha_1, \alpha_2\}(\alpha_1 \to (\alpha_2 \to \alpha_1)) > \tau_7 \overset{(C4)}{=} \tau_8 \to \tau_1$

(C6) : " " " $> \tau_8$

So $\begin{cases} \tau_8 \to \tau_1 = \tau_9 \to (\tau_{10} \to \tau_9) \\ \tau_8 = \tau_{11} \to (\tau_{12} \to \tau_{11}) \end{cases}$ for some $\tau_9, \tau_{10}, \tau_{11}, \tau_{12}$

$$\tau_2 \overset{(C1)}{=\!=} \tau_3 \to \tau_4 \overset{(C2)}{=\!=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=\!=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{array}{l} \tau_6 = \alpha_1 \\ \tau_5 = \alpha_2 \end{array}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

(C5) : $\forall\{\alpha_1, \alpha_2\}(\alpha_1 \to (\alpha_2 \to \alpha_1)) > \tau_7 \overset{(C4)}{=\!=} \tau_8 \to \tau_1$

(C6) : " " " $> \tau_8$

So $\begin{cases} \tau_8 = \tau_9 \ \& \ \tau_1 = (\tau_{10} \to \tau_9) \\ \tau_8 = \tau_{11} \to (\tau_{12} \to \tau_{11}) \end{cases}$ for some $\tau_9, \tau_{10}, \tau_{11}, \tau_{12}$

So $\tau_1 = \tau_{10} \to \tau_9 = \tau_{10} \to \tau_8 = \tau_{10} \to (\tau_{11} \to (\tau_{12} \to \tau_{11}))$

Thus

$$\{\} \vdash (\text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f) : \tau_{10} \to (\tau_{11} \to (\tau_{12} \to \tau_{11}))$$

holds for any $\tau_{10}, \tau_{11}, \tau_{12}$

So

$$\boxed{\vdash (\text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f) : \forall \alpha_1, \alpha_2, \alpha_3 (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_2)))}$$

Two examples involving self-application

$$M \triangleq \texttt{let}\ f = \lambda x_1\,(\lambda x_2\,(x_1))\ \texttt{in}\ f\ f$$

$$M' \triangleq (\lambda f\,(f\ f))\,\lambda x_1\,(\lambda x_2\,(x_1))$$

Are M and M' typeable in the Mini-ML type system?

[Page 17]

The constraints generated from trying to type

$$(\lambda f(ff))\ \lambda x_1(\lambda x_2(x_1))$$

give

$$\tau_7 \overset{(C13)}{=} \tau_4 \overset{(C12)}{=} \tau_6 \overset{(C11)}{=} \tau_7 \to \tau_5$$

[Page 17]

The constraints generated from trying to type

$$(\lambda f(ff))\ \lambda x_1(\lambda x_2(x_1))$$

give

$$\tau_7 \overset{(C13)}{=} \tau_4 \overset{(C12)}{=} \tau_6 \overset{(C11)}{=} \tau_7 \to \tau_5$$

these cannot be equal – they have different numbers of the symbol "$\to$" in them

Two examples involving self-application

$$M \triangleq \texttt{let}\ f = \lambda x_1\,(\lambda x_2\,(x_1))\ \texttt{in}\ f\ f$$

$$M' \triangleq (\lambda f\,(f\ f))\ \lambda x_1\,(\lambda x_2\,(x_1))$$

Are M and M' typeable in the Mini-ML type system?

This is not typeable

Principal type schemes for closed expressions

A type scheme $\forall A\,(\tau)$ is the *principal* type scheme of a closed Mini-ML expression M if

Principal type schemes for closed expressions

A type scheme $\forall A\,(\tau)$ is the *principal* type scheme of a closed Mini-ML expression M if

(a) $\vdash M : \forall A\,(\tau)$

(i.e. $\{\} \vdash M : \tau$ is provable
& $A = ftv(\tau)$)

Principal type schemes for closed expressions

A type scheme $\forall A\,(\tau)$ is the *principal* type scheme of a closed Mini-ML expression M if

(a) $\vdash M : \forall A\,(\tau)$

(b) for any other type scheme $\forall A'\,(\tau')$,
if $\vdash M : \forall A'\,(\tau')$, then $\forall A\,(\tau) \succ \tau'$

Principal type schemes for closed expressions

A type scheme $\forall A\,(\tau)$ is the *principal* type scheme of a closed Mini-ML expression M if

(a) $\vdash M : \forall A\,(\tau)$

(b) for any other type scheme $\forall A'\,(\tau')$, if $\vdash M : \forall A'\,(\tau')$, then $\forall A\,(\tau) \succ \tau'$

eg $\forall \alpha_1, \alpha_2, \alpha_3\,(\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_2)))$ is principal type scheme

$\text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f$

Theorem (Hindley; Damas-Milner)

Theorem. If the closed Mini-ML expression M is typeable (i.e. $\vdash M : \sigma$ holds for some type scheme σ), then there is a principal type scheme for M.

Theorem (Hindley; Damas-Milner)

Theorem. If the closed Mini-ML expression M is typeable (i.e. $\vdash M : \sigma$ holds for some type scheme σ), then there is a principal type scheme for M.

Indeed, there is an algorithm which, given any closed Mini-ML expression M as input, decides whether or not it is typeable and returns a principal type scheme if it is.

An ML expression with a principal type scheme hundreds of pages long

$$
\begin{array}{l}
\texttt{let}\ \mathit{pair} = \lambda x\,(\lambda y\,(\lambda z\,(z\,x\,y)))\ \texttt{in} \\
\quad \texttt{let}\ x_1 = \lambda y\,(\mathit{pair}\,y\,y)\ \texttt{in} \\
\quad\quad \texttt{let}\ x_2 = \lambda y\,(x_1(x_1\,y))\ \texttt{in} \\
\quad\quad\quad \texttt{let}\ x_3 = \lambda y\,(x_2(x_2\,y))\ \texttt{in} \\
\quad\quad\quad\quad \texttt{let}\ x_4 = \lambda y\,(x_3(x_3\,y))\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{let}\ x_5 = \lambda y\,(x_4(x_4\,y))\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad x_5(\lambda y\,(y))
\end{array}
$$

Unification of ML types

There is an algorithm $\boldsymbol{mgu}$ which when input two Mini-ML types $\boldsymbol{\tau_1}$ and $\boldsymbol{\tau_2}$ decides whether $\boldsymbol{\tau_1}$ and $\boldsymbol{\tau_2}$ are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Moreover, if they are unifiable, $\mathit{mgu}(\tau_1, \tau_2)$ returns the *most general unifier*—an S satisfying both (a) and

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Moreover, if they are unifiable, $mgu(\tau_1, \tau_2)$ returns the *most general unifier*—an S satisfying both (a) and

(b) for all $S' \in \mathbf{Sub}$, if $S'(\tau_1) = S'(\tau_2)$, then $S' = TS$ for some $T \in \mathbf{Sub}$

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Moreover, if they are unifiable, $mgu(\tau_1, \tau_2)$ returns the *most general unifier*—an S satisfying both (a) and

(b) for all $S' \in \mathbf{Sub}$, if $S'(\tau_1) = S'(\tau_2)$, then $S' = TS$ for some $T \in \mathbf{Sub}$

(any other substitution S' can be factored through S, by specialising S with T)

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Moreover, if they are unifiable, $mgu(\tau_1, \tau_2)$ returns the *most general unifier*—an S satisfying both (a) and

(b) for all $S' \in \mathbf{Sub}$, if $S'(\tau_1) = S'(\tau_2)$, then $S' = TS$ for some $T \in \mathbf{Sub}$

(any other substitution S' can be factored through S, by specialising S with T)

By convention $mgu(\tau_1, \tau_2) = FAIL$ if (and only if) τ_1 and τ_2 are not unifiable.

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

<u>capture-avoiding</u> substitution

eg if $S = \{\alpha \mapsto \beta\}$ and $\sigma_1 = \forall \beta\,(\alpha)$

then $S\sigma_1 = \forall \gamma\,(\beta)$ (any $\gamma \neq \beta$)

$S\sigma_1 \neq \forall \beta\,(\beta)$

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

Such a solution is *principal* if given any other, (S', σ'), there is some $T \in \mathbf{Sub}$ with $TS = S'$ and $T(\sigma) \succ \sigma'$.

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

Such a solution is *principal* if given any other, (S', σ'), there is some $T \in \mathbf{Sub}$ with $T S = S'$ and $T(\sigma) \succ \sigma'$.

(For type schemes σ and σ', with $\sigma' = \forall A'\,(\tau')$ say, we define $\boxed{\sigma \succ \sigma'}$ to mean $A' \cap ftv(\sigma) = \{\}$ and $\sigma \succ \tau'$.)

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

Such a solution is *principal* if given any other, (S', σ'), there is some $T \in \mathbf{Sub}$ with $T S = S'$ and $T(\sigma) \succ \sigma'$.

(For type schemes σ and σ', with $\sigma' = \forall A'\,(\tau')$ say, we define $\boxed{\sigma \succ \sigma'}$ to mean $A' \cap ftv(\sigma) = \{\}$ and $\sigma \succ \tau'$.)

eg $\forall \alpha\,(\alpha \to \beta) \succ (\beta \to \beta) \to \beta$

but $\forall \alpha\,(\alpha \to \beta) \not\succ \forall \beta\,((\beta \to \beta) \to \beta)$

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\,\texttt{true} : ?}$$

has solutions:

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\,\texttt{true} : ?}$$

has solutions:

- $S_1 = \{\beta \mapsto bool\}, \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall \alpha \, (\beta \to (\gamma \to \alpha)) \vdash x\,\texttt{true} : ?}$$

has solutions:

- $S_1 = \{\beta \mapsto bool\}, \sigma_1 = \forall \alpha \, (\gamma \to \alpha)$
- $S_2 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_2 = \forall \alpha' \, (\alpha \to \alpha')$

NB $S_2(\forall \alpha \, (\beta \to (\gamma \to \alpha))) = \forall \alpha' (bool \to (\alpha \to \alpha'))$

(any $\alpha' \neq \alpha$)

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\;\texttt{true} : ?}$$

has solutions:

- $S_1 = \{\beta \mapsto bool\}, \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$
- $S_2 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_2 = \forall\alpha'\,(\alpha \to \alpha')$
- $S_3 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_3 = \forall\alpha'\,(\alpha \to (\alpha' \to \alpha'))$

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\,\mathtt{true} : ?}$$

has solutions:

- $S_1 = \{\beta \mapsto bool\}, \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$
- $S_2 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_2 = \forall\alpha'\,(\alpha \to \alpha')$
- $S_3 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_3 = \forall\alpha'\,(\alpha \to (\alpha' \to \alpha'))$
- $S_4 = \{\beta \mapsto bool, \gamma \mapsto bool\}, \sigma_3 = \forall\{\,\}\,(bool \to bool)$

Example typing problem and solutions

Typing problem

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\,\mathtt{true} : ?}$$

has solutions:

- $S_1 = \{\beta \mapsto bool\}, \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$
- $S_2 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_2 = \forall\alpha'\,(\alpha \to \alpha')$
- $S_3 = \{\beta \mapsto bool, \gamma \mapsto \alpha\}, \sigma_3 = \forall\alpha'\,(\alpha \to (\alpha' \to \alpha'))$
- $S_4 = \{\beta \mapsto bool, \gamma \mapsto bool\}, \sigma_3 = \forall\{\,\}\,(bool \to bool)$

Both (S_1, σ_1) and (S_2, σ_2) are in fact principal solutions.

Properties of the Mini-ML typing relation with respect to substitution and type scheme specialisation

- If $\Gamma \vdash M : \sigma$, then for any type substitution $S \in \mathbf{Sub}$

$$S\Gamma \vdash M : S\sigma$$

Properties of the Mini-ML typing relation with respect to substitution and type scheme specialisation

- If $\Gamma \vdash M : \sigma$, then for any type substitution $S \in \mathbf{Sub}$

$$S\Gamma \vdash M : S\sigma$$

- If $\Gamma \vdash M : \sigma$ and $\sigma \succ \sigma'$, then

$$\Gamma \vdash M : \sigma'$$

Requirements for a principal typing algorithm, pt

pt operates on typing problems $\Gamma \vdash M : ?$ (consisting of a typing environment Γ and a Mini-ML expression M).

It returns either a pair (S, τ) consisting of a type substitution $S \in \mathbf{Sub}$ and a Mini-ML type τ, or the exception $FAIL$.

- If $\Gamma \vdash M : ?$ has a solution (cf. Slide 28), then $pt(\Gamma \vdash M : ?)$ returns (S, τ) for some S and τ;
  moreover, setting $A = (ftv(\tau) - ftv(S\,\Gamma))$, then $(S, \forall A\,(\tau))$ is a principal solution for the problem $\Gamma \vdash M : ?$.
- If $\Gamma \vdash M : ?$ has no solution, then $pt(\Gamma \vdash M : ?)$ returns $FAIL$.

How the principal typing algorithm pt works

$$pt(\Gamma \vdash M : ?) = (S, \tau) \mid FAIL$$

- Call pt recursively following the structure of M and guided by the typing rules, bottom-up.
- Thread substitutions sequentially and compose them together when returning from a recursive call.
- When types need to agree to satisfy a typing rule, use mgu (and pt returns $FAIL$ only if mgu does).
- When types are unknown, generate a fresh type variable.

Some of the clauses in a definition of $\boldsymbol{pt}$

Function abstractions: $pt(\Gamma \vdash \lambda x\,(M) : ?) \triangleq$
  let α = **fresh in**
  let $(S, \tau) = pt(\Gamma, x : \alpha \vdash M : ?)$ **in** $(S, S(\alpha) \to \tau)$

$$(\mathrm{fn})\ \frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \lambda x(M) : \tau_1 \to \tau_2}\ x \notin \operatorname{dom} \Gamma$$

Some of the clauses in a definition of pt

$$(app)\ \frac{\Gamma \vdash M : \tau_1 \to \tau_2 \qquad \Gamma \vdash N : \tau_1}{\Gamma \vdash M N : \tau_2}$$

Function applications: $pt(\Gamma \vdash M_1\, M_2 : ?) \triangleq$

$\quad$ **let** $(S_1, \tau_1) = pt(\Gamma \vdash M_1 : ?)$ **in**

$\quad$ **let** $(S_2, \tau_2) = pt(S_1\,\Gamma \vdash M_2 : ?)$ **in**

$\quad$ **let** α = **fresh in**

$\quad$ **let** $S_3 = mgu(S_2\,\tau_1, \tau_2 \to \alpha)$ **in** $(S_3 S_2 S_1, S_3(\alpha))$

$$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$$

$$S_1\Gamma \vdash M_1 : \tau_1$$

$$\hookrightarrow pt(\Gamma \vdash M_1 M_2 : ?) =$$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

+slide 28

$S_2 S_1 \Gamma \vdash M_1 : S_2 \tau_1$

$pt(S_1 \Gamma \vdash M_2 : ?) = (S_2, \tau_2)$

$S_2 S_1 \Gamma' \vdash M_2 : \tau_2$

$\hookrightarrow pt(\Gamma' \vdash M_1 M_2 : ?) =$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

$pt(S_1\Gamma \vdash M_2 : ?) = (S_2, \tau_2)$

$mgu(S_2\tau_1, \tau_2 \to \alpha) = S_3$

+ Slide 28

$S_3\tau_2 \to S_3\alpha$

$\|$

$S_3S_2S_1\Gamma \vdash M_1 : S_3S_2\tau_1$

$S_3S_2S_1\Gamma' \vdash M_2 : S_3\tau_2$

$\hookrightarrow pt(\Gamma' \vdash M_1M_2 : ?) =$

$(S_3S_2S_1 , S_3\alpha)$

$$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$$

$$pt(S_1\Gamma \vdash M_2 : ?) = (S_2, \tau_2)$$

$$mgu(S_2\tau_1, \tau_2 \to \alpha) = S_3$$

$$\frac{S_3S_2S_1\Gamma \vdash M_1 : \overset{S_3S_2\tau_1}{\overset{\|}{S_3\tau_2}} \to S_3\alpha \qquad S_3S_2S_1\Gamma' \vdash M_2 : S_3\tau_2}{S_3S_2S_1\Gamma' \vdash M_1M_2 : S_3\alpha}\ (app)$$

$$pt(\Gamma' \vdash M_1M_2 : ?) = (S_3S_2S_1, S_3\alpha)$$