Last time on **Types**...

- Modified ML with polymorphic types anywhere

Identity, Generalisation and Specialisation

$$\Gamma \vdash x : \pi \quad \text{if } (x : \pi) \in \Gamma \qquad \text{(id)}$$

$$\frac{\Gamma \vdash M : \pi}{\Gamma \vdash M : \forall\, \alpha\, (\pi)} \quad \text{if } \alpha \notin \mathit{ftv}(\Gamma) \qquad \text{(gen)}$$

$$\frac{\Gamma \vdash M : \forall\, \alpha\, (\pi)}{\Gamma \vdash M : \pi[\pi'/\alpha]} \qquad \text{(spec)}$$

Last time on **Types**...

- Modified ML with polymorphic types anywhere
- Polymorphic λ-calculus

PLC type system

$$\Gamma \vdash x : \tau \quad \text{if } (x : \tau) \in \Gamma \tag{var}$$

$$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \lambda\, x : \tau_1\,(M) : \tau_1 \to \tau_2} \quad \text{if } x \notin \mathit{dom}(\Gamma) \tag{fn}$$

$$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\, M_2 : \tau_2} \tag{app}$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash \Lambda\, \alpha\,(M) : \forall\, \alpha\,(\tau)} \quad \text{if } \alpha \notin \mathit{ftv}(\Gamma) \tag{gen}$$

$$\frac{\Gamma \vdash M : \forall\, \alpha\,(\tau_1)}{\Gamma \vdash M\, \tau_2 : \tau_1[\tau_2/\alpha]} \tag{spec}$$

Last time on **Types**...

- Modified ML with polymorphic types anywhere
- Polymorphic λ-calculus
  - $\Lambda\alpha\,(\lambda x : \alpha\,(x))\ : \forall\alpha(\alpha \to \alpha)$
- Decideability of typing for PLC

Today...

- Results on reduction (semantics) of PLC
- Encoding data types in PLC (part 1)

Beta-reduction of PLC expressions

M beta-reduces to M' in one step, $\boxed{M \to M'}$ means M' can be obtained from M (up to alpha-conversion, of course) by replacing a subexpression which is a redex by its corresponding reduct. The redex-reduct pairs are of two forms:

$$(\lambda\, x : \tau\, (M_1))\, M_2 \to M_1[M_2/x]$$
$$(\Lambda\, \alpha\, (M))\, \tau \to M[\tau/\alpha].$$

$M \to^* M'$ indicates a chain of finitely † many beta-reductions.

(† possibly zero—which just means M and M' are alpha-convertible).

M is in beta-normal form if it contains no redexes.

Properties of PLC beta-reduction on typeable expressions

Suppose $\Gamma \vdash M : \tau$ is provable in the PLC type system. Then the following properties hold:

Subject Reduction. If $M \to M'$, then $\Gamma \vdash M' : \tau$ is also a provable typing.

Subject reduction requires substitution lemma...

Subject Reduction. If $M \to M'$, then $\Gamma \vdash M' : \tau$ is also a provable typing.

For example for: $(\lambda\, x : \sigma\, (M_1))\, M_2 \to M_1[M_2/x]$

$$
(\text{app})\,\dfrac{(\text{abs})\,\dfrac{\boxed{\Gamma, x : \sigma \vdash M_1 : \tau}}{\Gamma \vdash \lambda\, x : \sigma\,(M_1) : \sigma \to \tau} \qquad \boxed{\Gamma \vdash M_2 : \sigma}}{\Gamma \vdash (\lambda\, x : \sigma\,(M_1))\, M_2 : \tau}
\quad\longrightarrow\quad
\dfrac{\boxed{\Gamma, x : \sigma \vdash M_1 : \tau} \qquad \boxed{\Gamma \vdash M_2 : \sigma}}{\Gamma \vdash M_1[M_2/x] : \tau}
$$

Lemma(substitution)

If $\Gamma, x : \sigma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \sigma$ then $\Gamma \vdash M_1[M_2/x] : \tau$.

Proof By induction over the typing relation on M_1.

Properties of PLC beta-reduction on typeable expressions

Suppose $\Gamma \vdash M : \tau$ is provable in the PLC type system. Then the following properties hold:

Subject Reduction. If $M \to M'$, then $\Gamma \vdash M' : \tau$ is also a provable typing.

Church Rosser Property. If $M \to^* M_1$ and $M \to^* M_2$, then there is M' with $M_1 \to^* M'$ and $M_2 \to^* M'$.

Strong Normalisation Property. There is no infinite chain $M \to M_1 \to M_2 \to \ldots$ of beta-reductions starting from M.

Theorem 16, p.43

Extra

Church-Rosser (CR) + Strong Normalisation (SN)
$\Rightarrow$ exists unique *beta*-normal forms for **typeable** PLC expressions

- **Existence**: start from M and reduce any redexes... by (SN) this must eventually stop
- **Uniqueness:** by (CR), if $M \rightarrow^* M_1$ and $M \rightarrow^* M_2$ then

$$\begin{tikzcd} & M_1 \arrow[dr, "*"] & \\ M \arrow[ur, "*"] \arrow[dr, "*"'] & & M' \\ & M_2 \arrow[ur, "*"'] & \end{tikzcd}$$

(where $M_1 \rightarrow^* M'$ and $M_2 \rightarrow^* M'$ are zero length β-reduction chanins if M_1 and M_2 are in β-normal form).

Y-combinator

$$Y = \lambda f.((\lambda x.f(x\,x))\,(\lambda x.f(x\,x)))$$

- Satisfies fixed-point combinator equation $Y\,f = f(Y\,f)$
- for some f, $Y\,f$ does not have a beta-normal form
  (see Remark 17, p.43, where $f = id$)
- Y is not typeable in PLC

Exercise (2 min). Show that $Y\,id$ has an infinite β-reduction chain (i.e., no β-normal form)

$$Y = \lambda f\,(\lambda x.\, f\,(x\ x))\,(\lambda x.\, f\,(x\ x))$$

$$Y\ id \xrightarrow{\beta} \cancel{f}\,(\lambda x.\, id\,(x\ x))(\lambda x.\, id\,(x\ x))$$

$$\xrightarrow{*} id\,(\lambda x.\, id\,(x\ x))(\lambda x.\, id\,(x\ x)))$$

or, alternatively

(reduce id on the inside) $\xrightarrow{\beta} (\lambda x.\ (x\ x))\ (\lambda x.\ x\ x)$

$$\to (\lambda x.\,(x\ x))(\lambda x.\, x\ x)$$

PLC beta-conversion, $=_\beta$

By definition, $\boxed{M =_\beta M'}$ holds if there is a finite chain

$$M - \cdot - \cdots - \cdot - M'$$

where each $-$ is either $\rightarrow$ or $\leftarrow$, i.e. a beta-reduction in one direction or the other. (A chain of length zero is allowed—in which case M and M' are equal, up to alpha-conversion, of course.)

Church Rosser + Strong Normalisation properties imply that, for typeable PLC expressions, $M =_\beta M'$ holds if and only if there is some beta-normal form N with

$$M \rightarrow^* N \; {}^*\!\!\leftarrow M'$$

Data types in PLC (Section 4.4)

- define a suitable PLC type for the data
- define suitable PLC expressions for values & on the data
- show PLC expressions have correct typings & behaviour (use the semantics)

Polymorphic booleans

$$bool \stackrel{\text{def}}{=} \forall\,\alpha\,(\alpha \to (\alpha \to \alpha))$$

$$True \stackrel{\text{def}}{=} \Lambda\,\alpha\,(\lambda\,x_1 : \alpha, x_2 : \alpha\,(x_1))$$

$$False \stackrel{\text{def}}{=} \Lambda\,\alpha\,(\lambda\,x_1 : \alpha, x_2 : \alpha\,(x_2))$$

$$if \stackrel{\text{def}}{=} \Lambda\,\alpha\,(\lambda\,b : bool, x_1 : \alpha, x_2 : \alpha\,(b\,\alpha\,x_1\,x_2))$$

Exercise (5 min)

$$\textit{Given } \Gamma \vdash M_1 : \textit{bool} \ , \ \Gamma \vdash M_2 : \tau \ , \ \Gamma \vdash M_3 : \tau$$

$$\textit{and } \begin{cases} M_1 \to^* \textit{True} \\ M_2 \to^* N_2 \\ M_3 \to^* N_3 \end{cases}$$

Exercise (5 min)

$$\textit{Given } \Gamma \vdash M_1 : \textit{bool} \;,\; \Gamma \vdash M_2 : \tau \;,\; \Gamma \vdash M_3 : \tau$$

$$\textit{and } \begin{cases} M_1 \to^* \textit{True} \\ M_2 \to^* N_2 \\ M_3 \to^* N_3 \end{cases}$$

$$\textit{then} \qquad \mathbf{if}\,\tau\; M_1\; M_2\; M_3 \;\to^*\; ?$$

if :

$$\forall\alpha.(\underbrace{\text{bool}}_{b} \to \underbrace{\alpha}_{x_1} \to \underbrace{\alpha}_{x_2} \to \alpha)$$

$$\text{if } \tau \; M_1 \; M_2 \; M_3$$

$$\to^* \text{if } \tau \; \text{True} \; M_2 \; M_3$$

$$\overset{\text{by def. if}}{=} (\Lambda\alpha(\lambda b:\text{bool}, \lambda x_1:\alpha \ldots)\tau \; \text{True} \; M_2 \; M_3$$

$$\xrightarrow{\beta} (\lambda b:\text{bool}, x_1:\tau, x_2:\tau\,(b\,\tau\,x_1\,x_2))\;\text{True}\;M_2\;M_3$$

$$\to^* \text{true} \; \tau \; M_2 \; M_3$$

$$\to^* (\lambda x_1:\tau, x_2:\tau(x_1)) \; M_2 \; M_3$$

$$\to^* M_2$$

$$\to^* N_2$$

FACT: $\text{True} \triangleq \Lambda\alpha\,(\lambda x_1, x_2 : \alpha\,(x_1))$

$\text{False} \triangleq \Lambda\alpha\,(\lambda x_1, x_2 : \alpha\,(x_2))$

are the only closed expressions in β-normal form of type $\text{bool} \triangleq \forall\alpha\,(\alpha \to (\alpha \to \alpha))$.

Polymorphic lists

$$\alpha\ \mathit{list} \stackrel{\text{def}}{=} \forall\, \alpha'\, (\alpha' \to (\alpha \to \alpha' \to \alpha') \to \alpha')$$

$$\mathit{Nil} \stackrel{\text{def}}{=} \Lambda\, \alpha, \alpha'\, (\lambda\, x' : \alpha', f : \alpha \to \alpha' \to \alpha'\, (x'))$$

$$\begin{aligned} \mathit{Cons} \stackrel{\text{def}}{=} {} & \Lambda\alpha(\lambda x : \alpha, \ell : \alpha\, \mathit{list}(\Lambda\alpha'(\\ & \qquad \lambda x' : \alpha', f : \alpha \to \alpha' \to \alpha'(\\ & \qquad\quad f\, x\, (\ell\, \alpha'\, x'\, f))))) \end{aligned}$$

$$\text{Bool} = \forall\alpha.\ \underbrace{\alpha}_{\text{False}} \to \underbrace{\alpha}_{\text{True}} \to \alpha.$$

$$\alpha\ \text{List} = \forall \underbrace{\alpha'}_{}.\ \underbrace{\alpha'}_{\text{Nil}} \to \underbrace{(\alpha \to \alpha' \to \alpha')}_{\text{Cons}} \to \underbrace{\alpha'}_{}$$

$$\text{data } \alpha\ \text{List} = \text{Nil} \mid \text{Cons of } \alpha * (\alpha\ \text{List})$$

$$\text{Nil} : \alpha\ \text{List}$$

$$\text{Cons} : \alpha * \alpha\ \text{List} \to \alpha\ \text{List}$$

$$\text{curry Cons} : \alpha \to (\alpha\ \text{List} \to \alpha\ \text{List})$$

$$\text{Nil} : \forall\alpha.\ \alpha\ \text{list}$$

$$\text{Cons} : \forall\alpha.\ \underbrace{\alpha}_{x} \to \underbrace{\alpha\ \text{List}}_{L} \to \underbrace{\alpha\ \text{List}}_{\forall\alpha'.\ \underbrace{\alpha'}_{x'} \to \underbrace{(\alpha \to \alpha' \to \alpha')}_{f} \to \alpha'}$$

$$\Downarrow$$

$$\underbrace{f\ \underset{\alpha}{\underset{\uparrow}{x}}\ (\underbrace{L\ \alpha'\ x'\ f}_{\alpha'})}_{\alpha'}$$