λ-bound variables in ML cannot be used polymorphically within a function abstraction

E.g. $\lambda f((f\ \mathtt{true}) :: (f\ \mathtt{nil}))$ and $\lambda f(f\ f)$ are not typeable in the ML type system.

Syntactically, because in rule

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

the abstracted variable has to be assigned a *trivial* type scheme (recall $x : \tau_1$ stands for $x : \forall\,\{\,\}\,(\tau_1)$).

Semantically, because $\forall\, A\, (\tau_1) \to \tau_2$ is not semantically equivalent to an ML type when $A \neq \{\,\}$.

$$\dfrac{\dfrac{\dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_4}\,(\text{var}>) \qquad \dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_5}\,(\text{var}>)}{\{f:\forall\{\}\tau_2\}\vdash ff:\tau_3}\,(\text{app})}{\{\}\vdash \lambda f(ff):\tau_1}\,(\text{lam})$$

$$\dfrac{\dfrac{\dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_4}\,{\scriptstyle(var>)}\ (1) \qquad \dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_5}\,{\scriptstyle(var>)}\ (2)}{\{f:\forall\{\}\tau_2\}\vdash ff:\tau_3}\,{\scriptstyle(app)}\ (3)}{\{\}\vdash \lambda f(ff):\tau_1}\,{\scriptstyle(lam)}\ (4)$$

(1) $\forall\{\}\tau_2 > \tau_4$

(2) $\forall\{\}\tau_2 > \tau_5$

(3) $\tau_4 = \tau_5 \to \tau_3$

(4) $\tau_1 = \tau_2 \to \tau_3$

$$
\text{4}\;\dfrac{\text{3}\;\dfrac{\text{1}\;\dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_4}\,(\text{var}>) \qquad \text{2}\;\dfrac{}{\{f:\forall\{\}\tau_2\}\vdash f:\tau_5}\,(\text{var}>)}{\{f:\forall\{\}\tau_2\}\vdash ff:\tau_3}\,(\text{app})}{\{\}\vdash \lambda f(ff):\tau_1}\,(\text{lam})
$$

1 $\forall\{\}\tau_2 > \tau_4$ so $\tau_2 = \tau_4$

2 $\forall\{\}\tau_2 > \tau_5$ so $\tau_2 = \tau_5$

3 $\tau_4 = \tau_5 \to \tau_3$

4 $\tau_1 = \tau_2 \to \tau_3$

$$
\dfrac{
  \dfrac{①}{\{f : \forall\{\}\tau_2\} \vdash f : \tau_4}\,(\mathrm{var}\succ)
  \qquad
  \dfrac{②}{\{f : \forall\{\}\tau_2\} \vdash f : \tau_5}\,(\mathrm{var}\succ)
}{
  \dfrac{③ \quad \{f : \forall\{\}\tau_2\} \vdash f f : \tau_3}{\{\} \vdash \lambda f(f f) : \tau_1}\,④\,(\mathrm{lam})
}\,(\mathrm{app})
$$

① $\forall\{\}\tau_2 \succ \tau_4$ so $\tau_2 = \tau_4$

② $\forall\{\}\tau_2 \succ \tau_5$ so $\tau_2 = \tau_5$

③ $\tau_4 = \tau_5 \to \tau_3$

④ $\tau_1 = \tau_2 \to \tau_3$

$\Rightarrow \tau_2 = \tau_2 \to \tau_3$

No such τ_2 & τ_3 can exist (by counting $\to$ symbols on LHS & RHS of the equation).

Monomorphic types ...

$$\tau ::= \alpha \mid bool \mid \tau \to \tau \mid \tau\ list$$

... and *type schemes*

$$\sigma ::= \tau \mid \forall\, \alpha\ (\sigma)$$

Polymorphic types

$$\pi ::= \alpha \mid bool \mid \pi \to \pi \mid \pi\ list \mid \forall\, \alpha\ (\pi)$$

E.g. $\alpha \to \alpha'$ is a type, $\forall\, \alpha\ (\alpha \to \alpha')$ is a type scheme and a polymorphic type (but not a monomorphic type), $\forall\, \alpha\ (\alpha) \to \alpha'$ is a polymorphic type, but not a type scheme.

Identity, Generalisation and Specialisation

(**id**) $\Gamma \vdash x : \pi$ if $(x : \pi) \in \Gamma$

(**gen**) $$\frac{\Gamma \vdash M : \pi}{\Gamma \vdash M : \forall \alpha\, (\pi)}$$ if $\alpha \notin ftv(\Gamma)$

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

[Example 4.1.1, p41]

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{f:\forall\alpha_1(\alpha_1) \vdash f:\forall\alpha_1(\alpha_1)}\,(\mathrm{id})
}{f:\forall\alpha_1(\alpha_1) \vdash f:\alpha_2 \to \alpha_2}\,(\mathrm{spec})
\qquad
\dfrac{
\dfrac{}{f:\forall\alpha_1(\alpha_1) \vdash f:\forall\alpha_1(\alpha_1)}\,(\mathrm{id})
}{f:\forall\alpha_1(\alpha_1) \vdash f:\alpha_2}\,(\mathrm{spec})
}{f:\forall\alpha_1(\alpha_1) \vdash ff:\alpha_2}\,(\mathrm{app})
}{
\dfrac{}{}
}
$$

$$
\dfrac{
\dfrac{f:\forall\alpha_1(\alpha_1) \vdash ff:\alpha_2}{f:\forall\alpha_1(\alpha_1) \vdash ff:\forall\alpha_2(\alpha_2)}\,(\mathrm{gen})
}{\{\} \vdash \lambda f(ff) : \underbrace{\forall\alpha_1(\alpha_1) \to \forall\alpha_2(\alpha_2)}_{\displaystyle \parallel \atop \displaystyle \forall\alpha(\alpha) \to \forall\alpha(\alpha)}}\,(\mathrm{fn})
$$

Fact (see Wells 1994):

For the modified ML type system with polymorphic types and $(\mathbf{var}\succ)$ replaced by the axiom and rules on Slide 39, *the type checking and typeability problems* (cf. Slide 7) *are equivalent and undecidable.*

Explicitly versus implicitly typed languages

Implicit: little or no type information is included in program phrases and typings have to be inferred (ideally, entirely at compile-time). (E.g. Standard ML.)

Explicit: most, if not all, types for phrases are explicitly part of the syntax. (E.g. Java.)

E.g. self application function of type $\forall\,\alpha\,(\alpha) \to \forall\,\alpha\,(\alpha)$ (cf. Example 4.1.1)

Implicitly typed version: $\lambda\,f\,(f\ f)$

Explicitly type version: $\lambda\,f : \forall\,\alpha_1\,(\alpha_1)\,(\Lambda\,\alpha_2\,(f(\alpha_2 \to \alpha_2)(f\,\alpha_2)))$

PLC syntax

Types

τ	$::=$	α	type variable
	$\mid$	$\tau \to \tau$	function type
	$\mid$	$\forall\, \alpha\, (\tau)$	$\forall$-type

Expressions

M	$::=$	x	variable
	$\mid$	$\lambda\, x : \tau\, (M)$	function abstraction
	$\mid$	$M\ M$	function application
	$\mid$	$\Lambda\, \alpha\, (M)$	type generalisation
	$\mid$	$M\ \tau$	type specialisation

(α and x range over fixed, countably infinite sets $\mathbf{TyVar}$ and $\mathbf{Var}$ respectively.)

PLC typing judgement

takes the form $\boxed{\Gamma \vdash M : \tau}$ where

- the *typing environment* Γ is a finite function from variables to PLC types.
  (We write $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ to indicate that Γ has domain of definition $dom(\Gamma) = \{x_1, \ldots, x_n\}$ and maps each x_i to the PLC type τ_i for $i = 1..n$.)
- M is a PLC expression
- τ is a PLC type.

PLC type system

(var) $\Gamma \vdash x : \tau$ if $(x : \tau) \in \Gamma$

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

(app) $$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\, M_2 : \tau_2}$$

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

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

[Example 4.2.6, p49]

$$\dfrac{}{f : \forall\alpha_1(\alpha_1) \vdash f : \forall\alpha_1(\alpha_1)}\,(\mathrm{var}) \qquad \dfrac{}{f : \forall\alpha_1(\alpha_1) \vdash f : \forall\alpha_1(\alpha_1)}\,(\mathrm{var})$$

$$\dfrac{f : \forall\alpha_1(\alpha_1) \vdash f : \forall\alpha_1(\alpha_1)}{f : \forall\alpha_1(\alpha_1) \vdash f(\alpha_2 \to \alpha_2) : \alpha_2 \to \alpha_2}\,(\mathrm{spec}) \qquad \dfrac{f : \forall\alpha_1(\alpha_1) \vdash f : \forall\alpha_1(\alpha_1)}{f : \forall\alpha_1(\alpha_1) \vdash f\alpha_2 : \alpha_2}\,(\mathrm{spec})$$

$$\dfrac{f : \forall\alpha_1(\alpha_1) \vdash f(\alpha_2 \to \alpha_2) : \alpha_2 \to \alpha_2 \qquad f : \forall\alpha_1(\alpha_1) \vdash f\alpha_2 : \alpha_2}{f : \forall\alpha_1(\alpha_1) \vdash f(\alpha_2 \to \alpha_2)(f\alpha_2) : \alpha_2}\,(\mathrm{app})$$

$$\dfrac{f : \forall\alpha_1(\alpha_1) \vdash f(\alpha_2 \to \alpha_2)(f\alpha_2) : \alpha_2}{f : \forall\alpha_1(\alpha_1) \vdash \Lambda\alpha_2(f(\alpha_2 \to \alpha_2)(f\alpha_2)) : \forall\alpha_2(\alpha_2)}\,(\mathrm{gen})$$

$$\dfrac{f : \forall\alpha_1(\alpha_1) \vdash \Lambda\alpha_2(f(\alpha_2 \to \alpha_2)(f\alpha_2)) : \forall\alpha_2(\alpha_2)}{: \{\} \vdash \lambda f : \forall\alpha_1(\alpha_1)\big(\Lambda\alpha_2(f(\alpha_2 \to \alpha_2)(f\alpha_2))\big) : \underbrace{\forall\alpha_1(\alpha_1) \to \forall\alpha_2(\alpha_2)}_{\substack{\| \\ \forall\alpha(\alpha) \to \forall\alpha(\alpha)}}}\,(\mathrm{fn})$$

[Example 4.2.7
p 49]

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \tau'''}\,(\text{var})}{\{\} \vdash \lambda x : \alpha\,(x) : \tau''}\,(\text{fn})}{\{\} \vdash (\lambda x : \alpha\,(x))\,\alpha : \tau'}\,(\text{spec})}{\{\} \vdash \Lambda \alpha\,((\lambda x : \alpha\,(x))\,\alpha) : \tau}\,(\text{gen})
$$

$$\text{C1}\quad \frac{}{x : \alpha \vdash x : \tau'''}\,(\text{var})$$

$$\text{C2}\quad \frac{x : \alpha \vdash x : \tau'''}{\{\} \vdash \lambda x : \alpha\,(x) : \tau''}\,(\text{fn})$$

$$\text{C3}\quad \frac{\{\} \vdash \lambda x : \alpha\,(x) : \tau''}{\{\} \vdash (\lambda x : \alpha\,(x))\,\alpha : \tau'}\,(\text{spec})$$

$$\text{C4}\quad \frac{\{\} \vdash (\lambda x : \alpha\,(x))\,\alpha : \tau'}{\{\} \vdash \Lambda\alpha\,((\lambda x : \alpha\,(x))\,\alpha) : \tau}\,(\text{gen})$$

C1 : $\tau''' = \alpha$

C2 : $\tau'' = \alpha \to \tau'''$

C3 : $\tau' = \tau^{iv}[\alpha/\alpha']$, $\tau'' = \forall\alpha'\,(\tau^{iv})$

C4 : $\tau = \forall\alpha\,(\tau')$

$$\text{C4}\quad \dfrac{}{x : \alpha \vdash x : \tau'''}\ (\text{var})$$

$$\text{C3}\quad \dfrac{x : \alpha \vdash x : \tau'''}{\{\} \vdash \lambda x : \alpha\,(x) : \tau''}\ (\text{fn})$$

$$\text{C2}\quad \dfrac{\{\} \vdash \lambda x : \alpha\,(x) : \tau''}{\{\} \vdash (\lambda x : \alpha\,(x))\,\alpha : \tau'}\ (\text{spec})$$

$$\text{C1}\quad \dfrac{\{\} \vdash (\lambda x : \alpha\,(x))\,\alpha : \tau'}{\{\} \vdash \Lambda\alpha\,((\lambda x : \alpha\,(x))\,\alpha) : \tau}\ (\text{gen})$$

$$\text{C1} : \quad \tau''' = \alpha$$

$$\text{C2} : \quad \tau'' = \alpha \to \tau'''$$

$$\text{C3} : \quad \tau' = \tau^{iv}[\alpha/\alpha'],\quad \tau'' = \forall \alpha'(\tau^{iv})$$

$$\text{C4} : \quad \tau = \forall \alpha\,(\tau')$$

impossible

PLC binding forms

$$\forall \alpha(-) \qquad \lambda x : \tau(-) \qquad \Lambda \alpha(-)$$

Eg.

$$\lambda x : \forall \alpha(\beta)\Big(\Lambda \alpha\big(x(\alpha \to \beta)\big)\Big)$$

PLC binding forms

$$\forall\alpha(-) \qquad \lambda x : \tau(-) \qquad \Lambda\alpha(\ \ )$$

Eg.

$$\lambda x : \forall\beta(\underset{\text{free}}{\alpha})\,(\Lambda\alpha(x(\alpha \to \underset{\text{free}}{\beta})))$$

An incorrect "proof"

$$\dfrac{\dfrac{\overline{x_1 : \alpha, x_2 : \alpha \vdash x_2 : \alpha}\ (\mathrm{var})}{x_1 : \alpha \vdash \lambda\, x_2 : \alpha\, (x_2) : \alpha \to \alpha}\ (\mathrm{fn})}{x_1 : \alpha \vdash \Lambda\, \alpha\, (\lambda\, x_2 : \alpha\, (x_2)) : \forall\, \alpha\, (\alpha \to \alpha)}\ (\mathrm{wrong!})$$

An ~~I~~ncorrect “proof”

$(\alpha \neq \alpha_1)$

$$\dfrac{\dfrac{\dfrac{}{x_1 : \alpha, x_2 : \alpha_1 \vdash x_2 : \alpha_1}\ (\mathbf{var})}{x_1 : \alpha \vdash \lambda x_2 : \alpha_1\,(x_2) : \alpha_1 \to \alpha_1}\ (\mathbf{fn})}{x_1 : \alpha \vdash \underbrace{\Lambda\, \alpha_1\,(\lambda x_2 : \alpha_1\,(x_2))}_{=\ \Lambda \alpha (\lambda x_2 : \alpha (x_2))} : \underbrace{\forall\, \alpha_1\,(\alpha_1 \to \alpha_1)}_{=\ \forall \alpha (\alpha \to \alpha)}}\ (\cancel{\mathbf{wrong!}}\ \text{gen})$$

Decidability of the PLC typeability and type-checking problems

Theorem.
For each PLC typing problem, $\Gamma \vdash M : ?$, there is at most one PLC type τ for which $\Gamma \vdash M : \tau$ is provable. Moreover there is an algorithm, typ, which when given any $\Gamma \vdash M : ?$ as input, returns such a τ if it exists and $FAIL$s otherwise.

Corollary.
The PLC type checking problem is decidable: we can decide whether or not $\Gamma \vdash M : \tau$ is provable by checking whether $typ(\Gamma \vdash M : ?) = \tau$.

(N.B. equality of PLC types up to alpha-conversion is decidable.)

PLC type-checking algorithm, I

Variables:

$$typ(\Gamma, x : \tau \vdash x : ?) \stackrel{\text{def}}{=} \tau$$

Function abstractions:

$$typ(\Gamma \vdash \lambda\, x : \tau_1\,(M) : ?) \stackrel{\text{def}}{=}$$

$$\text{let } \tau_2 = typ(\Gamma, x : \tau_1 \vdash M : ?) \text{ in } \tau_1 \to \tau_2$$

Function applications:

$$typ(\Gamma \vdash M_1\, M_2 : ?) \stackrel{\text{def}}{=}$$

$$\text{let } \tau_1 = typ(\Gamma \vdash M_1 : ?) \text{ in}$$

$$\text{let } \tau_2 = typ(\Gamma \vdash M_2 : ?) \text{ in}$$

$$\begin{array}{lll} \text{case } \tau_1 \text{ of } & \tau \to \tau' & \mapsto \text{ if } \tau = \tau_2 \text{ then } \tau' \text{ else } \mathit{FAIL} \\ \qquad\qquad\;\; \mid & \_ & \mapsto \mathit{FAIL} \end{array}$$

PLC type-checking algorithm, II

Type generalisations:

$$\mathit{typ}(\Gamma \vdash \Lambda\,\alpha\,(M) : ?) \stackrel{\text{def}}{=}$$
$$\text{let } \tau = \mathit{typ}(\Gamma \vdash M : ?) \text{ in } \forall\,\alpha\,(\tau)$$

Type specialisations:

$$\mathit{typ}(\Gamma \vdash M\,\tau_2 : ?) \stackrel{\text{def}}{=}$$
$$\text{let } \tau = \mathit{typ}(\Gamma \vdash M : ?) \text{ in}$$
$$\begin{array}{lrcl} \text{case } \tau \text{ of} & \forall\,\alpha\,(\tau_1) & \mapsto & \tau_1[\tau_2/\alpha] \\ \mid & \_ & \mapsto & \mathit{FAIL} \end{array}$$