PLC type system

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

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

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

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

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

PLC operator association

$M_1 M_2 M_3$ means $(M_1 M_2) M_3$

$M_1 M_2 \tau$ means $(M_1 M_2) \tau$, etc.

$\forall \alpha_1, \alpha_2 (\tau)$ means $\forall \alpha_1 (\forall \alpha_2 (\tau))$

$\lambda x_1 : \tau_1, x_2 : \tau_2 (M)$ means $\lambda x_1 : \tau_1 (\lambda x_2 : \tau_2 (M))$

$\Lambda \alpha_1, \alpha_2 (M)$ means $\Lambda \alpha_1 (\Lambda \alpha_2 (M))$

Datatypes in PLC [Sect. 4.4]

- define a suitable PLC type for the data
- define suitable PLC expressions for values & operations on the data
- show PLC expressions have correct typings & computational behaviour

need to give PLC an operational semantics

Functions on types

In PLC, $\boxed{\Lambda\alpha\,(M)}$ is an anonymous notation for the function F mapping each type τ to the value of $M[\tau/\alpha]$ (of some particular type).

Functions on types

In PLC, $\boxed{\Lambda\alpha\,(M)}$ is an anonymous notation for the function F mapping each type τ to the value of $M[\tau/\alpha]$ (of some particular type).

$\boxed{F\,\tau}$ denotes the result of applying such a function to a type.

Functions on types

In PLC, $\boxed{\Lambda\alpha\,(M)}$ is an anonymous notation for the function F mapping each type τ to the value of $M[\tau/\alpha]$ (of some particular type).

$\boxed{F\,\tau}$ denotes the result of applying such a function to a type.

Computation in PLC involves beta-reduction for such functions on types

$$(\Lambda\alpha\,(M))\,\tau \to M[\tau/\alpha]$$

Functions on types

In PLC, $\boxed{\Lambda\alpha\,(M)}$ is an anonymous notation for the function F mapping each type τ to the value of $M[\tau/\alpha]$ (of some particular type).

$\boxed{F\,\tau}$ denotes the result of applying such a function to a type.

Computation in PLC involves beta-reduction for such functions on types

$$(\Lambda\alpha\,(M))\,\tau \to M[\tau/\alpha]$$

as well as the usual form of beta-reduction from λ-calculus

$$(\lambda x : \tau\,(M_1))\,M_2 \to M_1[M_2/x]$$

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:

$$(\lambda x : \tau\,(M_1))\, M_2 \to M_1[M_2/x]$$

$$(\Lambda \alpha\,(M))\, \tau \to M[\tau/\alpha]$$

$M_1[M_2/x]$ = result of substituting M_2 for all free occurrences of x in M_1 (avoiding capture of free vars & tyvars in M_2 by binders in M_1)

$M[\tau/\alpha]$ = result of substituting τ for all free occurrences of α in M (avoiding capture)

[p44]

$$(\lambda x : \alpha_1 \to \alpha_1\ (x\, y))\ ((\Lambda \alpha_2 (\lambda z : \alpha_2 (z)))(\alpha_1 \to \alpha_1))$$

[p44]

$$(\lambda x:\alpha_1\to\alpha_1\,(x\,y))\ \boxed{(\Lambda\alpha_2\,(\lambda z:\alpha_2\,(z)))(\alpha_1\to\alpha_1)}$$

$$(\lambda z:\alpha_1\to\alpha_1\,(z))$$

[p44]

$$(\lambda x:\alpha_1\to\alpha_1\,(x\,y))\ \boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}$$

$$(\lambda x:\alpha_1\to\alpha_1(x\,y))(\lambda z:\alpha_1\to\alpha_1(z))$$

[p44]

$$(\lambda x:\alpha_1\to\alpha_1\,(x\,y))\ \boxed{(\Lambda\alpha_2\,(\lambda z:\alpha_2\,(z)))(\alpha_1\to\alpha_1)}$$

$$\boxed{(\lambda x:\alpha_1\to\alpha_1(x\,y))(\lambda z:\alpha_1\to\alpha_1(z))}$$

$$(\lambda z:\alpha_1\to\alpha_1(z))\,y$$

[p44]

$$\boxed{(\lambda x:\alpha_1\to\alpha_1\,(x\,y))\ \boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}}$$

$$\boxed{(\lambda x:\alpha_1\to\alpha_1(x\,y))(\lambda z:\alpha_1\to\alpha_1(z))}$$

$$(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)\ y$$

$$(\lambda z:\alpha_1\to\alpha_1(z))\,y$$

[p44]

$$
\begin{tikzcd}[column sep=small]
& \boxed{(\lambda x:\alpha_1\to\alpha_1\,(xy))\ \boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}} \arrow[dl] \arrow[dr] & \\
\boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}\ y \arrow[dr] & & \boxed{(\lambda x:\alpha_1\to\alpha_1(xy))(\lambda z:\alpha_1\to\alpha_1(z))} \arrow[dl] \\
& (\lambda z:\alpha_1\to\alpha_1(z))\,y &
\end{tikzcd}
$$

[p44]

$$
\begin{tikzcd}[column sep=tiny]
& \boxed{(\lambda x:\alpha_1\to\alpha_1\,(x y))\;\boxed{(\Lambda\alpha_2\,(\lambda z:\alpha_2\,(z)))(\alpha_1\to\alpha_1)}} \arrow[dl] \arrow[dr] & \\
\boxed{(\Lambda\alpha_2\,(\lambda z:\alpha_2\,(z)))(\alpha_1\to\alpha_1)}\; y \arrow[dr] & & \boxed{(\lambda x:\alpha_1\to\alpha_1\,(x y))\,(\lambda z:\alpha_1\to\alpha_1\,(z))} \arrow[dl] \\
& (\lambda z:\alpha_1\to\alpha_1\,(z))\, y \arrow[d] & \\
& y &
\end{tikzcd}
$$

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.

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).

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: if $\Gamma \vdash M : \tau$ & $M \to M'$, then $\Gamma \vdash M' : \tau$

$$\dfrac{\dfrac{\Gamma, x : \tau' \vdash M : \tau}{\Gamma \vdash \lambda x : \tau'(M) : \tau' \to \tau} \qquad \Gamma \vdash M' : \tau'}{\Gamma \vdash (\lambda x : \tau'(M))\, M' : \tau}$$

$$\dfrac{\dfrac{\Gamma, x : \tau' \vdash M : \tau}{\Gamma \vdash \lambda x : \tau'(M) : \tau' \to \tau} \qquad \Gamma \vdash M' : \tau'}{\Gamma \vdash (\lambda x : \tau'(M))\, M' : \tau}$$

$$\big\downarrow_{\beta}$$

$$M[M'/x]$$

$$\dfrac{\dfrac{\Gamma, x:\tau' \vdash M : \tau}{\Gamma \vdash \lambda x:\tau'(M) : \tau' \to \tau} \qquad \Gamma \vdash M' : \tau'}{\Gamma \vdash (\lambda x:\tau'(M))\,M' : \tau}$$

$$\downarrow_\beta$$

$$M[M'/x] \leftarrow$$

to see that this has type τ, need to prove a <u>Substitution Lemma</u>

If $\Gamma, x : \tau' \vdash M : \tau$

and $\Gamma \vdash M' : \tau'$

then

$$\Gamma \vdash M[M'/x] : \tau$$

Substitution Lemma

(proved by induction on structure of M)

Subject reduction: if $\Gamma \vdash M : \tau$ & $M \to M'$, then $\Gamma \vdash M' : \tau$

$$\dfrac{\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash \Lambda\alpha(M) : \forall\alpha(\tau)}\ \alpha \notin ftv(\Gamma)}{\Gamma \vdash (\Lambda\alpha(M))\tau' : \tau[\tau'/\alpha]}$$

$$\dfrac{\dfrac{\Gamma \vdash M : \tau}{\Gamma \vdash \Lambda\alpha(M) : \forall\alpha(\tau)}\ \alpha \notin ftv(\Gamma)}{\Gamma \vdash (\Lambda\alpha(M))\tau' : \tau[\tau'/\alpha]}$$

$$(\Lambda\alpha(M))\tau' \to_\beta M[\tau'/\alpha]$$

to see ~~that~~ ~~this~~ has type $\tau[\tau'/\alpha]$, need to prove a substitution lemma

If $\Gamma \vdash M : \tau$ & $\alpha \notin ftv(\Gamma)$

then

$\Gamma \vdash M[\tau'/\alpha] : \tau[\tau'/\alpha]$

(proved by induction of structure of M)

Substitution Lemma

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 $M_2 \to^* M'$.

[p44]

$$
\begin{tikzcd}[column sep=small]
& \boxed{(\lambda x:\alpha_1\to\alpha_1\,(xy))\ \boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}} \arrow[dl] \arrow[dr] & \\
\boxed{(\Lambda\alpha_2(\lambda z:\alpha_2(z)))(\alpha_1\to\alpha_1)}\ y \arrow[dr] & & \boxed{(\lambda x:\alpha_1\to\alpha_1(xy))(\lambda z:\alpha_1\to\alpha_1(z))} \arrow[dl] \\
& (\lambda z:\alpha_1\to\alpha_1(z))\,y &
\end{tikzcd}
$$

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.

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.

$\Omega \triangleq (\lambda x : \alpha\,(x\,x))(\lambda x : \alpha\,(x\,x))$ satisfies $\Omega \to \Omega \to \Omega \to \cdots$

but it's not typeable (nor is the fixpoint combinator, Y)

Theorem 15: [p46]

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

Existence: start from M & reduce any old way ... must eventually stop by SN

Uniqueness: if

$$
\begin{tikzcd}
 & N_1 \not\to \\
M \arrow[ur, "*"] \arrow[dr, "*"'] & \\
 & N_2 \not\to
\end{tikzcd}
$$

Theorem 15 [p46]

$\boxed{\text{Church Rosser (CR) + Strong Normalization (SN)} \Rightarrow \text{exist unique beta-normal forms for typeable PLC expressions}}$

Existence: start from M & reduce any old way ... must eventually stop by SN

Uniqueness: if

$$\begin{tikzcd} & N_1 \arrow[dr, "*"] & \\ M \arrow[ur, "*"] \arrow[dr, "*"'] & & M' \\ & N_2 \arrow[ur, "*"'] & \end{tikzcd}$$

by CR

Theorem 15 [p46]

> Church Rosser (CR) + Strong Normalization (SN)
> $\Rightarrow$ exist unique beta-normal forms for typeable PLC expressions

Existence: start from M & reduce any old way… must eventually stop by SN

Uniqueness: if

$$\begin{tikzcd}
 & N_1 \arrow[dr, "*"] \arrow[r, "/" marking] & {} \\
M \arrow[ur, "*"] \arrow[dr, "*"'] & & M' \\
 & N_2 \arrow[ur, "*"'] \arrow[r, "/" marking] & {}
\end{tikzcd}$$

so

$$N_1 = M' = N_2$$

α-equiv

PLC beta-conversion, $=_\beta$

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

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 $\to$ or $\leftarrow$, i.e. a beta-reduction in one direction or the other.

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 $\to$ 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.)

So $=_\beta$ is the smallest equivalence relation containing $\to$

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

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

Datatypes in PLC [Sect. 4.4]

- define a suitable PLC type for the data
- define suitable PLC expressions for values & operations on the data
- show PLC expressions have correct typings & computational behaviour

Polymorphic booleans

$$\boldsymbol{bool} \triangleq \forall\alpha\,(\alpha \to (\alpha \to \alpha))$$

Polymorphic booleans

$$\textit{bool} \triangleq \forall \alpha\, (\alpha \to (\alpha \to \alpha))$$

In ML/Haskell/Scala/.... have

datatype bool = True | False

and each B : bool gives us a polymorphic function $\lambda x, y.\ \text{if } B \text{ then } x \text{ else } y : \forall \alpha (\alpha \to \alpha \to \alpha)$

Polymorphic booleans

$$\mathit{bool} \triangleq \forall \alpha\, (\alpha \to (\alpha \to \alpha))$$

In ML/Haskell/Scala/.... have

datatype bool = True | False

and each $B : bool$ gives us a polymorphic function $\lambda x, y.\ \text{if } B \text{ then } x \text{ else } y : \forall \alpha (\alpha \to \alpha \to \alpha)$

IDEA: identify Booleans with expressions of this type.

Polymorphic booleans

$$bool \triangleq \forall\alpha\,(\alpha \to (\alpha \to \alpha))$$

$$True \triangleq \Lambda\alpha\,(\lambda x_1 : \alpha, x_2 : \alpha\,(x_1))$$

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

$\{\} \vdash True : bool$

$\{\} \vdash False : bool$

Polymorphic booleans

$$bool \triangleq \forall\alpha\,(\alpha \to (\alpha \to \alpha))$$

$$True \triangleq \Lambda\alpha\,(\lambda x_1 : \alpha, x_2 : \alpha\,(x_1))$$

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

$$if \triangleq \Lambda\alpha\,(\lambda b : bool, x_1 : \alpha, x_2 : \alpha\,(b\,\alpha\,x_1\,x_2))$$

$$\{\} \vdash if : \forall\alpha\,(bool \to (\alpha \to (\alpha \to \alpha)))$$

If $\begin{cases} M_1 \to^* \text{True} \\ M_2 \to^* N \end{cases}$, then

$$\text{if}\,\tau\; M_1\; M_2\; M_3 \to^* \text{if}\,\tau\; \text{True}\; M_2\; M_3$$

If $\begin{cases} M_1 \to^* \mathrm{True} \\ M_2 \to^* N \end{cases}$, then

$$\mathrm{if}\ \tau\ M_1\ M_2\ M_3 \to^* \mathrm{if}\ \tau\ \mathrm{True}\ M_2\ M_3$$

$$= \Lambda\alpha(\ldots)\ \tau\ \mathrm{True}\ M_2\ M_3$$

If $\begin{cases} M_1 \to^* \mathrm{True} \\ M_2 \to^* N \end{cases}$, then

$$\mathrm{if}\ \tau\ M_1\ M_2\ M_3 \to^* \mathrm{if}\ \tau\ \mathrm{True}\ M_2\ M_3$$

$$\| $$

$$\boxed{\Lambda\alpha(\dots)\ \tau}\ \mathrm{True}\ M_2\ M_3$$

$$\downarrow$$

$$(\lambda b:\mathrm{bool}, x_1:\tau, x_2:\tau\,(b\ \tau\ x_1\ x_2))\ \mathrm{True}\ M_2\ M_3$$

If $\begin{cases} M_1 \to^* \mathrm{True} \\ M_2 \to^* N \end{cases}$, then

$$\begin{aligned}
\mathrm{if}\ \tau\ M_1\ M_2\ M_3 &\to^* \mathrm{if}\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&= \Lambda\alpha(\dots)\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&\to (\lambda b{:}\mathrm{bool}, x_1{:}\tau, x_2{:}\tau\ (b\ \tau\ x_1\ x_2))\ \mathrm{True}\ M_2\ M_3 \\
&\to^* \mathrm{True}\ \tau\ M_2\ M_3
\end{aligned}$$

If $\begin{cases} M_1 \to^* \mathrm{True} \\ M_2 \to^* N \end{cases}$, then

$$
\begin{aligned}
\mathrm{if}\ \tau\ M_1\ M_2\ M_3 &\to^* \mathrm{if}\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&= \Lambda\alpha(\ldots)\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&\to (\lambda b{:}\mathrm{bool}, x_1{:}\tau, x_2{:}\tau\,(b\ \tau\ x_1\ x_2))\ \mathrm{True}\ M_2\ M_3 \\
&\to^* \mathrm{True}\ \tau\ M_2\ M_3 \\
&= \Lambda\alpha\,(\lambda x_1{:}\alpha, x_2{:}\alpha\,(x_1))\ \tau\ M_2\ M_3
\end{aligned}
$$

If $\begin{cases} M_1 \to^* \mathrm{True} \\ M_2 \to^* N \end{cases}$, then

$$
\begin{aligned}
\mathrm{if}\ \tau\ M_1\ M_2\ M_3 &\to^* \mathrm{if}\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&= \Lambda\alpha(\dots)\ \tau\ \mathrm{True}\ M_2\ M_3 \\
&\to (\lambda b{:}\mathrm{bool}, x_1{:}\tau, x_2{:}\tau\,(b\ \tau\ x_1\ x_2))\ \mathrm{True}\ M_2\ M_3 \\
&\to^* \mathrm{True}\ \tau\ M_2\ M_3 \\
&= \Lambda\alpha\,(\lambda x_1{:}\alpha, x_2{:}\alpha\,(x_1))\ \tau\ M_2\ M_3 \\
&\to^* M_2 \\
&\to^* N
\end{aligned}
$$

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 $\text{bool} \triangleq \forall\alpha\,(\alpha \to (\alpha \to \alpha))$.