PLC type system

\[(\text{var}) \quad \Gamma \vdash x : \tau \quad \text{if } (x : \tau) \in \Gamma\]

\[(\text{fn}) \quad \Gamma, x : \tau_1 \vdash M : \tau_2 \quad \text{if } x \notin \text{dom}(\Gamma)\]

\[(\text{app}) \quad \Gamma \vdash M : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash M' : \tau_1 \quad \Gamma \vdash MM' : \tau_2\]

\[(\text{gen}) \quad \Gamma \vdash M : \tau \quad \text{if } \alpha \notin \text{ftv}(\Gamma)\]

\[(\text{spec}) \quad \Gamma \vdash M : \forall \alpha (\tau_1) \quad \Gamma \vdash M \tau_2 : \tau_1[\tau_2/\alpha]\]
Datatypes in PLC

- 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, $\Lambda\alpha (M)$ is an anonymous notation for the function $F$ mapping each type $\tau$ to the value of $M[\tau/\alpha]$ (of some particular type).
Functions on types

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

$F \tau$ denotes the result of applying such a function to a type.
Functions on types

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

$F \tau$ denotes the result of applying such a function to a type.

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

$$(\Lambda \alpha (M)) \tau \rightarrow M[\tau/\alpha]$$
Functions on types

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

$F \tau$ denotes the result of applying such a function to a type.

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

$$(\Lambda \alpha (M)) \tau \rightarrow M[\tau/\alpha]$$

as well as the usual form of beta-reduction from $\lambda$-calculus

$$(\lambda x : \tau (M_1)) \ M_2 \rightarrow M_1[M_2/x]$$
Beta-reduction of PLC expressions

\( M \) beta-reduces to \( M' \) in one step, \( M \rightarrow M' \) 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:

\[
(\lambda x : \tau (M_1)) M_2 \rightarrow M_1[M_2/x]
\]

\[
(\Lambda \alpha (M)) \tau \rightarrow M[\tau/\alpha]
\]

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

\( M[\tau/\alpha] \) = result of substituting \( \tau \) for all free occurrences of \( \alpha \) in \( M \) (avoiding capture)
\[(\lambda x : \alpha_1 \to \alpha_1 \, (x \, y)) \, ((\forall \alpha_2 \, (\lambda z : \alpha_2 \, (z))) \, (\alpha_1 \to \alpha_1))\]
\[(\lambda x : \alpha_1 \to \alpha_1 (xy)) \quad (\forall \alpha_2 (\lambda z : \alpha_2 (z)))(\alpha_1 \to \alpha_1)\]

\[(\lambda z : \alpha_1 \to \alpha_1 (z))\]
\[(\lambda x : \alpha_1 \to \alpha_1 (x y)) (\Lambda \alpha_2 (\lambda z : \alpha_2 (z)) (\alpha_1 \to \alpha_1))\]
(\lambda x : \alpha_1 \to \alpha_1 (xy)) \ (\lambda z : \alpha_2 (z)) \ (\alpha_1 \to \alpha_1)

(\lambda x : \alpha_1 \to \alpha_1 (xy)) \ (\lambda z : \alpha_1 \to \alpha_1 (z))

(\lambda z : \alpha_1 \to \alpha_1 (z)) y
\((\lambda x : \alpha_1 \rightarrow \alpha_1 (xy)) \ (\forall \alpha_2 (\lambda z : \alpha_2 (z))) (\alpha_1 \rightarrow \alpha_1)\)

\((\lambda x : \alpha_1 \rightarrow \alpha_1 (xy)) (\lambda z : \alpha_1 \rightarrow \alpha_1 (z)) \ y\)

\((\forall \alpha_2 (\lambda z : \alpha_2 (z))) (\alpha_1 \rightarrow \alpha_1) \ y\)
\[(\lambda x: \alpha_1 \to \alpha_1 (xy)) \quad (\Lambda \alpha_2 (\lambda z: \alpha_2 (z))) \quad (\alpha_1 \to \alpha_1)\]

\[(\lambda x: \alpha_1 \to \alpha_1 (xy)) \quad (\lambda z: \alpha_1 \to \alpha_1 (z))\]

\[(\Lambda \alpha_2 (\lambda z: \alpha_2 (z))) \quad (\alpha_1 \to \alpha_1)\quad y\]

\[(\lambda z: \alpha_1 \to \alpha_1 (z)) \quad y\]
\[(\lambda x: \alpha_1 \to \alpha_1 (xy)) \quad \left( \forall \alpha_2 (\lambda z: \alpha_2 (z)) \right) (\alpha_1 \to \alpha_1) \]

\[(\lambda x: \alpha_1 \to \alpha_1 (xy)) (\lambda z: \alpha_1 \to \alpha_1 (z)) \quad y \]

\[(\forall \alpha_2 (\lambda z: \alpha_2 (z)))(\alpha_1 \to \alpha_1) \quad y \]

\[(\lambda z: \alpha_1 \to \alpha_1 (z)) y \quad y \]
Beta-reduction of PLC expressions

\( M \) beta-reduces to \( M' \) in one step, \( M \rightarrow M' \) 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:

\[
(\lambda x : \tau (M_1)) M_2 \rightarrow M_1[M_2/x]
\]

\[
(\Lambda \alpha (M)) \tau \rightarrow M[\tau/\alpha]
\]

\( M \rightarrow^* M' \) indicates a chain of finitely\(^\dagger\) many beta-reductions.
Beta-reduction of PLC expressions

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

\[
(\lambda x : \tau (M_1)) \ M_2 \rightarrow \ M_1[M_2/x] \\
(\Lambda \alpha (M)) \ \tau \rightarrow \ M[\tau/\alpha]
\]

\( M \rightarrow^* M' \) indicates a chain of finitely\(^\dagger\) many beta-reductions.

(\(\dagger \) possibly zero – which just means \( M \) and \( M' \) are alpha-convertible).
Beta-reduction of PLC expressions

\( M \) beta-reduces to \( M' \) in one step, \( M \rightarrow M' \) 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:

\[
(\lambda x : \tau (M_1)) M_2 \rightarrow M_1[M_2/x]
\]

\[
(\Lambda \alpha (M)) \tau \rightarrow M[\tau/\alpha]
\]

\( M \rightarrow^* M' \) indicates a chain of finitely\(^\dagger\) many beta-reductions.

(\( \dagger \) 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 \( M \rightarrow M' \), then \( \Gamma \vdash M' : \tau \) is also a provable typing.
Subject reduction: if $\Gamma \vdash w : 2 \& W \rightarrow W'$, then $\Gamma \vdash W' : 2$

\[
\frac{
\Gamma, x : 2 \vdash M : 2'}{
\Gamma \vdash \lambda x : 2 (\text{\texttt{w}}) : 2 \rightarrow 2'}
\]

\[
\frac{
\Gamma \vdash M : 2'}{
\Gamma \vdash (\lambda x : 2 (\text{\texttt{w}})) M' : 2}
\]
\[ \Gamma, x : 2 \vdash M : 2 \]
\[ \text{\underline{\Gamma} \vdash \lambda x : 2 (M) : 2 \to 2'} \quad \text{\underline{\Gamma} \vdash M' : 2'} \]
\[ \begin{array}{c}
\text{\underline{\Gamma} \vdash (\lambda x : 2' (M)) M' : 2} \\
\downarrow_{\beta} \\
M[M'/x]
\end{array} \]
\[
\Gamma, x : 2 \vdash M : 2
\]

\[
\Gamma \vdash \lambda x : 2 \to 2 \rightarrow (M) : 2 \to 2
\]

\[
\Gamma \vdash M' : 2'
\]

\[
\Gamma \vdash (\lambda x : 2 \to 2 (M)) : M' : 2
\]

\[
\downarrow_{\beta}
\]

\[
M[M'/x]
\]

To see that this has type 2, we need to prove a Substitution Lemma.
If \( \Gamma \vdash M : \tau \) and \( \Gamma \vdash M' : \tau' \)

then

\( \Gamma \vdash M[M'/x] : \tau \)  

**Substitution Lemma**
(proved by induction on structure of \( M \))
Subject reduction: if \( \Gamma \vdash M : \tau \) \& \( M \rightarrow M' \),
then \( \Gamma \vdash M' : \tau \)

\[
\frac{
\begin{align*}
\Gamma & \vdash M : \tau \\
\alpha & \in \text{fv}(\Gamma)
\end{align*}
}{
\Gamma \vdash \lambda \alpha(M) : \forall \alpha \tau}
\]

\[
\frac{
\Gamma \vdash \lambda \alpha(M) : \forall \alpha \tau
}{
\Gamma \vdash (\forall \alpha(M)) \tau' : \tau}
\]
\[ \Gamma \vdash \text{let } w : 2 \Rightarrow \text{let } \lambda \alpha (m) : \forall \alpha (2) \\vdash \alpha \notin \text{fv}(\Gamma) \]

\[ \Rightarrow \Gamma \vdash (\lambda \alpha (m)) (2) : 2 \Rightarrow \text{let } \lambda \alpha (m) : \forall \alpha (2) \\vdash (2/\alpha)^\prime : 2 \]

\[ \Rightarrow \Gamma \vdash \beta \Rightarrow \text{M}[2/\alpha] \]

To see that this has type \(2 \Rightarrow \text{M}[2/\alpha]\), need to prove a substitution lemma.
If \( \Gamma \vdash M : \tau \) \& \( \alpha \in \text{dom}(\Gamma) \)

then \( \Gamma' \vdash (\alpha/\tau) M : \tau \)

(proved by induction on structure of \( \Gamma \))

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 $M \rightarrow M'$, then $\Gamma \vdash M' : \tau$ is also a provable typing.

**Church Rosser Property.** If $M \rightarrow^* M_1$ and $M \rightarrow^* M_2$, then there is $M'$ with $M_1 \rightarrow^* M'$ and $M_2 \rightarrow^* M'$. 
\[ (\lambda x : \alpha_1 \to \alpha_1 (xy)) \quad (\forall \alpha_2 (\lambda z : \alpha_2 (z))) (\alpha_1 \to \alpha_1) \]

\[ (\lambda x : \alpha_1 \to \alpha_1 (xy)) (\lambda z : \alpha_1 \to \alpha_1 (z)) \]

\[ (\forall \alpha_2 (\lambda z : \alpha_2 (z))) (\alpha_1 \to \alpha_1) \]

\[ \text{y} \]

\[ (\lambda z : \alpha_1 \to \alpha_1 (z)) \text{y} \]
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 $M$. 
Suppose $\Gamma \vdash M : \tau$ is provable in the PLC type system. Then the following properties hold:

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

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

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

$\Omega \triangleq (\lambda x : \alpha . (xx))(\lambda x : \alpha . (xx))$ satisfies $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \ldots$

but it's not typeable (nor is the fixpoint combinator, $Y$)
Theorem 15: [p46]

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 \( M \rightharpoonup N_1 \rightharpoonup N_2 \), ...
Theorem 15. [p. 146]

Church Rosser (CR) + Strong Normalization (SN) 

⇒ Exist unique beta-normal forms for typeable PLC expressions

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

Uniqueness: if \( M \xrightarrow{*} N_1 \xrightarrow{*} M' \) by CR

\[ M \xrightarrow{*} N_2 \xrightarrow{*} M' \]
Theorem 15 [p146]  
Church Rosser (CR) + Strong Normalization (SN)  
⇒ Exist unique beta-normal forms for typeable PLC expressions

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

Uniqueness: if $M \xrightarrow{*} N_1 \xrightarrow{*} M'$, so $N_1 \equiv M'$ α-equiv $N_2 \equiv M'$
PLC beta-conversion, $\equiv_\beta$

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

...
PLC beta-conversion, $=_{\beta}$

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

$$M \rightarrow \cdots \rightarrow M'$$

where each $\rightarrow$ is either $\rightarrow$ or $\leftarrow$, i.e. a beta-reduction in one direction or the other.
PLC beta-conversion, $=_{\beta}$

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

$$M \rightarrow \cdots \rightarrow M'$$

where each $\rightarrow$ 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.)

So $=_{\beta}$ is the smallest equivalence relation containing $\rightarrow$. 
PLC beta-conversion, $=_{\beta}$

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

$$M \rightarrow \cdots \rightarrow \cdots \rightarrow \rightarrow M'$$

where each $\rightarrow$ 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

- 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

\[ \text{bool} \triangleq \forall \alpha (\alpha \to (\alpha \to \alpha)) \]
PLC operator association

\[ M_1 M_2 M_3 \text{ means } (M_1 M_2) M_3 \]

\[ M_1 M_2 \mathcal{L} \text{ means } (M_1 M_2) \mathcal{L} \text{, etc.} \]

\[ \forall \alpha_1, \alpha_2 (\mathcal{L}) \text{ means } \forall \alpha_1 (\forall \alpha_2 (\mathcal{L})) \]

\[ \lambda x_1 : \mathcal{L}_1, x_2 : \mathcal{L}_2 (M) \text{ means } \lambda x_1 : \mathcal{L}_1 (\lambda x_2 : \mathcal{L}_2 (M)) \]

\[ \forall \alpha_1, \alpha_2 (M) \text{ means } \forall \alpha_1 (\forall \alpha_2 (M)) \]
Polymorphic booleans

\[ \text{bool} \triangleq \forall \alpha (\alpha \to (\alpha \to \alpha)) \]

\[ \text{True} \triangleq \Lambda \alpha (\lambda x_1 : \alpha, x_2 : \alpha (x_1)) \]

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

\{ } \vdash \text{True} : \text{bool}

\{ } \vdash \text{False} : \text{bool} \]
Polymorphic booleans

\[ \text{bool} \triangleq \forall \alpha (\alpha \rightarrow (\alpha \rightarrow \alpha)) \]

\[ \text{True} \triangleq \Lambda \alpha (\lambda x_1 : \alpha, x_2 : \alpha (x_1)) \]

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

\[ \text{if} \triangleq \Lambda \alpha (\lambda b : \text{bool}, x_1 : \alpha, x_2 : \alpha (b \alpha x_1 x_2)) \]

\[ \{ \text{if} : \forall \alpha (\text{bool} \rightarrow (\alpha \rightarrow (\alpha \rightarrow \alpha))) \} \]
If \( \{ M_1 \rightarrow^* \text{Tme}, \ M_2 \rightarrow^* \text{N} \} \), then

\[ \mathcal{T} \rightarrow^* \text{Tme} M_2 \ \mathcal{T} \rightarrow^* \text{Tme} M_3 \]
If \( \{ \begin{align*}
M_1 & \rightarrow^* \text{True} \\
M_2 & \rightarrow^* \text{False}
\end{align*} \), then

\[
\text{if } \exists M_1, M_2, M_3 \rightarrow^* \text{ if } \tau \text{ True } M_2 M_3 \parallel \\
\Lambda \alpha(\ldots) \tau \text{ True } M_2 M_3
\]
If \( \{ M_1 \rightarrow^* \text{True}, M_2 \rightarrow^* N \} \), then

\[
\forall \alpha(\ldots) \implies \text{True} \quad M_2 \quad M_3
\]

\[
(\lambda b : b \oplus t, x_1 : t, x_2 : t \rightarrow b \rightarrow x_1 \times x_2) \rightarrow \text{True} \quad M_2 \quad M_3
\]
If \( \{ M_1 \rightarrow^* \text{True}, \ M_2 \rightarrow^* \text{N} \} \), then

\[
\text{if } \exists \ M_1, M_2, M_3 \rightarrow^* \text{ if } \tau \text{ True } M_2, M_3
\]

\[
\prod \alpha(...) \equiv \text{True } M_2, M_3
\]

\[
(\lambda b : \text{bool}, x_1 : \tau, x_2 : \tau (b \Rightarrow x_1, x_2)) \text{ True } M_2, M_3
\]

\[
\downarrow_x
\]

\[
\text{True } \tau \Rightarrow M_2, M_3
\]
If \( \{ M_1 \rightarrow^* \text{Tme} \), then

\[
\text{if } \exists M_1, M_2, M_3 \rightarrow^* \text{ if } \tau \text{Tme } M_2, M_3
\]

\[
\lambda \alpha(...) \tau \text{Tme } M_2, M_3
\]

\[
(\lambda b : \text{bool}, x_1 : \tau, x_2 : \tau (b \Rightarrow x_1, x_2)) \text{Tme } M_2, M_3
\]

\[
\lambda x \text{Tme } \tau \text{ M}_2, M_3
\]

\[
\lambda \alpha (\lambda x_1 : \alpha, x_2 : \alpha(x_1)) \tau \text{ M}_2, M_3
\]
If \( \{ M_i \} \rightarrow M' \rightarrow \top \), then
\[
\left( \begin{array}{c}
M_1 \\
M_2 \\
M_3
\end{array} \right) \Rightarrow
\left( \begin{array}{c}
\alpha(\lambda x_1 \cdot \alpha(\lambda x_2 \cdot \alpha(x_3))) \\
\top \end{array} \right)
\]
FACT: True $\equiv \forall \alpha (\forall x_1, x_2 : \alpha (x_1))$
False $\equiv \forall \alpha (\forall x_1, x_2 : \alpha (x_2))$

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