Monads in ML

A monad in ML is given by type \( \tau(\alpha) \) with a free type variable \( \alpha \) together with expressions

\[
\text{unit} : \alpha \to \tau(\alpha) \\
\text{lift} : (\alpha \to \tau(\beta)) \to \tau(\alpha) \to \tau(\beta)
\]

(writing \( \tau(\beta) \) for the result of replacing \( \alpha \) by \( \beta \) in \( \tau \)) satisfying some equational properties [omitted].
PLC versus the Pure Type System $\lambda 2$

PTS signature:

$$2 \triangleq (S_2, A_2, R_2)$$

where

$$S_2 \triangleq \{*, \square\}$$
$$A_2 \triangleq \{(*, \square)\}$$
$$R_2 \triangleq \{(*, *, *), (\square, *, *)\}$$

Translation of PLC types and terms to $\lambda 2$ pseudo-terms:

$$[\alpha] = \alpha$$
$$[\tau \rightarrow \tau'] = \Pi x : [\tau] ([\tau'])$$
$$[\forall \alpha (\tau)] = \Pi \alpha : * ([\tau'])$$
$$[x] = x$$
$$[\lambda x : \tau (M)] = \lambda x : [\tau] ([M])$$
$$[M M'] = [M] [M']$$
$$[\Lambda \alpha (M)] = \lambda \alpha : * ([M])$$
$$[M \tau] = [M] [\tau]$$
System $\text{F}_\omega$ as a Pure Type System: $\lambda\omega$

PTS specification $\omega = (S_\omega, A_\omega, R_\omega)$:

\[
\begin{align*}
S_\omega & \triangleq \{ *, \Box \} \\
A & \triangleq \{ (*, \Box) \} \\
R & \triangleq \{ (*, *, *), (\Box, *, *), (\Box, \Box, \Box) \}
\end{align*}
\]

"$\text{F}_\omega$ is the work horse of modern compilers"  
(Greg Morrisett)
System \( F_\omega \) as a Pure Type System: \( \lambda \omega \)

PTS specification \( \omega = (S_\omega, A_\omega, R_\omega) \):

\[
S_\omega \triangleq \{*, \Box\} \\
A \triangleq \{(*, \Box)\} \\
R \triangleq \{(*,*,*), (\Box,*,*), (\Box,\Box,\Box)\}
\]

As in \( \lambda 2 \), sort * is a universe of types; but in \( \lambda \omega \), the rule \( \text{prod} \) for \((\Box, \Box, \Box)\) means that \( \Box \vdash t : \Box \) holds for all the ‘simple types’ over the ground type * – the ts generated by the grammar \( t ::= * \mid t \to t \)
System $\text{F}_\omega$ as a Pure Type System: $\lambda\omega$

PTS specification $\omega = (S_\omega, A_\omega, R_\omega)$:

$S_\omega \triangleq \{*, \square\}$

$A \triangleq \{(*, \square)\}$

$R \triangleq \{(\star, \star, \star), (\square, \star, \star), (\square, \square, \square)\}$

As in $\lambda2$, sort $\star$ is a universe of types; but in $\lambda\omega$, the rule (prod) for $(\square, \square, \square)$ means that $\vdash t : \square$ holds for all the ‘simple types’ over the ground type $\star$ – the $t$s generated by the grammar $t ::= \star \mid t \to t$

\[
\frac{\Gamma \vdash A : \square \quad \Gamma ; x : A \vdash B : \square}{\Gamma \vdash \Pi x : A (B) : \square}
\]

where $A \to B \triangleq \Pi x : A (B)$ with $x \notin \text{fv}(B)$.
System $F_\omega$ as a Pure Type System: $\lambda\omega$

PTS specification $\omega = (S_\omega, A_\omega, R_\omega)$:

$$S_\omega \triangleq \{\star, \Box\}$$

$$A \triangleq \{\langle\star, \Box\rangle\}$$

$$R \triangleq \{\langle\star, \star, \star\rangle, \langle\Box, \star, \star\rangle, \langle\Box, \Box, \Box\rangle\}$$

As in $\lambda\Omega$, sort $\star$ is a universe of types; but in $\lambda\omega$, the rule (prod) for $(\Box, \Box, \Box)$ means that $\Diamond \vdash t : \Box$ holds for all the ‘simple types’ over the ground type $\star$ – the $t$s generated by the grammar $t ::= \star | t \to t$

Hence rule (prod) for $(\Box, \star, \star)$ now gives many more legal pseudo-terms of type $\star$ in $\lambda\omega$ compared with $\lambda\Omega$ (PLC), such as

$$\Diamond \vdash (\Pi T : \star \to \star (\Pi \alpha : \star (\alpha \to T \alpha))) : \star$$

$$\Diamond \vdash (\Pi T : \star \to \star (\Pi \alpha, \beta : \star ((\alpha \to T \beta) \to T \alpha \to T \beta))) : \star$$

(\text{types for unit & lift operations, making } T \text{ a monad})
Examples of $\lambda \omega$ type constructions

- Monad transformer for state (using a type $\Diamond \vdash S : \star$ for states):

  \[
  M \triangleq \lambda T : \star \to \star \ (\lambda \alpha : \star (S \to T(\mathcal{P} \alpha S)))
  \]
  \[
  \Diamond \vdash M : (\star \to \star) \to \star \to \star
  \]
Examples of $\lambda\omega$ type constructions

- **Product types (cf. the PLC representation of product types):**
  \[
P \triangleq \lambda \alpha, \beta : \ast (\Pi \gamma : \ast ((\alpha \to \beta \to \gamma) \to \gamma))
  \]
  \[
  \triangledown \vdash P : \ast \to \ast \to \ast
  \]

- **Monad transformer for state (using a type $\triangledown \vdash S : \ast$ for states):**
  \[
  M \triangleq \lambda T : \ast \to \ast (\lambda \alpha : \ast (S \to T(P\alpha S)))
  \]
  \[
  \triangledown \vdash M : (\ast \to \ast) \to \ast \to \ast
  \]
Examples of $\lambda \omega$ type constructions

- Product types (cf. the PLC representation of product types):

  $\mathcal{P} \triangleq \lambda \alpha, \beta : \ast \ ((\Pi \gamma : \ast ((\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \gamma))$

  $\vdash \mathcal{P} : \ast \rightarrow \ast \rightarrow \ast$

- Existential types (cf. the PLC representation of existential types):

  $\exists \mathcal{T} \triangleq \forall \gamma \ ((\mathcal{T} \rightarrow \mathcal{T}' \rightarrow \gamma) \rightarrow \gamma)$

  where $\gamma \notin \text{fv}(\mathcal{T}, \mathcal{T}')$

  (one definition per each choice of types $\mathcal{T}$ & $\mathcal{T}'$)
Examples of $\lambda\omega$ type constructions

- **Product types** (cf. the PLC representation of product types):
  \[
  P \triangleq \lambda \alpha, \beta : \ast \left( \Pi \gamma : \ast \left( (\alpha \to \beta \to \gamma) \to \gamma \right) \right)
  \]
  \[\Diamond \vdash P : \ast \to \ast \to \ast\]

- **Monad transformer for state** (using a type $\uparrow$ for states):
  \[
  M \triangleq \lambda \alpha, \beta : \ast \left( \lambda \gamma : \ast \left( (\alpha \to \beta \to \gamma) \to \gamma \right) \right)
  \]

- **Existential types** (cf. the PLC representation of existential types):
  \[
  \exists \triangleq \lambda T : \ast \to \ast \left( \Pi \beta : \ast \left( (\Pi \alpha : \ast \left( T \alpha \to \beta \right)) \to \beta \right) \right)
  \]
  \[\Diamond \vdash \exists : \left( \ast \to \ast \right) \to \ast\]
Type-checking for $F_\omega$ \((\lambda \omega)\)

**Fact** (Girard): System $F_\omega$ is *strongly normalizing* in the sense that for any legal pseudo-term $t$, there is no infinite chain of beta-reductions $t \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$. 

\((\lambda \omega)\)
Type-checking for $F_\omega$ $(\lambda w)$

\[(\lambda w)\]

**Fact** (Girard): System $F_\omega$ is *strongly normalizing* in the sense that for any [legal] pseudo-term $t$, there is no infinite chain of beta-reductions $t \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$.

As as corollary we have that the type-checking and typeability problems for $F_\omega$ are decidable.
Propositions as Types

(sect. 6 of notes)
Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Logic</th>
<th>$\leftrightarrow$</th>
<th>Type system</th>
</tr>
</thead>
<tbody>
<tr>
<td>propositions $\phi$</td>
<td>$\leftrightarrow$</td>
<td>types $\tau$</td>
</tr>
<tr>
<td>proofs $p$</td>
<td>$\leftrightarrow$</td>
<td>expressions $M$</td>
</tr>
<tr>
<td>‘$p$ is a proof of $\phi$’</td>
<td>$\leftrightarrow$</td>
<td>‘$M$ is an expression of type $\tau$’</td>
</tr>
<tr>
<td>simplification of proofs</td>
<td>$\leftrightarrow$</td>
<td>reduction of expressions</td>
</tr>
</tbody>
</table>

First arose for constructive logics
Constructive interpretation of logic

- **Implication:** a proof of \( \varphi \rightarrow \psi \) is a construction that transforms proofs of \( \varphi \) into proofs of \( \psi \).
- **Negation:** a proof of \( \neg \varphi \) is a construction that from any (hypothetical) proof of \( \varphi \) produces a contradiction (\( = \) proof of falsity \( \bot \))
- **Disjunction:** a proof of \( \varphi \lor \psi \) is an object that manifestly is either a proof of \( \varphi \), or a proof of \( \psi \).
- **For all:** a proof of \( \forall x \ (\varphi(x)) \) is a construction that transforms the objects \( a \) over which \( x \) ranges into proofs of \( \varphi(a) \).
- **There exists:** a proof of \( \exists x \ (\varphi(x)) \) is given by a pair consisting of an object \( a \) and a proof of \( \varphi(a) \).
Constructive interpretation of logic

- **Implication:** a proof of $\varphi \rightarrow \psi$ is a construction that transforms proofs of $\varphi$ into proofs of $\psi$.
- **Negation:** a proof of $\neg \varphi$ is a construction that from any (hypothetical) proof of $\varphi$ produces a contradiction ($= $ proof of falsity $\bot$)
- **Disjunction:** a proof of $\varphi \lor \psi$ is an object that manifestly is either a proof of $\varphi$, or a proof of $\psi$.
- **For all:** a proof of $\forall x (\varphi(x))$ is a construction that transforms the objects $a$ over which $x$ ranges into proofs of $\varphi(a)$.
- **There exists:** a proof of $\exists x (\varphi(x))$ is given by a pair consisting of an object $a$ and a proof of $\varphi(a)$.

The *Law of Excluded Middle* (LEM) $\forall p (p \lor \neg p)$ is a classical tautology (has truth-value *true*), but is rejected by constructivists.
Example of a non-constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.
Example of a non-constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.

**Proof.** Either $\sqrt{2}^{\sqrt{2}}$ is rational, or it is not (LEM!).
Example of a non-constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.

**Proof.** Either $\sqrt{2}^{\sqrt{2}}$ is rational, or it is not (LEM!).

If it is, we can take $a = b = \sqrt{2}$, since $\sqrt{2}$ is irrational by a well-known theorem attributed to Euclid.
Example of a non-constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.

**Proof.** Either $\sqrt{2}^{\sqrt{2}}$ is rational, or it is not (LEM!).

If it is, we can take $a = b = \sqrt{2}$, since $\sqrt{2}$ is irrational by a well-known theorem attributed to Euclid.

If it is not, we can take $a = \sqrt{2}$ and $b = \sqrt{2}^{\sqrt{2}}$, since then $b^a = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \cdot \sqrt{2}} = \sqrt{2}^2 = 2$.

QED
Example of a constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.

**Proof.** $\sqrt{2}$ is irrational by a well-known constructive proof attributed to Euclid.

$2\log_2 3$ is irrational, by an easy constructive proof (exercise).

(If $2\log_2 3 = \frac{m}{n}$, then $3^n = 2^{2n\log_2 3} = 2^m \neq n$)
Example of a constructive proof

**Theorem.** There exist two irrational numbers $a$ and $b$ such that $b^a$ is rational.

**Proof.** $\sqrt{2}$ is irrational by a well-known constructive proof attributed to Euclid.

$2 \log_2 3$ is irrational, by an easy constructive proof (exercise).

So we can take $a = 2 \log_2 3$ and $b = \sqrt{2}$, for which we have that $b^a = (\sqrt{2})^{2 \log_2 3} = (\sqrt{2^2})^{\log_2 3} = 2^{\log_2 3} = 3$ is rational.

QED
Curry-Howard correspondence

Logic $\leftrightarrow$ Type system

propositions $\phi$ $\leftrightarrow$ types $\tau$

proofs $p$ $\leftrightarrow$ expressions $M$

‘$p$ is a proof of $\phi$’ $\leftrightarrow$ ‘$M$ is an expression of type $\tau$’

simplification of proofs $\leftrightarrow$ reduction of expressions

E.g.

2IPC $\leftrightarrow$ PLC
Second-order intuitionistic propositional calculus (2IPC)

2IPC propositions: \( \phi ::= p \mid \phi \rightarrow \phi \mid \forall p \, (\phi) \) where \( p \) ranges over an infinite set of propositional variables.

2IPC sequents: \( \Phi \vdash \phi \) where \( \Phi \) is a finite multiset (= unordered list) of 2IPC propositions and \( \phi \) is a 2IPC proposition.
Second-order intuitionistic propositional calculus (2IPC)

2IPC propositions: \( \phi ::= p \mid \phi \rightarrow \phi \mid \forall p \, (\phi) \) where \( p \) ranges over an infinite set of propositional variables.

2IPC sequents: \( \Phi \vdash \phi \) where \( \Phi \) is a finite multiset (= unordered list) of 2IPC propositions and \( \phi \) is a 2IPC proposition.

\( \Phi \vdash \phi \) is provable if it is in the set of sequents inductively generated by:

\[ \begin{align*}
(\text{Id}) \quad & \Phi \vdash \phi \quad \text{if} \quad \phi \in \Phi \\
(\rightarrow \text{I}) \quad & \frac{\Phi, \phi \vdash \phi'}{\Phi \vdash \phi \rightarrow \phi'} \\
(\rightarrow \text{E}) \quad & \frac{\Phi \vdash \phi \rightarrow \phi'} \quad \frac{\Phi \vdash \phi}{\Gamma \vdash \phi'} \\
(\forall \text{I}) \quad & \frac{\Phi \vdash \phi}{\Phi \vdash \forall p \, (\phi)} \quad \text{if} \quad p \notin \text{fv}(\Phi) \\
(\forall \text{E}) \quad & \frac{\Phi \vdash \forall p \, (\phi)}{\Phi \vdash \phi[f'/p]} 
\end{align*} \]
Logical operations definable in 2IPC

- **Truth** $\top \triangleq \forall p \ (p \to p)$
- **Falsity** $\bot \triangleq \forall p \ (p)$
Logical operations definable in 2IPC

- Truth $\top \triangleq \forall p \ (p \to p)$
- Falsity $\bot \triangleq \forall p \ (p)$
- Conjunction $\phi \land \psi \triangleq \forall p \ (((\phi \to \psi \to p) \to p)$
  (where $p \not\in \text{fv}(\phi, \psi)$)
Logical operations definable in 2IPC

- **Truth** $\top \triangleq \forall p \,(p \rightarrow p)$
- **Falsity** $\bot \triangleq \forall p \,(p)$
- **Conjunction** $\phi \land \psi \triangleq \forall p \,((\phi \rightarrow \psi \rightarrow p) \rightarrow p)$ (where $p \notin \text{fv}(\phi, \psi)$)
- **Disjunction** $\phi \lor \psi \triangleq \forall p \,((\phi \rightarrow p) \rightarrow (\psi \rightarrow p) \rightarrow p)$ (where $p \notin \text{fv}(\phi, \psi)$)

Fact: $\{\}$ $\vdash M: \forall p \,(p \land \neg p)$ is not provable in PLC for any expression $M$. 
Logical operations definable in 2IPC

- **Truth** $\top \triangleq \forall p \, (p \to p)$
- **Falsity** $\bot \triangleq \forall p \, (p)$
- **Conjunction** $\phi \land \psi \triangleq \forall p \, ((\phi \to \psi \to p) \to p)$ (where $p \notin \text{fv}(\phi, \psi)$)
- **Disjunction** $\phi \lor \psi \triangleq \forall p \, ((\phi \to p) \to (\psi \to p) \to p)$ (where $p \notin \text{fv}(\phi, \psi)$)
- **Negation** $\neg \phi \triangleq \phi \to \bot$
- **Bi-implication** $\phi \leftrightarrow \psi \triangleq (\phi \to \psi) \land (\psi \to \phi)$
Logical operations definable in 2IPC

- **Truth** $\top \triangleq \forall p \ (p \rightarrow p)$
- **Falsity** $\bot \triangleq \forall p \ (p)$
- **Conjunction** $\phi \land \psi \triangleq \forall p \ ((\phi \rightarrow \psi \rightarrow p) \rightarrow p)$ (where $p \notin \text{fv}(\phi, \psi)$)
- **Disjunction** $\phi \lor \psi \triangleq \forall p \ ((\phi \rightarrow p) \rightarrow (\psi \rightarrow p) \rightarrow p)$ (where $p \notin \text{fv}(\phi, \psi)$)
- **Negation** $\neg \phi \triangleq \phi \rightarrow \bot$
- **Bi-implication** $\phi \leftrightarrow \psi \triangleq (\phi \rightarrow \psi) \land (\psi \rightarrow \phi)$
- **Existential quantification** $\exists p \ (\phi) \triangleq \forall q \ (\forall p \ (\phi \rightarrow q) \rightarrow q)$ (where $q \notin \text{fv}(\phi, p)$)

Fact: $\{ \}$ $\vdash (\forall p \ (p \rightarrow \neg p))$ is not provable in PLC for any expression $M$. 
A 2IPC proof

Writing $p \land q$ as an abbreviation for $\forall r ((p \to q \to r) \to r)$, the sequent

$$\emptyset \vdash \forall p (\forall q ((p \land q) \to p))$$

is provable in 2IPC:
A 2IPC proof

Writing \( p \land q \) as an abbreviation for \( \forall r \ ((p \to q \to r) \to r) \), the sequent

\[
\{\} \vdash \forall p (\forall q ((p \land q) \to p))
\]

is provable in 2IPC:

\[
\begin{align*}
\text{(Id)} & \quad \{ p \land q, p, q \} \vdash p \\
\text{(¬I)} & \quad \{ p \land q, p \} \vdash q \to p \\
\text{(→I)} & \quad \{ p \land q \} \vdash p \to q \to p \\
\text{(→E)} & \quad \{ p \land q \} \vdash q \to p \\
\text{(∀I)} & \quad \{ \} \vdash \forall q ((p \land q) \to p) \\
\text{(∀I)} & \quad \{ \} \vdash \forall p (\forall q ((p \land q) \to p)) \\
\end{align*}
\]
Curry-Howard correspondence

\[ \text{2IPC} \quad \leftrightarrow \quad \text{PLC} \]

Logic \quad \leftrightarrow \quad \text{Type system}
Curry-Howard correspondence

\[ \text{Logic} \leftrightarrow \text{Type system} \]

propositions \( \phi \) \( \leftrightarrow \) types \( \tau \)
### Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Logic</th>
<th>Type system</th>
</tr>
</thead>
<tbody>
<tr>
<td>propositions $\phi$</td>
<td>types $\tau$</td>
</tr>
<tr>
<td>proofs $p$</td>
<td>expressions $M$</td>
</tr>
</tbody>
</table>

‘$p$ is a proof of $\phi$’ $\leftrightarrow$ ‘$M$ is an expression of type $\tau$’
Mapping 2IPC proofs to PLC expressions

(Id) \( \Phi, \phi \vdash \phi \) \quad \leftrightarrow \quad (id) \( \overline{x} : \Phi, x : \phi \vdash x : \phi \)

(\rightarrow I) \( \frac{\Phi, \phi \vdash \phi'}{\Phi \vdash \phi \rightarrow \phi'} \) \quad \leftrightarrow \quad (\text{fn}) \( \frac{\overline{x} : \Phi \vdash \lambda x : \phi (M) : \phi \rightarrow \phi'}{\overline{x} : \Phi \vdash \lambda x : \phi (M) : \phi \rightarrow \phi'} \)

(\rightarrow E) \( \frac{\Phi \vdash \phi \rightarrow \phi'}{\Phi \vdash \phi} \) \quad \leftrightarrow \quad (\text{app}) \( \frac{\overline{x} : \Phi \vdash M_1 : \phi \rightarrow \phi'}{\overline{x} : \Phi \vdash M_1 \ M_2 : \phi} \)

(\forall I) \( \frac{\Phi \vdash \phi}{\Phi \vdash \forall p (\phi)} \) \quad \leftrightarrow \quad (\text{gen}) \( \frac{\overline{x} : \Phi \vdash M : \phi}{\overline{x} : \Phi \vdash \Lambda p (M) : \forall p (\phi)} \)

(\forall E) \( \frac{\Phi \vdash \forall p (\phi)}{\Phi \vdash \phi[\phi' / p]} \) \quad \leftrightarrow \quad (\text{spec}) \( \frac{\overline{x} : \Phi \vdash M : \forall p (\phi)}{\overline{x} : \Phi \vdash M \phi' : \phi[\phi' / p]} \)
The proof of the 2IPC sequent

\( \{ \} \vdash \forall p (\forall q ((p \land q) \rightarrow p)) \)

given before is transformed by the mapping of 2IPC proofs to PLC expressions to

\( \{ \} \vdash \Lambda p, q (\lambda z : p \land q (z p (\lambda x : p, y : q(x)))) : \forall p (\forall q ((p \land q) \rightarrow p)) \)

with typing derivation:

\[
\begin{align*}
    \text{id} & : \{ z : p \land q, x : p, y : q \} \vdash x : p \\
    \text{fn} & : \{ z : p \land q, x : p \} \vdash \lambda y : q(x) : q \rightarrow p \\
    \text{app} & : \{ z : p \land q \} \vdash \lambda x : p, y : q(x) : p \rightarrow q \rightarrow p \\
    \text{id} & : \{ z : p \land q \} \vdash z p (\lambda x : p, y : q(x)) : p \\
    \text{fn} & : \{ z : p \land q \} \vdash z p (\lambda x : p, y : q(x)) : p \land q (z p (\lambda x : p, y : q(x))) : (p \land q) \rightarrow p \\
    \text{gen} & : \{ \} \vdash \Lambda q (\lambda z : p \land q (z p (\lambda x : p, y : q(x)))) : \forall q ((p \land q) \rightarrow p) \\
    \text{gen} & : \{ \} \vdash \Lambda p, q (\lambda z : p \land q (z p (\lambda x : p, y : q(x)))) : \forall p, q ((p \land q) \rightarrow p) 
\end{align*}
\]
Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Logic</th>
<th>$\iff$</th>
<th>Type system</th>
</tr>
</thead>
<tbody>
<tr>
<td>propositions $\phi$</td>
<td>$\iff$</td>
<td>types $\tau$</td>
</tr>
<tr>
<td>proofs $p$</td>
<td>$\iff$</td>
<td>expressions $M$</td>
</tr>
</tbody>
</table>

- ‘$p$ is a proof of $\phi$’ $\iff$ ‘$M$ is an expression of type $\tau$’

- simplification of proofs $\iff$ reduction of expressions
Proof simplification ↔ Expression reduction

\[
\begin{align*}
\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \rightarrow \psi} & \quad \frac{\Phi \vdash \phi}{\Phi \vdash \psi} \quad \blacktriangleright \quad \frac{\overline{x} : \Phi, x : \phi \vdash M : \psi}{\overline{x} : \Phi \vdash \lambda x : \phi (M) : \phi \rightarrow \psi} \quad \frac{\overline{x} : \Phi \vdash N : \phi}{\overline{x} : \Phi \vdash (\lambda x : \phi (M)) N : \psi}
\end{align*}
\]
Proof simplification $\leftrightarrow$ Expression reduction

The rule $\text{subst}$ for PLC is *admissible*: if its hypotheses are valid PLC typing judgements, then so is its conclusion.
Proof simplification $\leftrightarrow$ Expression reduction

$(\rightarrow I)$
$\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \rightarrow \psi}$
$\frac{\Phi \vdash \phi}{\Phi \vdash \psi}$

$(\rightarrow E)$
$\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \rightarrow \psi}$
$\frac{\Phi \vdash \phi}{\Phi \vdash \psi}$

$(\text{cut})$
$\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \psi}$
$\frac{\Phi \vdash \phi}{\Phi \vdash \psi}$

$(\text{subst})$
$\frac{\Phi, x : \phi \vdash M : \psi}{\Phi \vdash M[N/x] : \psi}$
$\frac{\Phi \vdash N : \phi}{\Phi \vdash \lambda x : \phi (M) : \phi \rightarrow \psi}$

The rule $(\text{subst})$ for PLC is admissible: if its hypotheses are valid PLC typing judgements, then so is its conclusion.
Proof simplification $\leftrightarrow$ Expression reduction

\[
\begin{align*}
(\rightarrow I) & \quad \frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \rightarrow \psi} & \quad \frac{\Phi \vdash \phi}{\Phi \vdash \psi} \quad \Rightarrow \quad \frac{x : \Phi, \lambda x : \phi (M) : \phi \rightarrow \psi}{x : \Phi \vdash \lambda x : \phi (M) \rightarrow \phi} & \quad \frac{x : \Phi \vdash N : \phi}{x : \Phi \vdash (\lambda x : \phi (M)) \rightarrow \phi} \\
(\rightarrow E) & \quad \frac{x : \Phi, x : \phi \vdash M : \psi}{x : \Phi \vdash N : \phi} & \quad \frac{x : \Phi \vdash M[N/x] : \psi}{x : \Phi \vdash N : \phi}
\end{align*}
\]

\text{beta-reduce expression}

The rule (\text{subst}) for PLC is \textit{admissible}: if its hypotheses are valid PLC typing judgements, then so is its conclusion.

Hence, the rule (\text{cut}) is admissible for 2IPC.