System $F_\omega$ as a Pure Type System: $\lambda\omega$

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

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

"$F_\omega$ is the work horse of modern compilers"

(C. 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 $\lambda2$, sort $*$ 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 $*$ – the $t$s generated by the grammar $t ::= * \mid t \to t$

\[
\Gamma \vdash A : \Box \\
\Gamma, \lambda : A \vdash B : \Box
\]

$\frac{}{\Gamma \vdash \Pi \lambda : A (B) : \Box}$

for $(\Box, \Box, \Box)$
System $F_\omega$ as a Pure Type System: $\lambda\omega$

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

$$S_\omega \triangleq \{*, \Box\}$$
$$A_\omega \triangleq \{(*, \Box)\}$$
$$R_\omega \triangleq \{(*, *, *), (\Box, *, *), (\Box, \Box, \Box)\}$$

As in $\lambda 2$, sort $*$ 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 $*$ – the $t$s generated by the grammar $t ::= * \mid t \to t$

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

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

$$
S_\omega \triangleq \{\ast, \Box\} \\
A_\omega \triangleq \{(*, \Box)\} \\
R_\omega \triangleq \{(\ast, \ast, \ast), (\Box, \ast, \ast), (\Box, \Box, \Box)\}
$$

As in $\lambda2$, sort $\ast$ 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 $\ast$ – the $t$s generated by the grammar $t ::= \ast \mid t \to t$

Hence rule (prod) for $(\Box, \ast, \ast)$ now gives many more legal pseudo-terms of type $\ast$ in $\lambda\omega$ compared with $\lambda2$ (PLC), such as

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

types for unit & lift operations, making $T$ a monad
Examples of $\lambda\omega$ type constructions

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

\[ P \triangleq \lambda \alpha, \beta : * (\Pi \gamma : * ((\alpha \to (\beta \to \gamma)) \to \gamma)) \]
\[ \Diamond \vdash P : * \to * \to * \]

Product $2 \times 2$:
\[ 2 \times 2 \triangleq A \forall x (2 \to 2 \to 2) \to x \to x \to x \to x \]

where $x \in \text{fin}(1, 2)$.
Examples of $\lambda\omega$ type constructions

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

- Existential types (cf. the PLC representation of existential types):
  \[
  \exists \alpha \triangleq \forall \beta \left( (\forall \alpha \, (T \alpha \to \beta)) \to \beta \right)
  \]
  \[
  \text{where } \beta \in \text{Fin}(\tau) \& \beta \neq \alpha
  \]
  \[
  \Diamond \vdash \exists : (\ast \to \ast) \to \ast
  \]
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 $t \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$. 
Type-checking for $F_\omega$

**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 $t \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$.

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

$$(\lambda w)$$
Propositions as Types
(sect. 6 of notes)
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>
<tr>
<td>‘$p$ is a proof of $\phi$’</td>
<td>$\iff$</td>
<td>‘$M$ is an expression of type $\tau$’</td>
</tr>
<tr>
<td>simplification of proofs</td>
<td>$\iff$</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 ($\equiv$ 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) $\varphi \lor \neg \varphi$ is a classical tautology (has truth-value true), but is rejected by constructivists.
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 ($\equiv$ 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).
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

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

E.g.

| 2IPC | ↔ | 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: $\phi ::= p \mid \phi \to \phi \mid \forall p \, (\phi)$ where $p$ ranges over an infinite set of propositional variables.

2IPC sequents: $\Phi \vdash \phi$ where $\Phi$ is a finite multiset ($= \text{unordered list}$) 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:

(Id) $\Phi \vdash \phi$ if $\phi \in \Phi$

(→I) $\frac{\Phi, \phi \vdash \phi'}{\Phi \vdash \phi \to \phi'}$

(→E) $\frac{\Phi \vdash \phi \to \phi' \quad \Phi \vdash \phi}{\Gamma \vdash \phi'}$

(∀I) $\frac{\Phi \vdash \phi}{\Phi \vdash \forall p \, (\phi)}$ if $p \notin \text{fv}(\Phi)$

(∀E) $\frac{\Phi \vdash \forall p \, (\phi)}{\Phi \vdash \phi[\phi'/p]}$
Logical operations definable in 2IPC

- **Truth** $\top \triangleq \forall p \ (p \rightarrow p)$
- **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 $p \notin \text{fv}(\phi, \psi)$)
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:** $M : \forall p \ (p \_ \neg p)$ is not provable in PLC for any expression $M$. 
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 \( p \not\in \text{fv}(\phi, \psi) \))
- **Disjunction** \( \phi \lor \psi \triangleq \forall p \ ((\phi \rightarrow p) \rightarrow (\psi \rightarrow p) \rightarrow p) \) (where \( p \not\in \text{fv}(\phi, \psi) \))
- **Negation** \( \neg \phi \triangleq \phi \rightarrow \bot \)
- **Bi-implication** \( \phi \leftrightarrow \psi \triangleq (\phi \rightarrow \psi) \land (\psi \rightarrow \phi) \)
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** $\phi \iff \psi \triangleq (\phi \to \psi) \land (\psi \to \phi)$
- **Existential quantification** $\exists p \ (\phi) \triangleq \forall q \ (\forall p \ (\phi \to q) \to q)$ (where $q \notin \text{fv}(\phi, p)$)
Writing $p \land q$ as an abbreviation for $\forall r ((p \rightarrow q \rightarrow r) \rightarrow r)$, the sequent

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

is provable in 2IPC:
A 2IPC proof

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

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

is provable in 2IPC:

1. (Id)
   $\{p \land q, p, q\} \vdash p$
2. (→I)
   $\{p \land q, p\} \vdash q \rightarrow p$
3. (Id)
   $\{p \land q\} \vdash \forall r ((p \rightarrow q \rightarrow r) \rightarrow r)$
4. (→I)
   $\{p \land q\} \vdash p \rightarrow q \rightarrow p$
5. (∨E)
   $\{p \land q\} \vdash (p \rightarrow q \rightarrow q) \rightarrow q$
6. (→I)
   $\{p \land q\} \vdash p$
7. (∀I)
   $\{\} \vdash (p \land q) \rightarrow p$
8. (∀I)
   $\{\} \vdash \forall q ((p \land q) \rightarrow p)$
9. (∀I)
   $\{\} \vdash \forall p (\forall q ((p \land q) \rightarrow p))$
Curry-Howard correspondence

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

\[ \begin{array}{ccc}
\text{Logic} & \leftrightarrow & \text{Type system} \\
\text{propositions } \phi & \leftrightarrow & \text{types } \tau
\end{array} \]
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>
<tr>
<td>‘$p$ is a proof of $\phi$’</td>
<td>‘$M$ is an expression of type $\tau$’</td>
</tr>
</tbody>
</table>
Mapping 2IPC proofs to PLC expressions

(Id) \( \Phi, \phi \vdash \phi \)    \( \rightarrow (id) \bar{x} : \Phi, x : \phi \vdash x : \phi \)

(\(\rightarrow\)I) \( \Phi, \phi \vdash \phi' \)    \( \Phi \vdash \phi \rightarrow \phi' \)    \( \rightarrow (fn) \bar{x} : \Phi, x : \phi \vdash \lambda x : \phi (M) : \phi \rightarrow \phi' \)

(\(\rightarrow\)E) \( \Phi \vdash \phi \rightarrow \phi' \)    \( \Phi \vdash \phi \)    \( \rightarrow (app) \bar{x} : \Phi \vdash M_1 : \phi \rightarrow \phi' \)

(\(\forall\)I) \( \Phi \vdash \phi \)    \( \Phi \vdash \forall p (\phi) \)    \( \rightarrow (gen) \bar{x} : \Phi \vdash M : \phi \)

(\(\forall\)E) \( \Phi \vdash \forall p (\phi) \)    \( \Phi \vdash \phi[\phi'/p] \)    \( \rightarrow (spec) \bar{x} : \Phi \vdash M : \forall p (\phi) \)

(\(\forall\)E) \( \Phi \vdash \forall p (\phi) \)    \( \Phi \vdash \phi[\phi'/p] \)    \( \rightarrow (spec) \bar{x} : \Phi \vdash M : \forall p (\phi) \)
The proof of the 2IPC sequent

\[
\{\} \vdash \forall p \left( \forall q \left( (p \land q) \rightarrow p \right) \right)
\]

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

\[
\{\} \vdash \forall p, q \left( \left( \forall z : p \land q \left( z p \left( \lambda x : p, y : q \left( x \right) \right) \right) \right) \right)
\]

with typing derivation:

\[
\begin{array}{c}
\{z : p \land q, x : p, y : q\} \vdash x : p \\
\{z : p \land q, x : p\} \vdash \lambda y : q \left( x \right) : q \rightarrow p \\
\{z : p \land q\} \vdash \lambda x : p, y : q \left( x \right) : p \rightarrow q \rightarrow p \\
\{z : p \land q\} \vdash z : \forall r \left( (p \rightarrow q \rightarrow r) \rightarrow r \right) \\
\{z : p \land q\} \vdash z : p \left( (p \rightarrow q \rightarrow p) \rightarrow p \right) \\
\{z : p \land q\} \vdash z p \left( \lambda x : p, y : q \left( x \right) \right) : (p \land q) \rightarrow p \\
\{\} \vdash \lambda z : p \land q \left( z p \left( \lambda x : p, y : q \left( x \right) \right) \right) : (p \land q) \rightarrow p \\
\{\} \vdash \forall p, q \left( \lambda z : p \land q \left( z p \left( \lambda x : p, y : q \left( x \right) \right) \right) \right) : \forall p, q \left( (p \land q) \rightarrow p \right)
\end{array}
\]
Curry-Howard correspondence

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

$\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \psi}$ \quad \frac{\Phi \vdash \phi}{\Phi \vdash \psi}

$\frac{\Phi \vdash \phi \rightarrow \psi}{\Phi \vdash \psi}$

$\Phi \vdash \psi$

$\frac{\bar{x} : \Phi, x : \phi \vdash M : \psi}{\bar{x} : \Phi \vdash \lambda x : \phi (M) : \phi \rightarrow \psi}$

$\frac{\bar{x} : \Phi \vdash N : \phi}{\bar{x} : \Phi \vdash (\lambda x : \phi (M)) N : \psi}$

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

Hence, the rule (cut) is admissible for 2IPC.
Proof simplification $\leftrightarrow$ Expression reduction

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

\[
\frac{\Phi, \phi \vdash \psi}{\Phi, \phi \vdash \phi \rightarrow \psi} \quad \frac{\Phi \vdash \phi}{\Phi \vdash \psi} \quad \iff \quad \frac{\bar{x} : \Phi, x : \phi \vdash M : \psi}{\bar{x} : \Phi, \lambda x : \phi (M) : \phi \rightarrow \psi} \quad \frac{\bar{x} : \Phi \vdash N : \phi}{\bar{x} : \Phi \vdash (\lambda x : \phi (M)) N : \psi}
\]

\[
\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \psi} \quad \iff \quad \frac{\bar{x} : \Phi, x : \phi \vdash M : \psi}{\bar{x} : \Phi \vdash M[N/x] : \psi}
\]

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

\[ \begin{align*}
\text{(→I)} & \quad \frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \to \psi} & \frac{\phi \vdash \psi}{\Phi \vdash \phi} \quad \iff \quad \frac{\bar{x} : \Phi, x : \phi \vdash M : \psi}{\bar{x} : \Phi \vdash \lambda x : \phi (M) : \phi \to \psi} & \frac{\bar{x} : \Phi \vdash N : \phi}{\bar{x} : \Phi \vdash (\lambda x : \phi (M)) N : \psi} \\
\text{(→E)} & \quad \frac{\Phi, \phi \vdash \psi}{\Phi \vdash \psi} \quad \iff \quad \frac{\bar{x} : \Phi, x : \phi \vdash M : \psi}{\bar{x} : \Phi \vdash N : \phi} & \frac{\bar{x} : \Phi \vdash N : \phi}{\bar{x} : \Phi \vdash M[N/x] : \psi} \quad \text{(subst)}
\end{align*} \]

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.
Type-inference versus proof search

*Type-inference*: given $\Gamma$ and $M$, is there a type $\tau$ such that $\Gamma \vdash M : \tau$?

(For PLC/2IPC this is decidable.)
Type-inference versus proof search

_Type-inference:_ given $\Gamma$ and $M$, is there a type $\tau$ such that $\Gamma \vdash M : \tau$?

(For PLC/2IPC this is decidable.)

_Proof-search:_ given $\Gamma$ and $\phi$, is there a proof term $M$ such that $\Gamma \vdash M : \phi$?

(For PLC/2IPC this is undecidable.)