# 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 |

# Curry-Howard correspondence

Applications

| 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 |

Girard's Linear logic $\rightsquigarrow$ usage analysis in compilers

## Linear implication $\multimap$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \multimap \psi} \qquad \frac{\Gamma \vdash \varphi \multimap \psi \quad \Delta \vdash \varphi}{\Gamma, \Delta \vdash \psi} \ (\Gamma \cap \Delta = \emptyset)$$

## Linear conjunction (tensor)

$$\frac{\Gamma \vdash \varphi \quad \Delta \vdash \psi}{\Gamma, \Delta \vdash \varphi \otimes \psi} \ (\Gamma \cap \Delta = \emptyset)$$

$$\frac{\Gamma \vdash \varphi \otimes \psi \quad \Delta, \varphi, \psi \vdash \theta}{\Gamma, \Delta \vdash \theta} \ (\Gamma \cap \Delta = \emptyset)$$

# Curry-Howard correspondence

Applications

| 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 |

Linear Temporal Logic $\rightsquigarrow$ functional reactive programming

Modal logics $\rightsquigarrow$ partial evaluation & run-time code generation

# Type-inference versus proof search

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

(For PLC/2IPC this is decidable.)

*Proof-search*: given $\boldsymbol{\Gamma}$ and $\boldsymbol{\phi}$, is there a proof term $\boldsymbol{M}$ such that $\boldsymbol{\Gamma \vdash M : \phi}$?

(For PLC/2IPC this is undecidable.)

# Curry-Howard correspondence

Applications

| Logic | | 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 |

proof assistants
( Agda,
Coq
⋮ )

dependently typed
λ-calculus
(eg. Calculus of Constructions)

# 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 |

a logic of propositions

E.g.

| | | |
|---|---|---|
| 2IPC | $\leftrightarrow$ | PLC |

C-H also applied to predicate logic

# Curry-Howard correspondence

higher-order intuitionistic predicate logic $\longleftrightarrow$ Calculus of Constructions

# Pure Type Systems – typing rules

$$(\textbf{axiom})\ \frac{}{\diamond \vdash s_1 : s_2}\ \text{if } (s_1, s_2) \in \mathcal{A}$$

$$(\textbf{start})\ \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ \text{if } x \notin dom(\Gamma)$$

$$(\textbf{weaken})\ \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma, x : B \vdash M : A}\ \text{if } x \notin dom(\Gamma)$$

$$(\textbf{conv})\ \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}\ \text{if } A =_\beta B$$

$$(\textbf{prod})\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A\,(B) : s_3}\ \text{if } (s_1, s_2, s_3) \in \mathcal{R}$$

$$(\textbf{abs})\ \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A\,(B) : s}{\Gamma \vdash \lambda x : A\,(M) : \Pi x : A\,(B)}$$

$$(\textbf{app})\ \frac{\Gamma \vdash M : \Pi x : A\,(B) \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B[N/x]}$$

($A, B, M, N$ range over pseudoterms, $s, s_1, s_2, s_3$ over sort symbols)

# Calculus of Constructions

is the Pure Type System $\lambda\mathbf{C}$, where $\mathbf{C} = (\mathcal{S}_{\mathrm{C}}, \mathcal{A}_{\mathrm{C}}, \mathcal{R}_{\mathrm{C}})$ is the PTS specification with

$$\mathcal{S}_{\mathrm{C}} \triangleq \{\mathtt{Prop}, \mathtt{Set}\} \quad (\mathtt{Prop} = \text{a sort of propositions}, \mathtt{Set} = \text{a sort of types})$$

$$\mathcal{A}_{\mathrm{C}} \triangleq \{(\mathtt{Prop}, \mathtt{Set})\} \quad (\mathtt{Prop} \text{ is one of the types})$$

$$\begin{aligned}\mathcal{R}_{\mathrm{C}} \triangleq \{&(\mathtt{Prop}, \mathtt{Prop}, \mathtt{Prop}), (\mathtt{Set}, \mathtt{Prop}, \mathtt{Prop}),\\ &(\mathtt{Prop}, \mathtt{Set}, \mathtt{Set}), (\mathtt{Set}, \mathtt{Set}, \mathtt{Set})\}\end{aligned}$$

# Calculus of Constructions

is the Pure Type System $\lambda\mathbf{C}$, where $\mathbf{C} = (\mathcal{S}_C, \mathcal{A}_C, \mathcal{R}_C)$ is the PTS specification with

$$\begin{aligned}
\mathcal{S}_C &\triangleq \{\texttt{Prop}, \texttt{Set}\} \\
\mathcal{A}_C &\triangleq \{(\texttt{Prop}, \texttt{Set})\} \\
\mathcal{R}_C &\triangleq \{(\texttt{Prop}, \texttt{Prop}, \texttt{Prop})^{1}, (\texttt{Set}, \texttt{Prop}, \texttt{Prop}), \\
&\qquad (\texttt{Prop}, \texttt{Set}, \texttt{Set}), (\texttt{Set}, \texttt{Set}, \texttt{Set})\}
\end{aligned}$$

1. $\texttt{Prop}$ has implications, $\phi \to \psi = \Pi x : \phi\ (\psi)$ (where $\phi, \psi : \texttt{Prop}$ and $x \notin fv(q)$).

# Calculus of Constructions

is the Pure Type System $\lambda\mathbf{C}$, where $\mathbf{C} = (\mathcal{S}_{\mathrm{C}}, \mathcal{A}_{\mathrm{C}}, \mathcal{R}_{\mathrm{C}})$ is the PTS specification with

$$
\begin{aligned}
\mathcal{S}_{\mathrm{C}} &\triangleq \{\mathtt{Prop}, \mathtt{Set}\} \\
\mathcal{A}_{\mathrm{C}} &\triangleq \{(\mathtt{Prop}, \mathtt{Set})\} \\
\mathcal{R}_{\mathrm{C}} &\triangleq \{(\mathtt{Prop}, \mathtt{Prop}, \mathtt{Prop})^{\mathbf{1}}, (\mathtt{Set}, \mathtt{Prop}, \mathtt{Prop})^{\mathbf{2}}, \\
&\qquad (\mathtt{Prop}, \mathtt{Set}, \mathtt{Set}), (\mathtt{Set}, \mathtt{Set}, \mathtt{Set})\}
\end{aligned}
$$

1. $\mathtt{Prop}$ has implications, $\phi \to \psi = \Pi x : \phi\ (\psi)$ (where $\phi, \psi : \mathtt{Prop}$ and $x \notin fv(q)$).

2. $\mathtt{Prop}$ has universal quantifications over elements of a type, $\Pi x : A\ (\phi(x))$ (where $A : \mathtt{Set}$ and $x : A \vdash \phi(x) : \mathtt{Prop}$).
N.B. $A$ might be $\mathtt{Prop}$ ($\lambda 2 \subseteq \lambda\mathbf{C}$).

# Calculus of Constructions

is the Pure Type System $\lambda\mathbf{C}$, where $\mathbf{C} = (\mathcal{S}_\mathrm{C}, \mathcal{A}_\mathrm{C}, \mathcal{R}_\mathrm{C})$ is the PTS specification with

$$\mathcal{S}_\mathrm{C} \triangleq \{\mathtt{Prop}, \mathtt{Set}\}$$
$$\mathcal{A}_\mathrm{C} \triangleq \{(\mathtt{Prop}, \mathtt{Set})\}$$
$$\mathcal{R}_\mathrm{C} \triangleq \{(\mathtt{Prop}, \mathtt{Prop}, \mathtt{Prop})^{1}, (\mathtt{Set}, \mathtt{Prop}, \mathtt{Prop})^{2}, (\mathtt{Prop}, \mathtt{Set}, \mathtt{Set}), (\mathtt{Set}, \mathtt{Set}, \mathtt{Set})^{4}\}$$

1. $\mathtt{Prop}$ has implications, $\phi \to \psi = \Pi x : \phi\ (\psi)$ (where $\phi, \psi : \mathtt{Prop}$ and $x \notin fv(q)$).

2. $\mathtt{Prop}$ has universal quantifications over elements of a type, $\Pi x : A\ (\phi(x))$ (where $A : \mathtt{Set}$ and $x : A \vdash \phi(x) : \mathtt{Prop}$).
N.B. $A$ might be $\mathtt{Prop}$ ($\lambda 2 \subseteq \lambda\mathbf{C}$).

4. $\mathtt{Set}$ has dependent function types, $\Pi x : A\ (B(x))$ (where $A : \mathtt{Set}$ and $x : A \vdash B(x) : \mathtt{Set}$).

# Calculus of Constructions

is the Pure Type System $\lambda\mathbf{C}$, where $\mathbf{C} = (\mathcal{S}_{\mathrm{C}}, \mathcal{A}_{\mathrm{C}}, \mathcal{R}_{\mathrm{C}})$ is the PTS specification with

$$
\begin{aligned}
\mathcal{S}_{\mathrm{C}} &\triangleq \{\mathtt{Prop}, \mathtt{Set}\} \\
\mathcal{A}_{\mathrm{C}} &\triangleq \{(\mathtt{Prop}, \mathtt{Set})\} \\
\mathcal{R}_{\mathrm{C}} &\triangleq \{(\mathtt{Prop}, \mathtt{Prop}, \mathtt{Prop})^{\mathbf{1}}, (\mathtt{Set}, \mathtt{Prop}, \mathtt{Prop})^{\mathbf{2}}, \\
&\qquad (\mathtt{Prop}, \mathtt{Set}, \mathtt{Set})^{\mathbf{3}}, (\mathtt{Set}, \mathtt{Set}, \mathtt{Set})^{\mathbf{4}}\}
\end{aligned}
$$

1. $\mathtt{Prop}$ has implications, $\phi \to \psi = \Pi x : \phi\ (\psi)$ (where $\phi, \psi : \mathtt{Prop}$ and $x \notin fv(q)$).

2. $\mathtt{Prop}$ has universal quantifications over elements of a type, $\Pi x : A\ (\phi(x))$ (where $A : \mathtt{Set}$ and $x : A \vdash \phi(x) : \mathtt{Prop}$).
N.B. $A$ might be $\mathtt{Prop}$ ($\lambda 2 \subseteq \lambda\mathbf{C}$).

3. $\mathtt{Set}$ has types of function dependent on proofs of a proposition, $\Pi x : p\ (A(x))$ (where $p : \mathtt{Prop}$ and $x : p \vdash A(x) : \mathtt{Set}$).

4. $\mathtt{Set}$ has dependent function types, $\Pi x : A\ (B(x))$ (where $A : \mathtt{Set}$ and $x : A \vdash B(x) : \mathtt{Set}$).

# Some general properties of $\boldsymbol{\lambda}\mathbf{C}$

- It extends both $\boldsymbol{\lambda}\mathbf{2}$ (PLC) and $\boldsymbol{\lambda\omega}$ ($\mathbf{F}_\omega$).

# Some general properties of $\boldsymbol{\lambda}\mathbf{C}$

- It extends both $\boldsymbol{\lambda}\mathbf{2}$ (PLC) and $\boldsymbol{\lambda\omega}$ ($\mathbf{F}_\omega$).
- $\boldsymbol{\lambda}\mathbf{C}$ is strongly normalizing.
- Type-checking and typeability are decidable.

# Logical operations definable in ~~2IPC~~ $\lambda C$

- *Truth* $\top \triangleq \forall p\,(p \to p)$
- *Falsity* $\bot \triangleq \forall p\,(p)$
- *Conjunction* $\phi \wedge \psi \triangleq \forall p\,((\phi \to \psi \to p) \to p)$ (where $p \notin fv(\phi, \psi)$)
- *Disjunction* $\phi \vee \psi \triangleq \forall p\,((\phi \to p) \to (\psi \to p) \to p)$ (where $p \notin fv(\phi, \psi)$)
- *Negation* $\neg\phi \triangleq \phi \to \bot$
- *Bi-implication* $\phi \leftrightarrow \psi \triangleq (\phi \to \psi) \wedge (\psi \to \phi)$

$p \to q \triangleq \Pi x : p\,(q)$ $\quad x \notin fv(p)$

$\forall p(\varphi) \triangleq \Pi p : Prop\,(\varphi)$

# Some general properties of $\boldsymbol{\lambda}\mathbf{C}$

- It extends both $\boldsymbol{\lambda}\mathbf{2}$ (PLC) and $\boldsymbol{\lambda\omega}$ ($\mathbf{F}_\omega$).

- $\boldsymbol{\lambda}\mathbf{C}$ is strongly normalizing.
- Type-checking and typeability are decidable.

- $\boldsymbol{\lambda}\mathbf{C}$ is logically consistent (relative to the usual foundations of classical mathematics), that is, there is no pseudo-term $\boldsymbol{t}$ satisfying $\diamond \vdash \boldsymbol{t} : \boldsymbol{\Pi p} : \texttt{Prop}\,(\boldsymbol{p})$.

  Indeed there is no proof of LEM ($\boldsymbol{\Pi p} : \texttt{Prop}\,(\neg \boldsymbol{p} \vee \boldsymbol{p})$).

# Leibniz equality in $\boldsymbol{\lambda}\mathbf{C}$

Gottfried Wilhelm Leibniz (1646–1716),
**identity of indiscernibles**:
*duo quaedam communes proprietates eorum nequaquam possit*
(two distinct things cannot have all their properties in common).

# Leibniz equality in $\lambda\mathbf{C}$

Gottfried Wilhelm Leibniz (1646–1716),
**identity of indiscernibles**:
*duo quaedam communes proprietates eorum nequaquam possit*
(two distinct things cannot have all their properties in common).

Given $\Gamma \vdash A : \mathtt{Set}$ in $\lambda\mathbf{C}$, we can define

$$\mathtt{Eq}_A \triangleq \lambda x, y : A\,(\Pi P : A \to \mathtt{Prop}\,(P\,x \leftrightarrow P\,y))$$

satisfying $\Gamma \vdash \mathtt{Eq}_A : A \to A \to \mathtt{Prop}$ and giving a well-behaved (but not extensional) equality predicate for elements of type $A$.

# Leibniz equality in $\lambda\mathbf{C}$

Gottfried Wilhelm Leibniz (1646–1716),
**identity of indiscernibles**:
*duo quaedam communes proprietates eorum nequaquam possit*
(two distinct things cannot have all their properties in common).

Given $\Gamma \vdash A : \mathtt{Set}$ in $\lambda\mathbf{C}$, we can define

$$\mathtt{Eq}_A \triangleq \lambda x, y : A\,(\Pi P : A \to \mathtt{Prop}\,\underline{(P\,x \leftrightarrow P\,y)})$$

satisfying $\Gamma \vdash \mathtt{Eq}_A : A \to A \to \mathtt{Prop}$ and giving a well-behaved (but not extensional) equality predicate for elements of type $A$.

$$p \leftrightarrow q \triangleq (p \to q) \wedge (q \to p)$$

# Extensionality

**Functional extensionality:**

$$\mathtt{FunExt}_{A,B} \triangleq \Pi f, g : A \to B\ ( \\ (\Pi x : A\ (\mathtt{Eq}_B\ (f\ x)\ (g\ x))) \to \mathtt{Eq}_{A \to B}\ f\ g)$$

# Extensionality

**Functional extensionality:**

$$\texttt{FunExt}_{A,B} \triangleq \Pi f,g : A \to B\ (\ (\Pi x : A\ (\texttt{Eq}_B\ (f\ x)\ (g\ x))) \to \texttt{Eq}_{A\to B}\ f\ g)$$

If $\Gamma \vdash A, B : \texttt{Set}$ in $\lambda C$, then $\Gamma \vdash \texttt{Ext}_{A,B} : \texttt{Prop}$ is derivable, but for some $A$,$B$ there does not exist a pseudo-term $t$ for which $\Gamma \vdash t : \texttt{Ext}_{A,B}$ is derivable.

FunExt

Typo!

# Extensionality

**Functional extensionality:**

$$\texttt{FunExt}_{A,B} \triangleq \Pi f, g : A \to B \; ( \\ (\Pi x : A \; (\texttt{Eq}_B \; (f\, x) \; (g\, x))) \to \texttt{Eq}_{A \to B} \; f \; g)$$

If $\Gamma \vdash A, B : \texttt{Set}$ in $\lambda\mathbf{C}$, then $\Gamma \vdash$ Fun$\texttt{Ext}_{A,B} : \texttt{Prop}$ is derivable, but for some $A$,$B$ there does not exist a pseudo-term $t$ for which $\Gamma \vdash t :$ Fun$\texttt{Ext}_{A,B}$ is derivable.

**Propositional extensionality:**

$$\texttt{PropExt} \triangleq \Pi p, q : \texttt{Prop} \; ((p \leftrightarrow q) \to \texttt{Eq}_{\texttt{Prop}} \; p \; q)$$

$\diamond \vdash \texttt{PropExt} : \texttt{Prop}$ is derivable in $\lambda\mathbf{C}$, but there does not exist a pseudo-term $t$ for which $\diamond \vdash t : \texttt{PropExt}$ is derivable.

# Extensionality

This is a weak form of Voevodsky's Univalence Axiom – currently a HOT topic in type theory research (Homotopy Type Theory)

**Propositional extensionality:**

$$\mathtt{PropExt} \triangleq \Pi p, q : \mathtt{Prop}\,((p \leftrightarrow q) \to \mathtt{Eq}_{\mathtt{Prop}}\, p\, q)$$

$\diamond \vdash \mathtt{PropExt} : \mathtt{Prop}$ is derivable in $\lambda C$, but there does not exist a pseudo-term $t$ for which $\diamond \vdash t : \mathtt{PropExt}$ is derivable.