Curry-Howard correspondence

Logic	$\leftrightarrow$	Type system
propositions ϕ	$\leftrightarrow$	types τ
proofs p	$\leftrightarrow$	expressions M
'p is a proof of ϕ'	$\leftrightarrow$	'M is an expression of type τ'
simplification of proofs	$\leftrightarrow$	reduction of expressions

Curry-Howard correspondence

Applications

Logic	$\leftrightarrow$	Type system
propositions ϕ	$\leftrightarrow$	types τ
proofs p	$\leftrightarrow$	expressions M
'p is a proof of ϕ'	$\leftrightarrow$	'M is an expression of type τ'
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 ϕ	$\leftrightarrow$	types τ
proofs p	$\leftrightarrow$	expressions M
'p is a proof of ϕ'	$\leftrightarrow$	'M is an expression of type τ'
simplification of proofs	$\leftrightarrow$	reduction of expressions

Linear Temporal logic $\leadsto$ functional reactive programming

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

Type-inference versus proof search

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

(For PLC/2IPC this is decidable.)

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

(For PLC/2IPC this is undecidable.)

Curry-Howard correspondence

Applications

Logic	$\leftrightarrow$	Type system
propositions ϕ	$\leftrightarrow$	types τ
proofs p	$\leftrightarrow$	expressions M
'p is a proof of ϕ'	$\leftrightarrow$	'M is an expression of type τ'
simplification of proofs	$\leftrightarrow$	reduction of expressions

proof assistants
(Agda,
Coq
⋮)

dependently typed
λ-calculus

(eg. Calculus of Constructions)

Curry-Howard correspondence

<u>Logic</u>	$\leftrightarrow$	<u>Type system</u>
propositions ϕ	$\leftrightarrow$	types τ
proofs p	$\leftrightarrow$	expressions M
'p is a proof of ϕ'	$\leftrightarrow$	'M is an expression of type τ'
simplification of proofs	$\leftrightarrow$	reduction of expressions

a logic of propositions

E.g.

2IPC	$\leftrightarrow$	PLC

C-H also applied to <u>predicate</u> logic

Curry-Howard correspondence

higher-order intuitionistic <u>predicate</u> 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

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

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

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

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

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

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

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

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

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

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

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 $\mathtt{Prop}$ ($\lambda\mathbf{2} \subseteq \lambda\mathbf{C}$).

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

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

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

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 $\mathtt{Prop}$ ($\lambda\mathbf{2} \subseteq \lambda\mathbf{C}$).

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.

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 $\diamond \vdash \boldsymbol{t} : \boldsymbol{\Pi p} : \mathtt{Prop}\,(\boldsymbol{p})$.

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

Logical operations definable in ~~2IPC~~ λC

- *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* $\phi \leftrightarrow \psi \triangleq (\phi \to \psi) \wedge (\psi \to \phi)$

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

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

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 λC

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 $\Gamma \vdash A : \texttt{Set}$ in λC, we can define

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

satisfying $\Gamma \vdash \texttt{Eq}_A : A \to A \to \texttt{Prop}$ 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:

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

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

Extensionality

Functional extensionality:

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

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

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