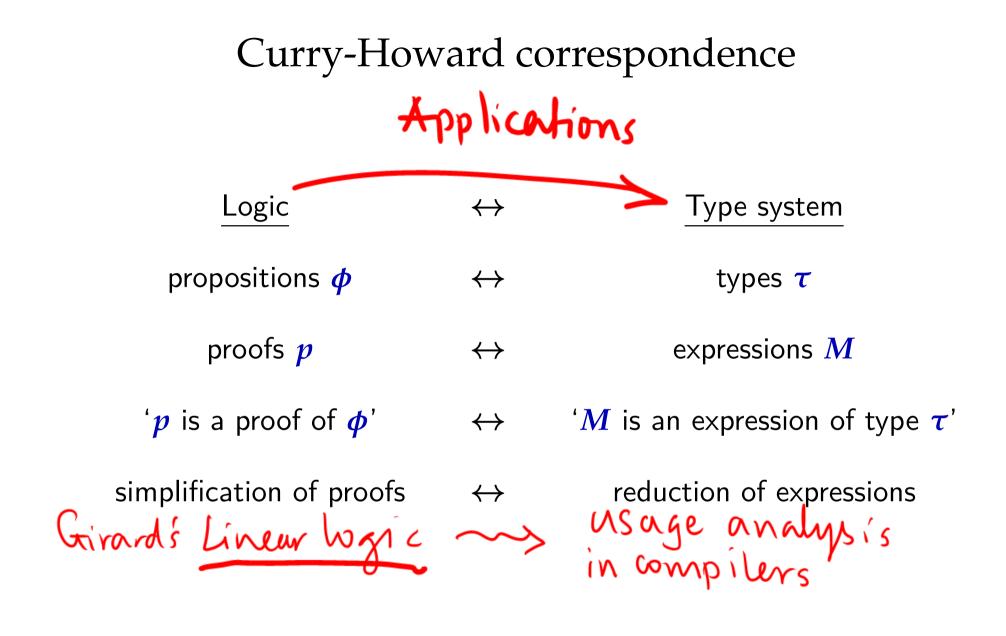
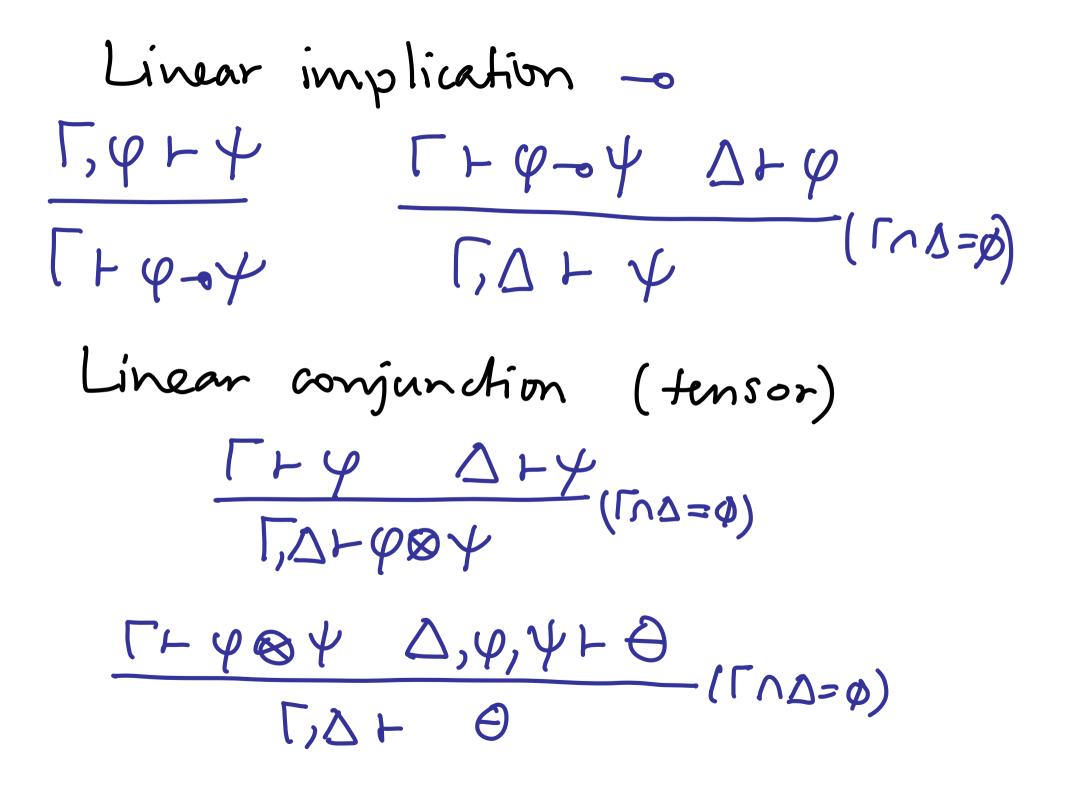
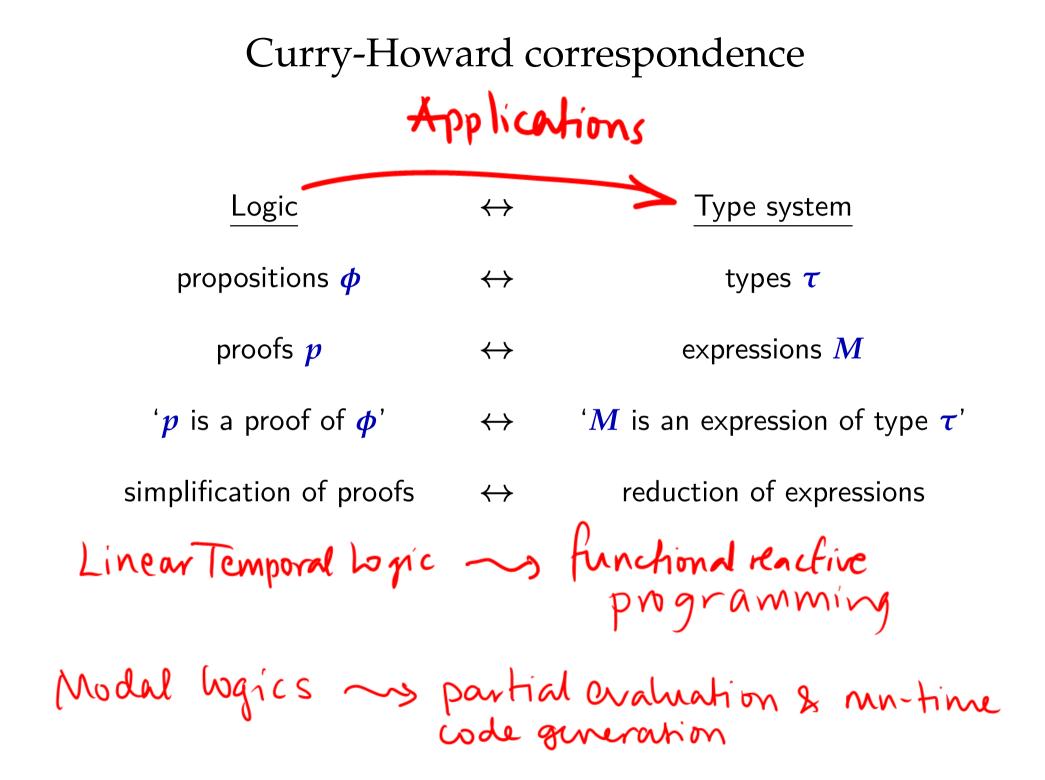
Curry-Howard correspondence

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





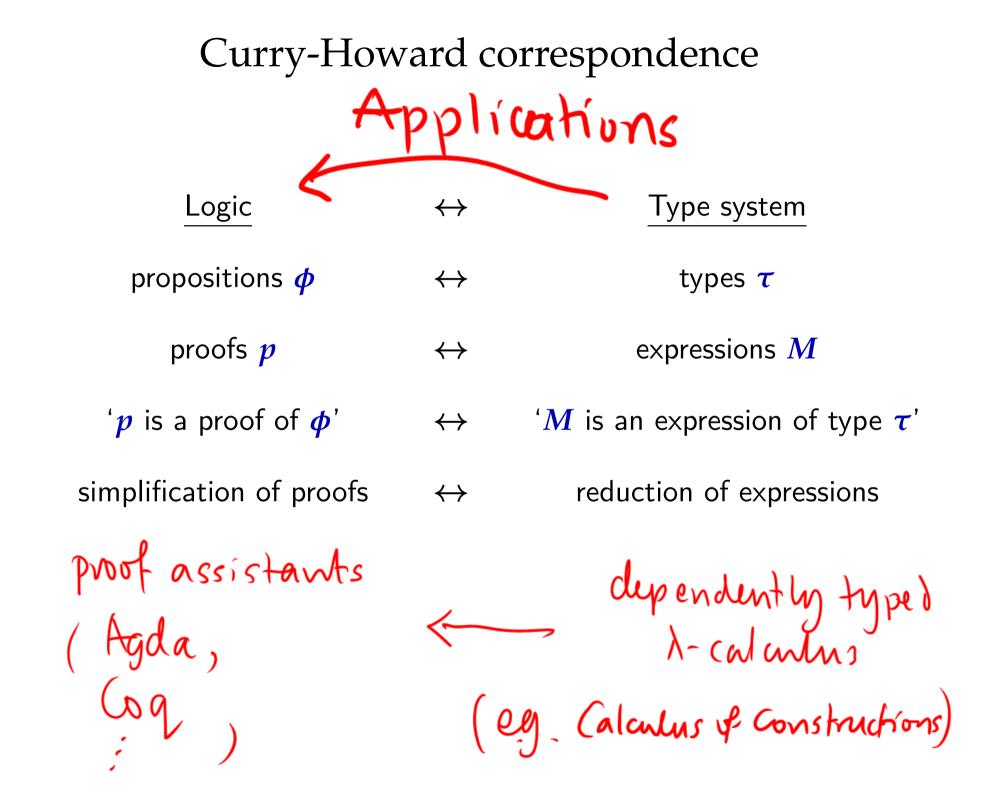


Type-inference versus proof search

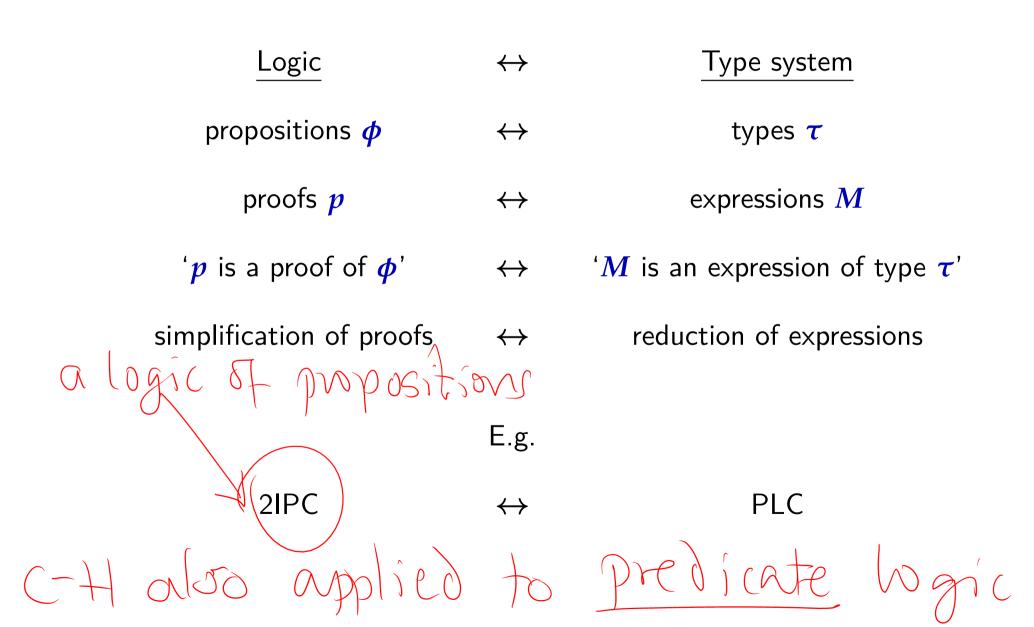
Type-inference: given Γ and M, is there a type τ such that $\Gamma \vdash M : \tau$?

(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



Curry-Howard correspondence

higher-order intuitionistic predicate OMC

Ca) culus Anstructions

Pure Type Systems – typing rules

$$(axiom) \xrightarrow{\diamond \vdash s_{1} : s_{2}} \text{ if } (s_{1}, s_{2}) \in \mathcal{A}$$

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

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

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

$$(prod) \xrightarrow{\Gamma \vdash A : s_{1}} \\ \Gamma \vdash \Pi x : A (B) : s_{3} \text{ if } (s_{1}, s_{2}, s_{3}) \in \mathcal{R}$$

$$(abs) \xrightarrow{\Gamma, x : A \vdash M : B} \\ \Gamma \vdash M : B \\ \Gamma \vdash N : A (B) : s_{3} \text{ if } (s_{1}, s_{2}, s_{3}) \in \mathcal{R}$$

$$(app) \xrightarrow{\Gamma \vdash M : \Pi x : A (B)} \\ \Gamma \vdash M : B[N/x] \\ (A, B, M, N \text{ range over pseudoterms, } s, s_{1}, s_{2}, s_{3} \text{ over sort symbols})$$

is the Pure Type System λC , where $C = (S_C, A_C, \mathcal{R}_C)$ is the PTS specification with

$$\begin{split} \boldsymbol{\mathcal{S}_C} &\triangleq \{\texttt{Prop, Set}\} \quad (\texttt{Prop} = \texttt{a sort of propositions, Set} = \texttt{a sort of types}) \\ \boldsymbol{\mathcal{A}_C} &\triangleq \{(\texttt{Prop, Set})\} \quad (\texttt{Prop is one of the types}) \\ \boldsymbol{\mathcal{R}_C} &\triangleq \{(\texttt{Prop, Prop, Prop}), (\texttt{Set}, \texttt{Prop, Prop}), \\ & (\texttt{Prop, Set}, \texttt{Set}), (\texttt{Set}, \texttt{Set})\} \end{split}$$

is the Pure Type System λC , where $C = (S_C, A_C, \mathcal{R}_C)$ is the PTS specification with

$$\begin{split} \mathcal{S}_{C} &\triangleq \{\texttt{Prop, Set}\} \\ \mathcal{A}_{C} &\triangleq \{(\texttt{Prop, Set})\} \\ \mathcal{R}_{C} &\triangleq \{(\texttt{Prop, Prop, Prop})^{1}, (\texttt{Set, Prop, Prop}), \\ & (\texttt{Prop, Set, Set}), (\texttt{Set, Set}, \texttt{Set})\} \end{split}$$

1. Prop has implications, $\phi \rightarrow \psi = \Pi x : \phi(\psi)$ (where ϕ, ψ : Prop and $x \notin fv(q)$).

is the Pure Type System λC , where $C = (S_C, A_C, \mathcal{R}_C)$ is the PTS specification with

$$\begin{split} \mathcal{S}_{C} &\triangleq \{\texttt{Prop, Set}\} \\ \mathcal{A}_{C} &\triangleq \{(\texttt{Prop, Set})\} \\ \mathcal{R}_{C} &\triangleq \{(\texttt{Prop, Prop, Prop})^{1}, (\texttt{Set, Prop, Prop})^{2}, \\ & (\texttt{Prop, Set, Set}), (\texttt{Set, Set}, \texttt{Set})\} \end{split}$$

1. Prop has implications, $\phi \rightarrow \psi = \Pi x : \phi(\psi)$ (where ϕ, ψ : Prop and $x \notin fv(q)$).

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

is the Pure Type System λC , where $C = (S_C, A_C, \mathcal{R}_C)$ is the PTS specification with

$$\begin{split} \mathcal{S}_{C} &\triangleq \{\texttt{Prop, Set}\} \\ \mathcal{A}_{C} &\triangleq \{(\texttt{Prop, Set})\} \\ \mathcal{R}_{C} &\triangleq \{(\texttt{Prop, Prop, Prop})^{1}, (\texttt{Set, Prop, Prop})^{2}, \\ & (\texttt{Prop, Set, Set}), (\texttt{Set, Set}, \texttt{Set})^{4}\} \end{split}$$

1. Prop has implications, $\phi \rightarrow \psi = \Pi x : \phi(\psi)$ (where ϕ, ψ : Prop and $x \notin fv(q)$).

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

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

is the Pure Type System λC , where $C = (S_C, A_C, \mathcal{R}_C)$ is the PTS specification with

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

1. Prop has implications, $\phi \rightarrow \psi = \Pi x : \phi(\psi)$ (where ϕ, ψ : Prop and $x \notin fv(q)$).

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

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

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

Some general properties of λC

• It extends both $\lambda 2$ (PLC) and $\lambda \omega$ (F_{ω}).

Some general properties of λC

- It extends both $\lambda 2$ (PLC) and $\lambda \omega$ (F_{ω}).
- λC is strongly normalizing.
- Type-checking and typeability are decidable.

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 $p \notin fv(\phi, \psi)$)
- *Disjunction* $\phi \lor \psi \triangleq \forall p ((\phi \rightarrow p) \rightarrow (\psi \rightarrow p) \rightarrow p)$ (where $p \notin fv(\phi, \psi)$)
- Negation $\neg \phi \triangleq \phi \rightarrow \bot$
- Bi-implication $\phi \leftrightarrow \psi \triangleq (\phi \rightarrow \psi) \land (\psi \rightarrow \phi)$

$$P \rightarrow q \stackrel{\Delta}{=} \Pi x: p(q) \quad x \notin fr(p)$$

 $\forall p(q) \stackrel{\Delta}{=} \Pi p: Prop(q)$

Some general properties of λC

- It extends both $\lambda 2$ (PLC) and $\lambda \omega$ (F_{ω}).
- λC is strongly normalizing.
- Type-checking and typeability are decidable.
- λC is logically consistent (relative to the usual foundations of classical mathematics), that is, there is no pseudo-term t satisfying ◇ ⊢ t : Πp : Prop (p).

Indeed there is no proof of LEM $(\Pi p : \operatorname{Prop}(\neg p \lor p))$.

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).

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

 $\operatorname{Eq}_A \triangleq \lambda x, y : A\left(\Pi P : A
ightarrow \operatorname{Prop}\left(P \, x \leftrightarrow P \, y\right)
ight)$

satisfying $\Gamma \vdash Eq_A : A \rightarrow A \rightarrow Prop$ 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 $\Gamma \vdash A$: Set in λC , we can define

 $\operatorname{Eq}_A riangleq \lambda x, y : A\left(\Pi P : A o \operatorname{Prop}\left(P \, x \leftrightarrow P \, y\right)
ight)$

satisfying $\Gamma \vdash Eq_A : A \rightarrow A \rightarrow Prop$ and giving a well-behaved (but not extensional) equality predicate for elements of type A.

$$p \leftrightarrow q \stackrel{\Delta}{=} (p \rightarrow q) \land (q \rightarrow p)$$

Functional extensionality:

 $ext{FunExt}_{A,B} \triangleq \Pi f, g : A o B ((\Pi x : A (\operatorname{Eq}_B (f x) (g x))) o \operatorname{Eq}_{A o B} f g)$

Functional extensionality:

 $\begin{aligned} & \operatorname{FunExt}_{A,B} \triangleq \Pi f, g : A \to B \left(\\ & \left(\Pi x : A \left(\operatorname{Eq}_B \left(f \, x \right) \left(g \, x \right) \right) \right) \to \operatorname{Eq}_{A \to B} f \, g \right) \end{aligned}$ $& \operatorname{If} \Gamma \vdash A, B : \operatorname{Set} \text{ in } \lambda C, \text{ then } \Gamma \vdash \operatorname{Ext}_{A,B} : \operatorname{Prop} \text{ is derivable, but} \\ & \operatorname{for some} A, B \text{ there does not exist a pseudo-term } t \text{ for which} \\ & \Gamma \vdash t : \operatorname{Ext}_{A,B} \text{ is derivable.} \end{aligned}$

Functional extensionality:

$$\begin{split} \operatorname{FunExt}_{A,B} &\triangleq \Pi f,g: A \to B \left(\\ \left(\Pi x: A \left(\operatorname{Eq}_B \left(f \, x \right) \left(g \, x \right) \right) \right) \to \operatorname{Eq}_{A \to B} f \, g \right) \\ \\ & \mathsf{If} \ \Gamma \vdash A,B: \mathsf{Set} \ \mathsf{in} \ \lambda \mathsf{C}, \ \mathsf{then} \ \Gamma \vdash \mathsf{Ext}_{A,B}: \mathsf{Prop} \ \mathsf{is} \ \mathsf{derivable}, \ \mathsf{but} \\ \\ & \mathsf{for} \ \mathsf{some} \ A,B \ \mathsf{there} \ \mathsf{does} \ \mathsf{not} \ \mathsf{exist} \ \mathsf{a} \ \mathsf{pseudo-term} \ t \ \mathsf{for} \ \mathsf{which} \\ \\ & \Gamma \vdash t: \mathsf{Ext}_{A,B} \ \mathsf{is} \ \mathsf{derivable}. \end{split}$$

Propositional extensionality:

 $\texttt{PropExt} \triangleq \Pi p, q : \texttt{Prop}\left((p \leftrightarrow q) \rightarrow \texttt{Eq}_{\texttt{Prop}} \, p \, q\right)$

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

This is a meak form of Voevodsky's Univalence Axiom - Unrenthy a Hot topic in type theory research Propositional extensionality: (Homotopy Type Theory) $\blacktriangleright \texttt{PropExt} \triangleq \Pi p, q : \texttt{Prop}\left((p \leftrightarrow q) \rightarrow \texttt{Eq}_{\texttt{Prop}} \, p \, q\right)$

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