The Curry-Howard correspondence journey

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA, Serbia

Lecture 2

Proof Society International School
Birmingham, September 2024

Journey

LOGIC	COMPUTATION
intuitionistic	λ-calculus combinatory logic
second-order	polymorphism
predicate	λ cube
classical	$\lambda\mu$-calculi
???	intersection types
communication	process calculi
??? federated (machine) learning	???

Roadmap Lecture 2

Connectives and constructors

Second-order logic and polymorphism

Higher-order logic, predicate logic and the λ cube

Classical logic and control operators

Roadmap

Connectives and constructors

Second-order logic and polymorphism

Higher-order logic, predicate logic and the λ cube

Classical logic and control operators

Full simply typed λ-calculus

Terms $$M, N, L ::= x \mid \lambda x.M \mid MN \mid$$
$$\mid \langle M, N \rangle \mid \pi_1(M) \mid \pi_2(M) \mid$$
$$\mid \mathsf{in}_1(M) \mid \mathsf{in}_2(M) \mid$$
$$\mid \mathsf{case}\ M\ \mathsf{of}\ \mathsf{in}_1(x) \Rightarrow N \mid\ \mathsf{in}_2(y) \Rightarrow L \mid$$
$$\mid\ \langle\rangle\ \mid\ \mathsf{abort}(M)$$

Types $$\sigma, \tau ::= \alpha \mid \sigma \to \tau \mid$$
$$\mid \sigma \wedge \tau \mid$$
$$\mid \sigma \vee \tau \mid$$
$$\mid 1 \mid 0$$

$\lambda \to$ extended

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \wedge \tau} \qquad \frac{\Gamma \vdash M : \sigma \wedge \tau}{\Gamma \vdash \pi_1(M) : \sigma} \quad \frac{\Gamma \vdash M : \sigma \wedge \tau}{\Gamma \vdash \pi_2(M) : \tau}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \ \mathsf{in}_1(M) : \sigma \vee \tau} \quad \frac{\Gamma \vdash M : \tau}{\Gamma \vdash \ \mathsf{in}_2(M) : \sigma \vee \tau}$$

$$\frac{\Gamma \vdash M : \sigma \vee \tau \quad \Gamma, x : \sigma \vdash N : \rho \quad \Gamma, y : \tau \vdash L : \rho}{\Gamma \vdash \mathsf{case}\ M\ \mathsf{of}\ \mathsf{in}_1(x) \Rightarrow N \mid \ \mathsf{in}_2(y) \Rightarrow L : \rho}$$

$$\frac{\Gamma \vdash M : 0}{\Gamma \vdash \mathsf{abort}(M) : \sigma}$$

Curry-Howard correspondence

Intuitionistic logic (all connectives) vs full $\lambda \to$

$$\vdash \sigma \quad \Leftrightarrow \quad \vdash M : \sigma$$

Theorem

A formula is provable in intuitionistic logic
if and only if it is inhabited in full $\lambda \to$

Semantics: λ-calculus vs logics

inverse limit models

$\hookrightarrow$ Scott models, D_∞ 1970s

term models

graph models

filter models

... other models

Kripke models

S. Kašterović, SG
Kripke semantics and completeness for full simply typed lambda calculus
Journal of Logic and Computation 30 (8), pp 1567-1608 (2020)

Kripke semantics:

- modal logics, late 1950s, 1960s, Kripke
- intuitionistic logic
- non-classical logics
- probability logics
- ...
- λ-calculus, 1990s, Mitchell, Moggi

Roadmap

Connectives and constructors

Second-order logic and polymorphism

Higher-order logic, predicate logic and the λ cube

Classical logic and control operators

Second-order propositional logic, PROP2

PROP2 = PROP (minimal IL) extended with

$$\frac{\Gamma \vdash \sigma}{\Gamma \vdash \forall\alpha.\sigma}\ \alpha \notin \mathsf{FV}(\Gamma) \qquad \frac{\Gamma \vdash \forall\alpha.\sigma}{\Gamma \vdash \sigma[\tau/\alpha]}$$

Constructive second order proposition logic

$\not\forall\ M : \forall\alpha.\forall\beta.((\alpha \to \beta) \to \alpha) \to \alpha$ Peirce's law

NB Peirce's law is not inhabited.

PROP2

Definability of the other connectives in PROP2:

$$\begin{aligned}
\bot &:= \forall\alpha.\alpha \\
\sigma\wedge\tau &:= \forall\alpha.(\sigma \to \tau \to \alpha) \to \alpha \\
\sigma\vee\tau &:= \forall\alpha.(\sigma \to \alpha) \to (\tau \to \alpha) \to \alpha \\
\exists\alpha.\sigma &:= \forall\beta.(\forall\alpha.\sigma \to \beta) \to \beta
\end{aligned}$$

NB

- In intuitionistic propositional logic connectives are independent

$\lambda 2\ (\mathcal{F})$ Polymorphic types - Why?

Problem:

- expressiveness of $\lambda \rightarrow$ is very limited
- in $\lambda \rightarrow$ a function is bound to its type and it cannot be reused
- $\lambda x : \alpha . x : \alpha \rightarrow \alpha$ and $\lambda x : \beta . x : \beta \rightarrow \beta$

Solution:

- define functions that treat types polymorphically
- add types $\forall \alpha . \sigma$
- $M : \forall \alpha . (\alpha \rightarrow \alpha)$, then M can map any type to itself.

$\lambda 2$ ($\mathcal{F}$) language

Terms and types

1970s

- Girard system $\mathcal{F}$
- Reynolds language FORSYTHE

Language

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid (\mathcal{T}\mathcal{T}) \mid \lambda x : \text{Type}.\mathcal{T} \mid \lambda\alpha.\mathcal{T} \mid (\mathcal{T}\,\text{Type})$$

$$\text{Type} := \text{TVar} \mid (\text{Type} \to \text{Type}) \mid \forall\alpha.\text{Type}$$

Reductions

There are two kinds of β-reductions

- $(\lambda x : \sigma.M)P \longrightarrow_\beta M[P/x]$
- $(\lambda\alpha.M)\tau \longrightarrow_\beta M[\tau/\alpha]$

$\lambda 2$ type system

$\lambda 2$ type system = $\lambda \rightarrow$ extended with

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \lambda\alpha.M : \forall\alpha.\sigma}\ \alpha \notin \mathsf{FV}(\Gamma) \qquad \frac{\Gamma \vdash M : \forall\alpha.\sigma}{\Gamma \vdash M\tau : \sigma[\tau/\alpha]}$$

Example

- $\mathsf{K} = \lambda x.\lambda y.x$
  $\lambda\alpha.\lambda\beta.\lambda x:\alpha.\lambda y:\beta.x : \forall\alpha.\forall\beta.\alpha \rightarrow \beta \rightarrow \alpha.$
- $\mathsf{K} = \lambda x.\lambda y.x$
  $\lambda x:(\forall\alpha.\alpha).\lambda y:\sigma.x\tau : (\forall\alpha.\alpha) \rightarrow \sigma \rightarrow \tau.$
- $\Delta = \lambda x.xx$
  $\lambda x:(\forall\alpha.\alpha).x(\sigma \rightarrow \tau)(x\,\sigma) : (\forall\alpha.\alpha) \rightarrow \tau$

I. $\lambda 2$ fundamental properties

- Uniqueness of types
  If $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$, then $\sigma = \tau$
- Church-Rosser property holds in $\lambda 2$
- Subject Reduction
  If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_{\beta\eta} N$, then $\Gamma \vdash N : \sigma$
- Strong Normalisation
  If $\Gamma \vdash M : \sigma$, then $M \in$ SN i.e. all $\beta\eta$-reductions from M terminate
- Expressiveness: all primitive recursive functions

Curry-Howard: $\lambda 2$ and PROP2

Theorem (Girard, Reynolds)

$$\vdash \sigma\ (\mathit{PROP2}) \quad \Leftrightarrow \quad \vdash M : \sigma\ (\lambda 2)$$

$\lambda 2$: 3 Type?

Theorem

In $\lambda 2$

- *Type checking ($(M : \sigma)$?) is undecidable*
- *Type inference (M :?) is undecidable*
- *Type inhabitation (? : σ) is undecidable*

Proof.

- Type inhabitation is equivalent to provability in *PROP*2, by the Curry-Howard correspondence
- Type checking and types inference were open questions till 1990, Wells

□

A restriction of $\lambda 2$ has an easy type inference algorithm (Hindley-Milner) and is used for many statically typed functional programming languages such as Haskell and the ML family

Roadmap

Connectives and constructors

Second-order logic and polymorphism

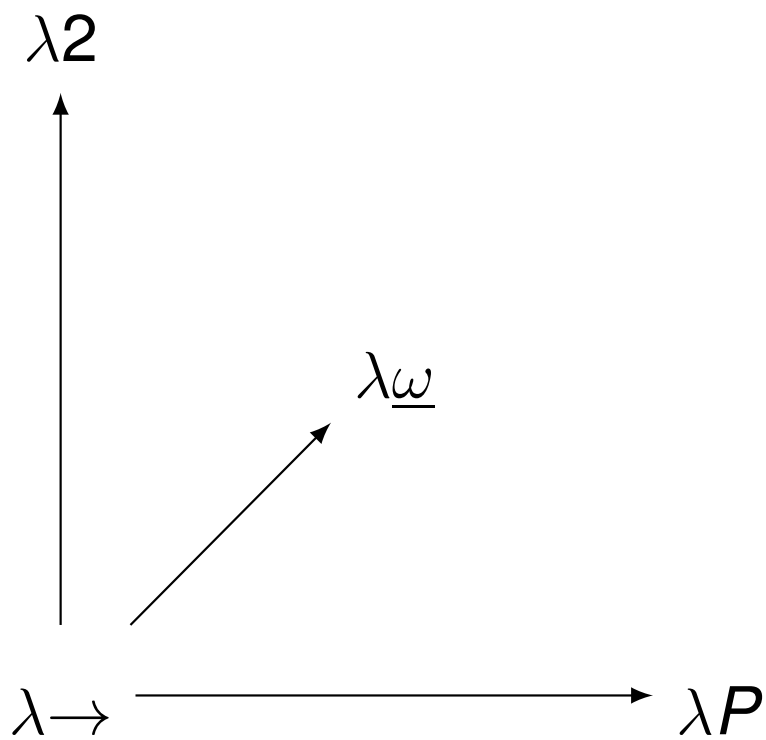
Higher-order logic, predicate logic and the λ cube

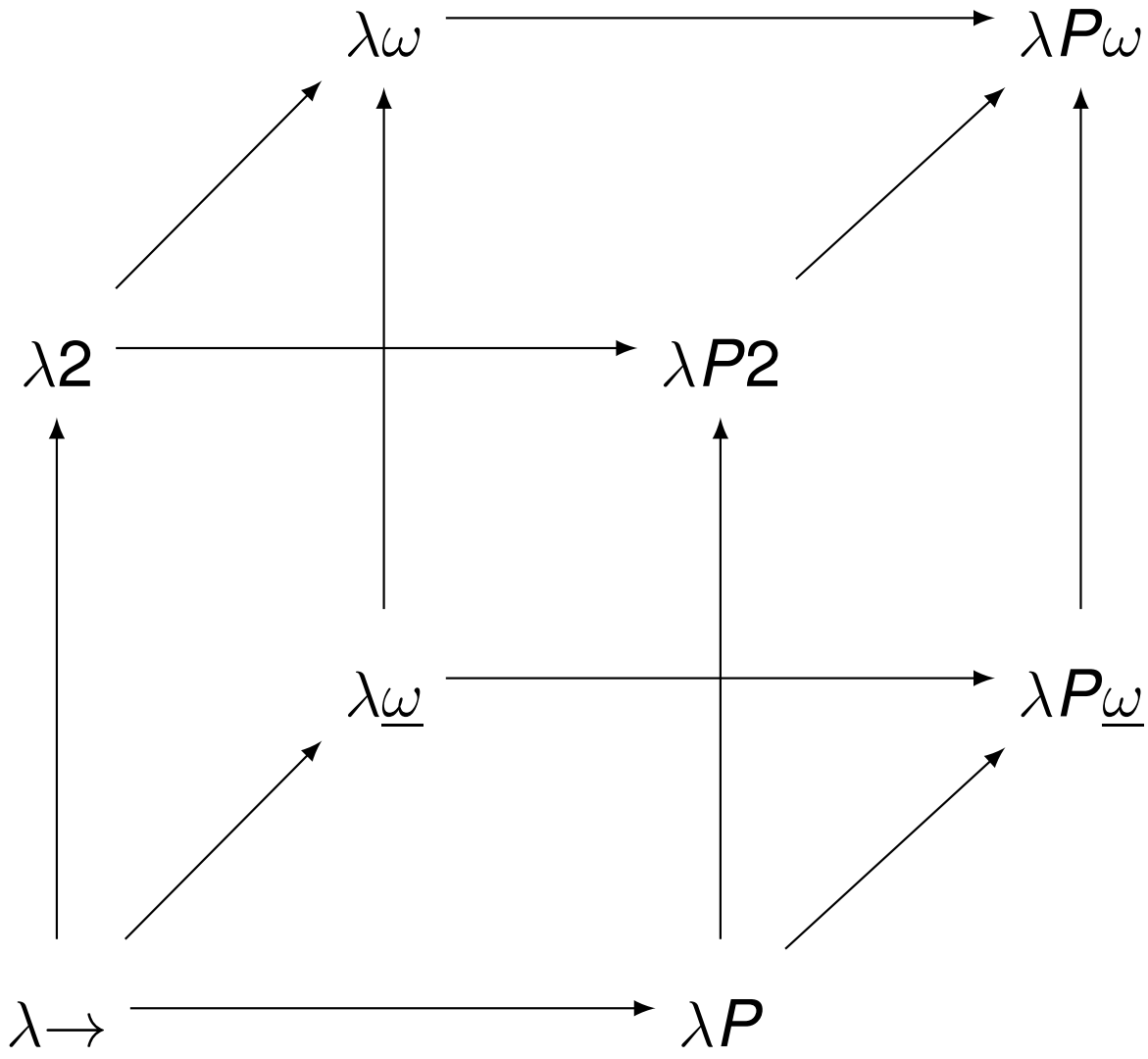
Classical logic and control operators

Dependency

Terms and types are mutually dependent

terms depending on terms	$\lambda \rightarrow$
terms depending on types	$\lambda 2$
types depending on types	$\lambda \underline{\omega}$
types depending on terms	λP


$\lambda 2$
$\lambda\underline{\omega}$
$\lambda\!\rightarrow$
λP


$\lambda\omega$
$\lambda P\omega$
$\lambda 2$
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda\!\rightarrow$
λP

λ cube

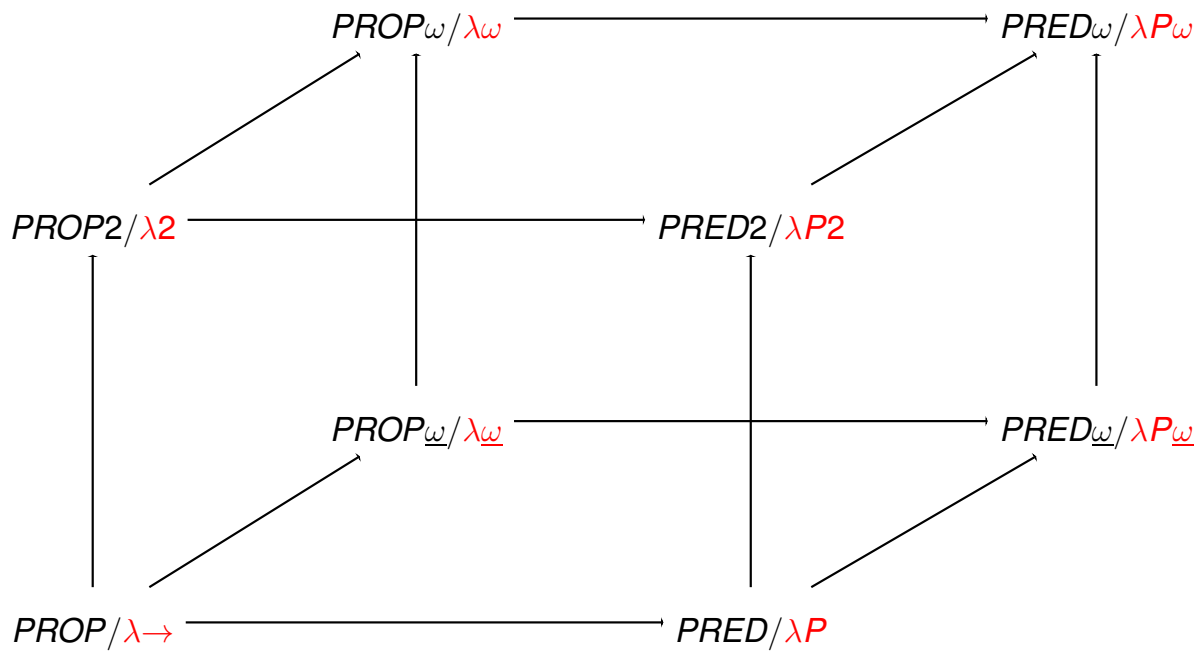
System		
$\lambda \rightarrow$	simple theory of types	Church (1940)
$\lambda 2$	system $\mathcal{F}$	Girard (1972) Reynolds (1974)
λP	AUT-QE (AUTOMAT) logical frameworks (LF)	de Bruijn (1970) Harper *et al.* (1987)
$\lambda \underline{\omega}$	POLYREC	Renardel de Lavalette (19
$\lambda P2$		Longo and Moggi (1988)
$\lambda \omega$	$\mathcal{F}\omega$	Girard (1972)
$\lambda P \omega$	calculus of constructions	Coquand and Huet (1988
$\lambda P \underline{\omega}$		

Good properties of the λ cube

- Uniqueness of types ✓
- Confluence (Church-Rosser property) ✓
- Type preservation under reduction (Subject Reduction) ✓
- Termination (Strong Normalisation) ✓
- Expressiveness

Cube: Logic vs Computation

PROP	proposition logic	$\lambda\!\rightarrow$
PROP2	second-order proposition logic	$\lambda 2$ $(\mathcal{F})$
PROP$\underline{\omega}$	weakly higher-order proposition logic	$\lambda\underline{\omega}$
PROPω	higher-order proposition logic	$\lambda\omega$ $(\mathcal{F}\omega)$
PRED	predicate logic	λP
PRED2	second-order predicate logic	$\lambda P2$
PRED$\underline{\omega}$	weakly higher-order predicate logic	$\lambda P\underline{\omega}$
PREDω	higher-order predicate logic	$\lambda P\omega$ (CC)



(a) Logic and Lambda cube

$\vdash A$ (LOGIC) if and only if $\vdash M : A$ (λLOGIC)

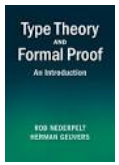
Proofs-as-programs extended

Proof assistants:

COQ, Agda, Lean, and many more

- theorem proving (e.g. fundamental theorem of algebra)
- proof checking (e.g. four colour theorem)
- verification (e.g. CompCert - formal verification of realistic compilers usable for critical embedded software)

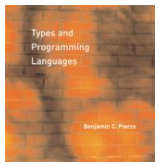
References



R. Nederpelt, H. Geuvers

Type Theory and Formal Proof

Cambridge University Press 2014



B. C. Pierce

Types and programming languages

MIT Press 2002

Software Foundations

`https://softwarefoundations.cis.upenn.edu`

The Coq proof assistant

`https://coq.inria.fr`

Agda, dependently typed functional programming language and proof assistant

`https://wiki.portal.chalmers.se/agda`

Lean, programming language and theorem prover

`https://lean-lang.org`

Roadmap

Connectives and constructors

Second-order logic and polymorphism

Higher-order logic, predicate logic and the λ cube

Classical logic and control operators

Proofs-as-programs for classical logic

CLASSICAL LOGIC	COMPUTATION
Axiomatic system	$\mathcal{C}$ calculus
Natural Deduction	$\lambda\mu$-calculus
Sequent system	$\overline{\lambda}\mu\widetilde{\mu}$-calculus

- Griffin, Felleisen, Filinsky 1990s (axiomatic)
  $\mathcal{C}$: *formulae-as-types notion of control,* `call/cc`
- Parigot 1992 (natural deduction)
  $\lambda\mu$: *algorithmic interpretation of classical logic*
- Curien, Herbelin 2000 (sequent)
  $\overline{\lambda}\mu\widetilde{\mu}$: *symmetric lambda calculus - duality of computation*

Sequent calculus

Intuitionistic Logic LJ, Classical Logic LK

- Axiom

$$(Ax) \qquad \frac{}{\Gamma, A \vdash A, \Delta}$$

- Rules

$$(\rightarrow_L) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash C, \Delta}{\Gamma, A \rightarrow B \vdash C, \Delta}$$

$$(\rightarrow_R) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}$$

$$(Cut) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash B, \Delta}{\Gamma \vdash B, \Delta}$$

A proof of the Peirce law

$$\vdash ((A \to B) \to A) \to A$$

$$\dfrac{\dfrac{\dfrac{}{A \vdash B, A}\,(ax)}{\vdash A \to B, A}\,(\to R) \qquad \dfrac{}{A \vdash A}\,(ax)}{\dfrac{(A \to B) \to A \vdash A}{\vdash ((A \to B) \to A) \to A}\,(\to R)}\,(\to L)$$

$\overline{\lambda}\mu\widetilde{\mu}$: symmetric lambda calculus

Syntax

$$\begin{array}{lrcl} \text{Term:} & t & ::= & x \mid \lambda x.t \mid \mu\alpha.c \\ \text{Co-term:} & e & ::= & \alpha \mid t \bullet e \mid \widetilde{\mu}x.c \\ \text{Command:} & c & ::= & < t \mid e > \end{array}$$

Reduction rules

$$\begin{array}{lrcl} (\lambda) & < \lambda x.t' \mid t \bullet e > & \rightarrow & < t' \mid \widetilde{\mu}x. < t \mid e > \\ (\mu) & < \mu\alpha.c \mid e > & \rightarrow & c[\alpha := e] \\ (\widetilde{\mu}) & < t \mid \widetilde{\mu}x.c > & \rightarrow & c[x := r] \end{array}$$

Failure of confluence - critical pair (non reparable)

$$< \mu\alpha. < y \mid \beta > \mid \widetilde{\mu}x. < z \mid \gamma >> \rightarrow < y \mid \beta >$$

$$< \mu\alpha. < y \mid \beta > \mid \widetilde{\mu}x. < z \mid \gamma >> \rightarrow < z \mid \gamma >$$

Duality of computation

call by name, call by value

$$< \mu\alpha.c \mid \widetilde{\mu}x.d > \rightarrow\ c[\alpha := \widetilde{\mu}x.d] \quad \textit{CBV}$$

$$< \mu\alpha.c \mid \widetilde{\mu}x.d > \rightarrow\ c[x := \mu\alpha.d] \quad \textit{CBN}$$

Two confluent subsystems $\overline{\lambda}\mu\widetilde{\mu}_{CBV}$ and $\overline{\lambda}\mu\widetilde{\mu}_{CBN}$

$\overline{\lambda}\mu\widetilde{\mu}$ with simple types

$$A ::= p \mid A \to A$$

$$(\to_L) \qquad \frac{\Gamma \vdash v : A; \Delta \qquad \Gamma; e : B \vdash C, \Delta}{\Gamma; v \bullet e : A \to B \vdash C, \Delta}$$

$$(\to_R) \qquad \frac{\Gamma, x : A \vdash v : B; \Delta}{\Gamma \vdash \lambda x.v : A \to B; \Delta}$$

$$(Cut) \qquad \frac{\Gamma \vdash v : A; \Delta \qquad \Gamma; e : A \vdash \Delta}{< v \mid e >: (\Gamma \vdash \Delta)}$$

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A; \Delta}\,(\mu) \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma\ \widetilde{\mu}x.c : A \vdash \Delta}\,(\widetilde{\mu})$$

- Pierce's law is inhabited ✓

$$\lambda x.\mu\alpha. < x \mid (\lambda y.\mu\beta. < y \mid \alpha >) \bullet \alpha >: ((A \to B) \to A) \to A$$

- Strong normalization of CBV and CBN:
  CPS translations of $\overline{\lambda}\mu\widetilde{\mu}_{CBV}$ and $\overline{\lambda}\mu\widetilde{\mu}_{CBN}$ into
  λ-calculus with simple types

F. Barbanera, S. Berardi
A Symmetric Lambda Calculus for "Classical" Program Extraction
TACS 1994: 495-515

P.-L. Curien, H. Herbelin
The duality of computation
ICFP 2000: 233-243

T. Griffin
A Formulae-as-Types Notion of Control
POPL 1990: 47-58

M. Parigot
Lambda-Mu-Calculus: An Algorithmic Interpretation of Classical Natural Deduction
LPAR 1992: 190-201

P. Wadler
Call-by-value is dual to call-by-name
ICFP 2003: 189-201

H. Herbelin, S. Ghilezan
An approach to call-by-name delimited continuations.
POPL 2008: 383-394

D. Dougherty, S. Ghilezan and P. Lescanne.
Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage.
Theoretical Computer Science 398: 114-128 (2008)