Introduction to Barendregt's Lambda Cube

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA, Serbia

Lecture 3

Oregon Programming Language Summer School
Eugene, June 2023

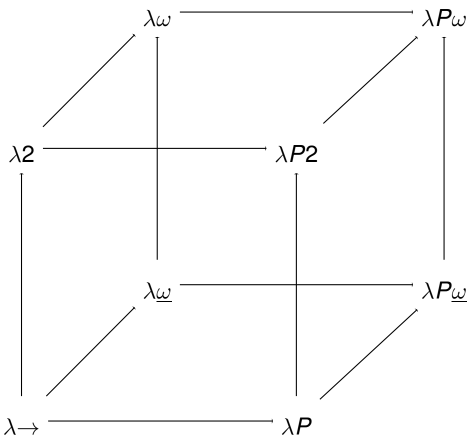


Figure: Lambda cube

- λ Untyped Lambda calculus
- $\lambda\!\to$ Simple types
- $\lambda 2$ Polymorphic types
- $\lambda\underline{\omega}$
- λP Dependent types
- Lambda cube
- Logic cube

Roadmap

λP dependent types

Lambda cube

Logic cube

Roadmap

λP dependent types

Lambda cube

Logic cube

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

Dependency - dependent types

4. types depending on terms - λP

$$A^n \to B$$

Notion of kinds extended to

$$A \text{ is a type}, \quad k \text{ is a kind} \Rightarrow A \ \to \ k \quad \text{is a kind}$$

- $A \to \star : \Box$

$$f : A \to \star \quad a : A \quad \Rightarrow \quad fa : \star \qquad (app)$$

$f(a)$ is a type depending on the term f (term dependent type)

$$\lambda x : A.f(x) : A \to \star \qquad (abstr)$$

λP dependent types

Terms and types

- Pseudo-expressions: terms and types

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid \mathcal{T}\mathcal{T} \mid \lambda\mathcal{V} : \mathcal{T}.\mathcal{T} \mid \Pi\mathcal{V} : \mathcal{T}.\mathcal{T}$$

- $\{\star, \square\} = \mathcal{S} \subseteq \mathcal{C}$ sorts $\lambda\underline{\omega}$
- statements $M : A$, where both $M, A \in \mathcal{T}$ $\lambda\underline{\omega}$
- bases linearly ordered $\Gamma = < x_1 : A_1, \ldots, x_n : A_n >$ $\lambda\underline{\omega}$

$$\begin{array}{rcl} \alpha : \star, x : \alpha & \vdash & x : \alpha \\ \alpha : \star & \vdash & \lambda x : \alpha.x : \alpha \to \alpha \end{array} \qquad \lambda\underline{\omega}$$

$$\begin{array}{rcl} \alpha : \star, x : \alpha & \vdash & x : \alpha \\ \alpha : \star & \vdash & \lambda x : \alpha.x : \Pi x : \alpha.\alpha \end{array} \qquad \lambda P$$

λP typing rules

(ax/sort) $\vdash \star : \Box$

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

(weak) $$\frac{\Gamma \vdash M : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash M : B} \text{ if } x \notin \Gamma$$

(λ) $$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash A \to B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

(app) $$\frac{\Gamma \vdash M : \Pi x : A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

(conv$_\beta$) $$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} A =_\beta B$$

(type/kind Π) $$\frac{\Gamma \vdash A : \star \quad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x : A.B : s}$$

λP examples

Example

$$\begin{array}{rcl}
A:\star & \vdash & (A\to\star):\square \\
A:\star,\ P:A\to\star,\ a:A & \vdash & Pa:\star \\
A:\star,\ P:A\to\star,\ a:A & \vdash & Pa\to\star:\square \\
A:\star,\ P:A\to\star & \vdash & \Pi a:A.Pa\to\star:\square \\
A:\star,\ P:A\to\star & \vdash & (\lambda a:A.\lambda x:Pa.x):(\Pi a:A.(Pa\to Pa))
\end{array}$$

λP and predicate logic

Curry-Howard, formulae as types, proofs as terms

AUTOMATH de Bruijn (1970)

- to represent mathematical theorems and their proofs
- interpretation of $\{\to \forall\}$ fragment of *PRED* (constructive) predicate logic

Translation

- P predicate on set (type) A is presented as $P : A \to \star$
- for $a \in A$, Pa is valid iff it is inhabited

$$\forall x \in A.Px \quad \text{is translated as} \quad \Pi x : A.Px$$

$$A \to B \quad \text{is translated as} \quad \Pi x : A.B \quad (A \to B)$$

Example

The formula

$$(\forall x \in A.\forall y \in A.Pxy) \to (\forall x \in A.Pxx)$$

is valid in PRED because its translation is inhabited in λP

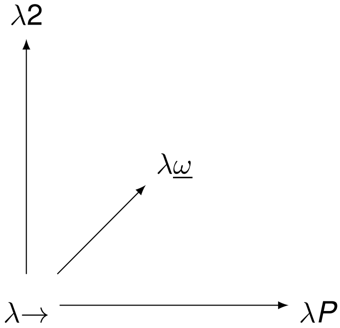
$$\begin{array}{lcl} A : \star, P : A \to A \to \star & \vdash & [\lambda z : (\Pi x : A.\Pi y : A.Pxy).\lambda x : A.zxx] : \\ & & ((\Pi x : A.\Pi y : A.Pxy) \to (\Pi x : A.Pxx)) \end{array}$$

Roadmap

λP dependent types

Lambda cube

Logic cube


λ2
λω
λ→
λP

Lambda cube language

Terms and types

- Pseudo-expressions: terms and types

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid \mathcal{T}\mathcal{T} \mid \lambda\mathcal{V} : \mathcal{T}.\mathcal{T} \mid \Pi\mathcal{V} : \mathcal{T}.\mathcal{T}$$

- $\mathcal{V}$ no distinction between type- and term-variables is made
- $A : B$ statements, $A, B \in \mathcal{T}$
- $x : A$ declarations, $x \in \mathcal{V},\ A \in \mathcal{T}$
- $\Gamma = < x_1 : A_1, \ldots, x_n : A_n >$ bases linearly ordered declarations
- $\Gamma \vdash A : B$ type assignments
- $\{\star, \square\} = \mathcal{S} \subseteq \mathcal{C}$ sorts (types, kinds)
- $s \in \{\star, \square\}$

Lambda cube

(ax) $\vdash \star : \square$

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

(weak) $$\frac{\Gamma \vdash M : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash M : B} \text{ if } x \notin \Gamma$$

(λ) $$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

(app) $$\frac{\Gamma \vdash M : \Pi x : A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

(conv_β) $$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} \; A =_\beta B$$

(Π) $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_2} \text{ if } (s_1, s_2) \in \mathcal{R}$$

Dependency

System	$\mathcal{R}$			
$\lambda\rightarrow$	$(\star,\star)$			
$\lambda 2$ (system F)	$(\star,\star)$	$(\Box,\star)$		
λP (LF)	$(\star,\star)$		$(\star,\Box)$	
$\lambda\underline{\omega}$	$(\star,\star)$			$(\Box,\Box)$
λP2	$(\star,\star)$	$(\Box,\star)$	$(\star,\Box)$	
$\lambda\omega$ (system Fω)	$(\star,\star)$	$(\Box,\star)$		$(\Box,\Box)$
λP$\underline{\omega}$	$(\star,\star)$		$(\star,\Box)$	$(\Box,\Box)$
λPω (λC, CC)	$(\star,\star)$	$(\Box,\star)$	$(\star,\Box)$	$(\Box,\Box)$

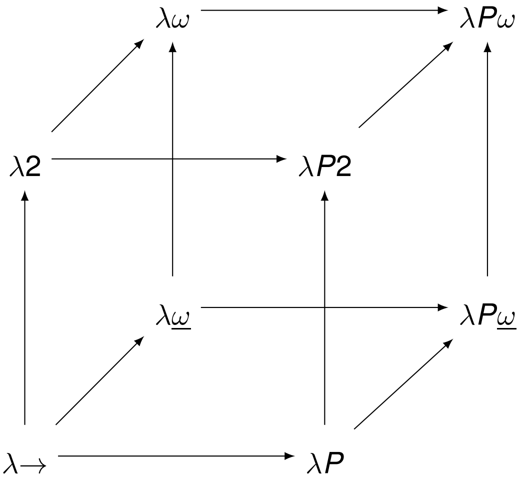


Figure: Lambda cube

Related systems

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 (1991)
λP2		Longo and Moggi (1988)
$\lambda\omega$	$\mathcal{F}\omega$	Girard (1972)
λPω	calculus of constructions	Coquand and Huet (1988)
λP$\underline{\omega}$		

Strong normalisation

Theorem
All eight systems of the cube are SN

Roadmap

λP dependent types

Lambda cube

Logic cube

Intuitionistic logics - Curry-Howard

PROP	proposition logic	$\lambda \to$
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)

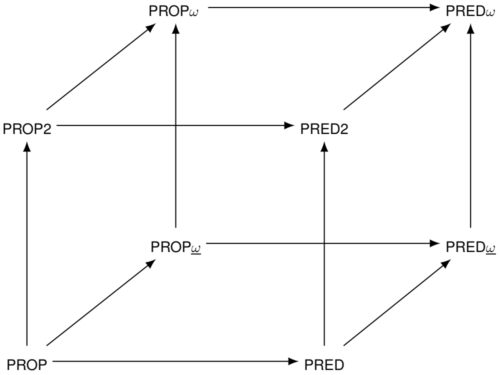


Figure: Logic cube

Theorem

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

Scientific Coincidence, late 1980s

- Lambda cube - Barendregt
- Pure Type Systems - Berardi, Terlouw

Pure Types Systems, PTS - Berardi, Terlouw (1989)

(sort) $\vdash s_1 : s_2$ if $(s_1, s_2) \in \mathcal{A}$

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

(weak) $$\frac{\Gamma \vdash M : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash M : B} \text{ if } x \notin \Gamma$$

(λ) $$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

(app) $$\frac{\Gamma \vdash M : \Pi x : A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

(conv_β) $$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} A =_\beta B$$

(Π) $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_3} \text{ if } (s_1, s_2, s_3) \in \mathcal{R}$$

Types in mathematics :

- $\hookrightarrow$ Cantor, Naive set theory
- $\hookrightarrow$ Russell paradox, $R = \{s \mid s \notin s\}$
- $\hookrightarrow$ Type theory, Russell, Whitehead, Ramsey
- $\hookrightarrow$ Constructive type theory, Martin-Löf
- $\hookrightarrow$ Homotopy type theory

Types in computation :

- ↪ eight systems of Barendregt's λ-cube
- ↪ type systems outside the cube: intersection, recursive types,...
- ↪ pure type systems

Types in concurrency:

- ↪ types in π-calculus
- ↪ session types, multiparty session types, behavioural types,...

A type system

- splits elements of a language (terms)
  - ↪ into sets (types, kinds, sorts,...)

- proves absence of certain undesired properties
  - ↪ meaningful computation
  - ↪ meaningful behaviour

- proves presence of desired properties
  - ↪ uniqueness of types, SN, expressiveness, ...
  - ↪ liveness, safety, deadlock freedom, ...

At OPLSS 2023

- COQ
- Logical relations
- Proof theory
- Implementing dependent types
- Program synthesis
- Category theory
- Program analysis
- HoTT