Formalizing Bishop Style Constructive Mathematics with Martin-Löf's Type Theory

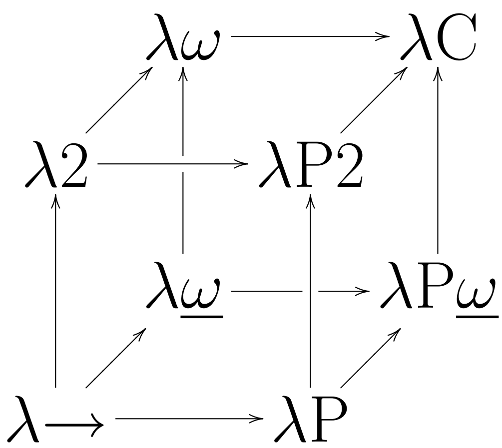
Frank Rosemeier
FernUniversität Hagen
D-58084 Hagen

e-mail: frank.rosemeier@fernuni-hagen.de

Motivation

Theory		Implementation
Bishop Style Constr. Math.	⟶	?
↓		
Martin-Löf's Type Theory	}	Alf Alfa/Agda
↓		
Barendregt's λ-cube	}	Coq Lego
↓		
Pure Type Systems	⟶	Yarrow

Barendregt's λ-cube



[Barendregt 1992]

$\lambda\!\rightarrow$	simply typed λ-calculus	[Church 1932–41]
$\lambda 2$	second order λ-calculus	[Girard 1972, Reynolds 1974]
$\lambda\omega$	higher order λ-calculus	[Girard 1972]
λC	calculus of constructions	[Coquand, Huet 1985–88]

λ-cube terms

Term	Meaning/Interpretation
$x_1, x_2, \ldots$	variables
(fa)	function application
$(\lambda x{:}A.b)$	function abstraction
$(\Pi x{:}A.B)$	(dependent) function type
$*$	sort of all types
$\square$	sort of all kinds
(fa)	$f(a)$
$(\lambda x{:}A.b)$	function $x \mapsto b$ for all x in A
$(\Pi x{:}A.B)$	$\{\, f \,\vert\, f(x)$ in B for all x in $A \,\}$
$*$	$\{\, \alpha \,\vert\, \alpha$ is a type $\}$
$\square$	$\{*, *\!\rightarrow\!*, \ldots\}$

β-conversion

Notation

$M[x := N]$ denotes *substitution* of N for all free occurrences of x in M.

Definition

Let *β-conversion* $=_\beta$ be the smallest equivalence relation on λ-cube terms such that

$$((\lambda x{:}M.N)L) =_\beta M[x := L],$$

$$M =_\beta M' \implies (LM) =_\beta (LM'),$$
$$M =_\beta M' \implies (MN) =_\beta (M'N),$$

$$M =_\beta M' \implies (\lambda x{:}L.M) =_\beta (\lambda x{:}L.M'),$$
$$M =_\beta M' \implies (\lambda x{:}M.N) =_\beta (\lambda x{:}M'.N),$$

$$M =_\beta M' \implies (\Pi x{:}L.M) =_\beta (\Pi x{:}L.M'),$$
$$M =_\beta M' \implies (\Pi x{:}M.N) =_\beta (\Pi x{:}M'.N).$$

General λ-cube rules

$\mathit{\Gamma}$ context $x_1{:}A_1, \ldots, x_n{:}A_n$ $(n \geqq 0)$

(Start) $$\frac{\mathit{\Gamma} \vdash A : s}{\mathit{\Gamma}, x{:}A \vdash x : A} \qquad (s \in \{*, \square\})$$

(Weaken) $$\frac{\mathit{\Gamma} \vdash A : s \quad \mathit{\Gamma} \vdash b : B}{\mathit{\Gamma}, x{:}A \vdash b : B}$$

(Conv) $$\frac{\mathit{\Gamma} \vdash a : A \quad \mathit{\Gamma} \vdash A' : s}{\mathit{\Gamma} \vdash a : A'} \quad \text{if } A =_\beta A'$$

(Π-intro) $$\frac{\mathit{\Gamma} \vdash (\Pi x{:}A.B) : s \quad \mathit{\Gamma}, x{:}A \vdash b : B}{\mathit{\Gamma} \vdash (\lambda x{:}A.b) : (\Pi x{:}A.B)}$$

(Π-elim) $$\frac{\mathit{\Gamma} \vdash f : (\Pi x{:}A.B) \quad \mathit{\Gamma} \vdash a : A}{\mathit{\Gamma} \vdash (fa) : B[x := a]}$$

(x fresh for $\mathit{\Gamma}$ and for A)

Axiom (Ax) $\vdash * : \Box$

Specific rules $(s_1, s_2 \in \{*, \Box\})$

(Π-form)
$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.B) : s_2}$$

(x fresh for Γ and for A)

λ-cube calculi

Calculus	Specific (s_1, s_2)-rules available			
$\lambda\!\rightarrow$	$(*,*)$			
λP	$(*,*)$	$(*,\Box)$		
λ2	$(*,*)$		$(\Box,*)$	
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
λP2	$(*,*)$	$(*,\Box)$	$(\Box,*)$	
$\lambda P\underline{\omega}$	$(*,*)$	$(*,\Box)$		$(\Box,\Box)$
$\lambda\omega$	$(*,*)$		$(\Box,*)$	$(\Box,\Box)$
$\lambda C = \lambda P\omega$	$(*,*)$	$(*,\Box)$	$(\Box,*)$	$(\Box,\Box)$

Notation

$(A{\rightarrow}B) \equiv (\Pi x{:}A.B)$ with x fresh for A, B

Derived rules $\quad (s, s_1, s_2 \in \{*, \Box\})$

(→-intro) $$\frac{\Gamma \vdash (A{\rightarrow}B) : s \quad \Gamma, x{:}A \vdash b : B}{\Gamma \vdash (\lambda x{:}A.b) : (A{\rightarrow}B)}$$

(x fresh for Γ and A)

(→-elim) $$\frac{\Gamma \vdash f : (A{\rightarrow}B) \quad \Gamma \vdash a : A}{\Gamma \vdash (fa) : B}$$

(→-form) $$\frac{\Gamma \vdash A : s_1 \quad \Gamma \vdash B : s_2}{\Gamma \vdash (A{\rightarrow}B) : s_2}$$

Rule	Dependency
$(*, *)$	objects depending on objects
$(*, \Box)$	types depending on objects
$(\Box, *)$	objects depending on types
$(\Box, \Box)$	types depending on types

Examples

In $\lambda{\to}$: $\alpha{:}*,\ \beta{:}* \vdash (\alpha{\to}\beta) : *$

$\alpha{:}* \vdash (\lambda x{:}\alpha.x) : (\alpha{\to}\alpha)$

In $\lambda 2$: $\vdash (\Pi\alpha{:}*.(\alpha{\to}\alpha)) : *$

$\vdash (\lambda\alpha{:}*.(\lambda x{:}\alpha.x)) : (\Pi\alpha{:}*.(\alpha{\to}\alpha))$

In $\lambda\underline{\omega}$: $\vdash (*{\to}*) : \Box$

$\vdash (\lambda\alpha{:}*.(\alpha{\to}\alpha)) : (*{\to}*)$

In λP: $\alpha{:}* \vdash (\alpha{\to}*) : \Box$

$\alpha{:}*,\ P{:}(\alpha{\to}*),\ x{:}\alpha \vdash (Px) : *$

Impredicativity

In $\lambda 2$: $\vdash (\Pi\alpha{:}*.\alpha) : *$

Martin-Löf's type theory

Judgements	Representation
$A\ set$	$A : Set$
$A = B$	$(SetEQ\ A\ B)$ true
$a \in A$	$a : (El\ A)$
$a = b \in A$	$(ElEQ\ A\ a\ b)$ true

where

$$\begin{aligned} Set &: * \\ SetEQ &: Set \rightarrow Set \rightarrow * \\ El &: Set \rightarrow * \\ ElEQ &: \Pi A{:}Set.((El\ A) \rightarrow (El\ A) \rightarrow *) \end{aligned}$$

(A type $\alpha{:}*$ is called *true* if it is inhabited, i. e. if $c : \alpha$ for some c.)

Judgements as Types Interpretation