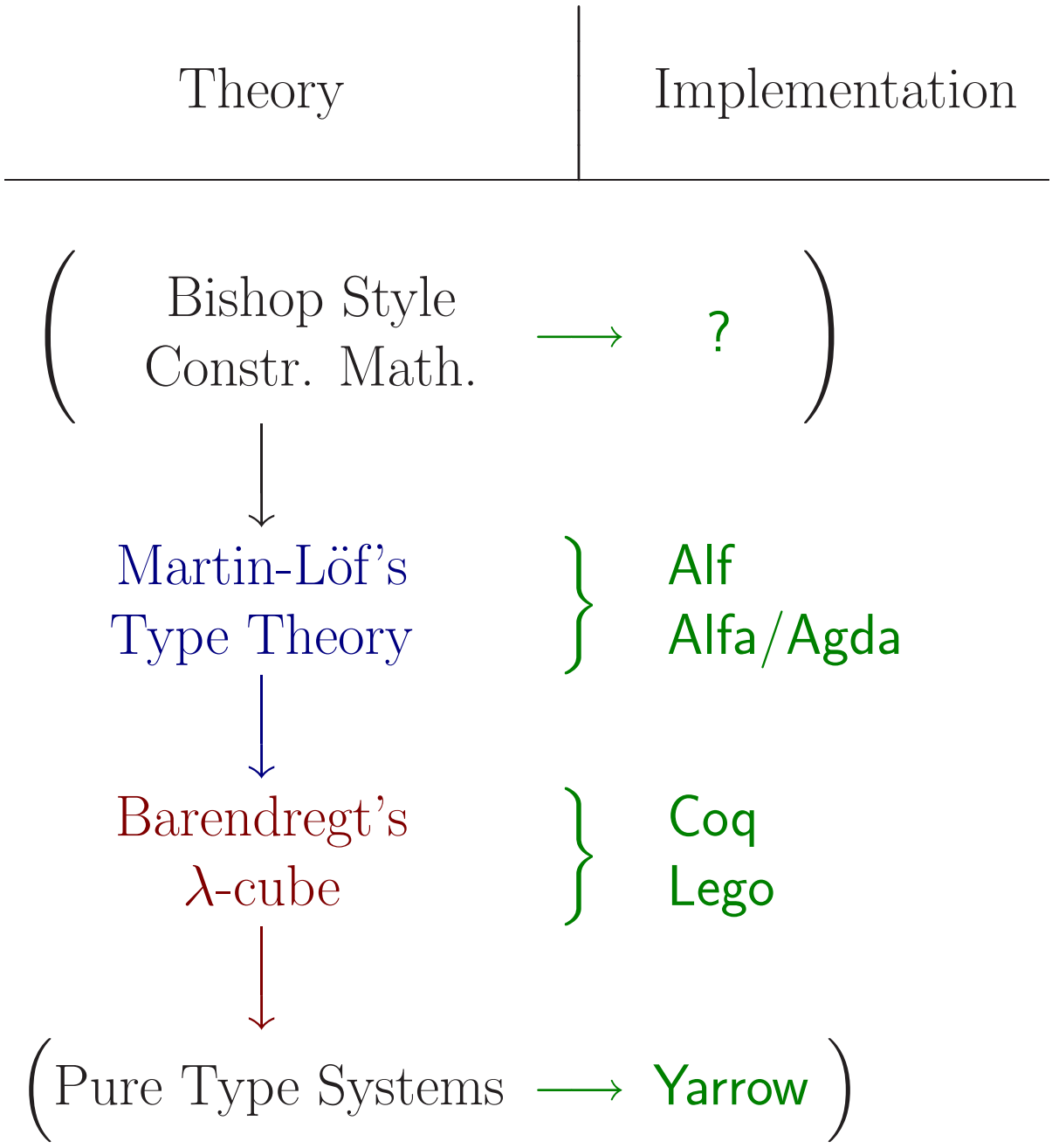
# Formalizing Bishop Style Constructive Mathematics with Martin-Löf's Type Theory

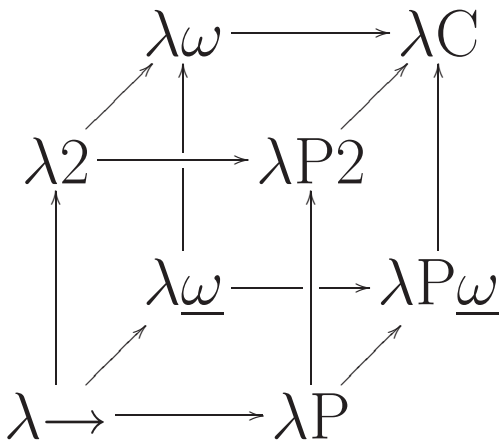
Frank Rosemeier
FernUniversität Hagen
D-58084 Hagen

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 $\lambda$-cube




[Barendregt 1992]

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

# $\lambda$-cube terms

| Term | Meaning/Interpretation |
|---|---|
| $x_1, x_2, \ldots$ | variables |
| $(f\,a)$ | function application |
| $(\lambda x{:}A.b)$ | function abstraction |
| $(\Pi x{:}A.B)$ | (dependent) function type |
| $*$ | sort of all types |
| $\Box$ | sort of all kinds |

| | |
|---|---|
| $(f\,a)$ | $f(a)$ |
| $(\lambda x{:}A.b)$ | function $x \mapsto b$ for all $x$ in $A$ |
| $(\Pi x{:}A.B)$ | $\{\, f \mid f(x)$ in $B$ for all $x$ in $A \,\}$ |
| $*$ | $\{\, \alpha \mid \alpha$ is a type $\,\}$ |
| $\Box$ | $\{*, * \longrightarrow *, \ldots\}$ |

# $\beta$-conversion

## Notation

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

## Definition

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

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

$$\begin{aligned} M =_\beta M' &\implies (LM) =_\beta (LM'), \\ M =_\beta M' &\implies (MN) =_\beta (M'N), \end{aligned}$$

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

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

## General $\lambda$-cube rules

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

(Start) $\qquad \dfrac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A} \qquad (s \in \{*, \Box\})$

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

(Conv) $\qquad \dfrac{\Gamma \vdash a : A \quad \Gamma \vdash A' : s}{\Gamma \vdash a : A'}$ if $A =_\beta A'$

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

($\Pi$-elim) $\qquad \dfrac{\Gamma \vdash f : (\Pi x{:}A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash (f\,a) : B[x := a]}$

($x$ fresh for $\Gamma$ 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 $\Gamma$ and for $A$)

## $\lambda$-cube calculi

| Calculus | Specific $(s_1, s_2)$-rules available | | | |
|---|---|---|---|---|
| $\lambda\!\rightarrow$ | $(*,*)$ | | | |
| $\lambda$P | $(*,*)$ | $(*,\Box)$ | | |
| $\lambda 2$ | $(*,*)$ | | $(\Box,*)$ | |
| $\lambda\underline{\omega}$ | $(*,*)$ | | | $(\Box,\Box)$ |
| $\lambda$P2 | $(*,*)$ | $(*,\Box)$ | $(\Box,*)$ | |
| $\lambda$P$\underline{\omega}$ | $(*,*)$ | $(*,\Box)$ | | $(\Box,\Box)$ |
| $\lambda\omega$ | $(*,*)$ | | $(\Box,*)$ | $(\Box,\Box)$ |
| $\lambda$C $=\lambda$P$\omega$ | $(*,*)$ | $(*,\Box)$ | $(\Box,*)$ | $(\Box,\Box)$ |

**Notation**

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

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

$$(\to\text{-intro}) \quad \frac{\Gamma \vdash (A \to B) : s \quad \Gamma, x{:}A \vdash b : B}{\Gamma \vdash (\lambda x{:}A.b) : (A \to B)} \quad (x \text{ fresh for } \Gamma \text{ and } A)$$

$$(\to\text{-elim}) \quad \frac{\Gamma \vdash f : (A \to B) \quad \Gamma \vdash a : A}{\Gamma \vdash (f\,a) : B}$$

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

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

## Examples

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

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

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

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

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

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

In $\lambda$P: $\qquad\qquad\qquad\qquad \alpha{:}* \vdash (\alpha{\to}*) : \Box$

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

## Impredicativity

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

# 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