$\lambda 2$

Henk Barendregt and Freek Wiedijk
assisted by Andrew Polonsky

Radboud University Nijmegen

March 19, 2012

λP

$$\frac{}{\vdash * : \Box}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma, y : B \vdash M : A}$$

$$\frac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$$

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x : A.\, B : s}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash A' : s}{\Gamma \vdash M : A'} \quad A =_\beta A'$$

$\lambda 2$

$$\frac{}{\vdash * : \Box}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma, y : B \vdash M : A}$$

$$\frac{\Gamma \vdash M \;:\; \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B \;:\; s}{\Gamma \vdash \lambda x : A.\, M \;:\; \Pi x : A.\, B}$$

$$\frac{\Gamma \vdash A : {\color{red}s} \qquad \Gamma, x : A \vdash B : {\color{red}*}}{\Gamma \vdash \Pi x : A.\, B : {\color{red}*}}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash A' : s}{\Gamma \vdash M : A'} \quad A =_\beta A'$$

second order propositional logic

second order logic

second order logic	=	quantification over predicates
	=	quantification over sets
	=	quantification over functions

quantification domains:

first order	objects
second order	objects, predicates on objects
third order	..., ..., predicates on predicates on objects
⋮	⋮
higher order	

second order propositional logic

quantification over predicates

↓

quantification over propositions

rules very similar to predicate logic
much simpler!

- no term variables
- no terms

'toy version' of predicate logic

syntax

$$A \quad ::= \quad a \mid A \to A \mid \forall a.\, A$$

A: formula

a: propositional variable

example:

$$\forall c.\, ((a \to b \to c) \to (b \to a \to c))$$

a and b: free variables

c: bound variable

free versus universally quantified variables

difference between:

$a \rightarrow a$	if $\overbrace{\text{it rains}}^{a}$, then it rains
$\forall a.\, a \rightarrow a$	for all propositions *a* holds: if *a*, then *a*

proof rules

$$\frac{}{\Gamma \vdash A} \; A \in \Gamma$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow I \qquad \frac{\Gamma \vdash A \rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow E$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall a.\, A} \forall I \qquad \frac{\Gamma \vdash \forall a.\, A}{\Gamma \vdash A[a := B]} \forall E$$

a not free in Γ

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \to b \to c^{\,x} \qquad a^{\,z}}{b \to c}\to E \qquad b^{\,y}}{c}\to E}{a \to c}\to I_{z}}{b \to a \to c}\to I_{y}}{(a \to b \to c) \to (b \to a \to c)}\to I_{x}}{\forall c.\,((a \to b \to c) \to (b \to a \to c))}\,\forall I_{c}$$

proof term:

$$\underbrace{\lambda c : *}_{\lambda 2!}.\, \lambda x : a \to b \to c.\, \lambda y : b.\, \lambda z : a.\, x\, z\, y$$

$$:$$

$$\Pi c : *.\,((a \to b \to c) \to (b \to a \to c))$$

polymorphic identity

$$\dfrac{\dfrac{a^x}{a \to a} \to I_x}{\forall a.\, a \to a} \forall I_a$$

proof term:

$$\lambda a : *.\, \lambda x : a.\, x \;\; : \;\; \Pi a : *.\, a \to a$$

polymorphic identity function

notations

here: PTS notation

variant $\lambda 2$ notations:

$$\lambda a : * \; \ldots \quad \text{also written as} \quad \Lambda a \; \ldots$$

$$\Pi a : * \; \ldots \quad \text{also written as} \quad \forall a \; \ldots$$

$$\Lambda a.\, \lambda x : a.\, x \quad : \quad \forall a.\, a \to a$$

$\lambda 2$

$\lambda 2$

PTS given by:

$$\begin{aligned} \mathcal{S} &= \{*, \Box\} \\ \mathcal{A} &= \{(*, \Box)\} \\ \mathcal{R} &= \{(*, *), (\Box, *)\} \end{aligned}$$

So, pseudo-terms:

$$M \quad ::= \quad * \mid \Box \mid x \mid MM \mid \lambda x : M.\, M \mid \Pi x : M.\, M$$

typing rules

PTS rules
with two product rules:

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.\, B : *}$$

$\Pi x : A.\, B$ always is of the form $A \to B$

$$\frac{\Gamma \vdash A : \Box \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.\, B : *}$$

A always is of the form $*$

simplified product rules

$$\frac{\Gamma \vdash A : * \qquad \Gamma \vdash B : *}{\Gamma \vdash A \rightarrow B : *}$$

$$\frac{\Gamma, a : * \vdash B : *}{\Gamma \vdash \Pi a : *. B : *}$$

stratified grammar

pseudo-terms of $\lambda 2$ can be *stratified*
pseudo-terms of λP can *not* be stratified

$$\Lambda a.\, M$$

$$\begin{array}{rcl} M & ::= & x \mid MM \mid MA \mid \lambda x : A.\, M \mid \lambda a : *.\, M \\ A & ::= & a \mid A \to A \mid \Pi a : *.\, A \end{array}$$

$$\forall a.\, A$$

polymorphism

$\lambda 2 =$ polymorphic lambda calculus

types not fixed $\rightsquigarrow$ type variables

polymorphism: $f : \Pi a : *.\, a \to a$

arbitrary type can be substituted for a

parametrized *family* of functions

overloading: f : nat $\to$ nat

f : real $\to$ real

don't need to be related

different functions with same name

polymorphic lists

polymorphic lists

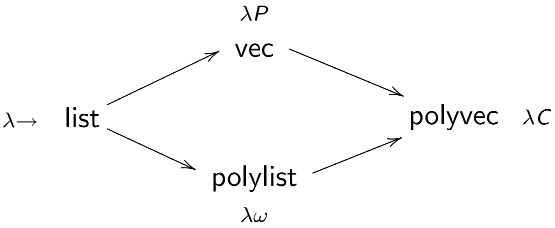
list	:	$*$	list of numbers
polylist A	:	$*$	list of objects of type $A : *$
polylist	:	$* \to *$	

note: $* \to *$ is not a $\lambda 2$ type!
$* \to *$ is a $\lambda \underline{\omega}$ type

$$\text{polynil} \quad : \quad \Pi a : *.\ \text{polylist } a$$
$$\text{polycons} \quad : \quad \Pi a : *.\ a \to \text{polylist } a \to \text{polylist } a$$

$\langle 0, 1 \rangle \quad \rightsquigarrow \quad$ polycons nat 0 (polycons nat (suc 0) (polynil nat))

polymorphic vectors



$$\begin{array}{rcl} \text{polyvec} & : & * \to \text{nat} \to * \\ \\ \text{polynil} & : & \Pi a : *.\ \text{polyvec}\ a\ 0 \\ \text{polycons} & : & \Pi a : *.\ \Pi n : \text{nat}.\ a \to \text{polyvec}\ a\ n \to \text{polyvec}\ a\ (\text{suc}\ n) \end{array}$$

$$\langle 0, 1 \rangle \quad \rightsquigarrow \quad \text{polycons nat (suc 0) 0 (polycons nat 0 (suc 0) (polynil nat))}$$

logical operations

logical operations

'impredicative definitions' in minimal second order logic:

$$
\begin{aligned}
A \to B & \qquad \text{primitive in } \lambda{\to} \\
\forall x.\, A & \qquad \text{primitive in } \lambda P \\
\forall a.\, A & \qquad \text{primitive in } \lambda 2
\end{aligned}
$$

$$
\begin{aligned}
\bot &:= \forall c.\, c \\
\top &:= \forall c.\, c \to c \\
\neg A &:= \forall c.\, A \to c \\
A \wedge B &:= \forall c.\, (A \to B \to c) \to c \\
A \vee B &:= \forall c.\, (A \to c) \to (B \to c) \to c \\
\exists x.\, A &:= \forall c.\, (\forall x.\, A \to c) \to c
\end{aligned}
$$

$\land$ introduction

$$\dfrac{\begin{matrix}\vdots \\ A\end{matrix} \qquad \begin{matrix}\vdots \\ B\end{matrix}}{A \land B}\ \land I$$

becomes:

$$\dfrac{\dfrac{\dfrac{\dfrac{\cancel{A \to B \to c}^{\,f} \qquad \begin{matrix}\vdots \\ A\end{matrix}}{B \to c}\ {\to}E \qquad \begin{matrix}\vdots \\ B\end{matrix}}{c}\ {\to}E}{(A \to B \to c) \to c}\ {\to}I_f}{\forall c.\,(A \to B \to c) \to c}\ \forall I_c$$

proof term for $\wedge$ introduction

$$\frac{\begin{array}{c}\vdots \\ \Gamma \vdash M : A\end{array} \qquad \begin{array}{c}\vdots \\ \Gamma \vdash N : B\end{array}}{\Gamma \vdash \langle M, N \rangle : A \wedge B} \wedge I$$

proof term:

$$\langle M, N \rangle \quad := \quad \lambda c : *. \lambda f : A \to B \to c. f\, M\, N$$

typed polymorphic version of:

$$\langle M, N \rangle \quad := \quad \lambda f . f\, M\, N$$

$\land$ elimination

$$\dfrac{\begin{array}{c}\vdots\\ A \land B\end{array}}{A}\ \land E_l$$

becomes:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \forall c.\,(A\to B\to c)\to c\end{array}}{(A\to B\to A)\to A}\ \forall E \qquad \dfrac{\dfrac{\cancel{A}^x}{B\to A}\ {\to}I_y}{A\to B\to A}\ {\to}I_x}{A}\ {\to}E$$

proof term for ∧ elimination

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash M : A \wedge B\end{array}}{\Gamma \vdash \pi_1 M : A} \wedge E_l$$

proof term:

$$\pi_1 M \quad := \quad M\, A\, (\lambda x : A.\, \lambda y : B.\, x)$$

typed polymorphic version of:

$$\pi_1 M \quad := \quad M\, (\lambda xy.x)$$

Curry-Howard

connective			data type	
$\bot$	falsity	$\leftrightarrow$	0	empty type
$\top$	truth	$\leftrightarrow$	1	unit type
		$\leftrightarrow$	2	Booleans
$A \rightarrow B$	implication	$\leftrightarrow$	$A \rightarrow B$	function type
$\forall x \in A.\, B$	universal	$\leftrightarrow$	$\Pi x : A.\, B$	dependent product
$A \wedge B$	conjunction	$\leftrightarrow$	$A \times B$	Cartesian product
$\exists x \in A.\, B$	existential	$\leftrightarrow$	$\Sigma x : A.\, B$	dependent sum
$A \vee B$	disjunction	$\leftrightarrow$	$A + B$	disjoint union

data types

Church numerals

$$\begin{array}{lcll} c_0 & := & \lambda a : *.\, \lambda f : a{\to}a.\, \lambda x : a.\, x & : \ \Pi a : *.\, (a \to a) \to a \to a \\ c_1 & := & \lambda a : *.\, \lambda f : a{\to}a.\, \lambda x : a.\, f\, x & : \ \Pi a : *.\, (a \to a) \to a \to a \\ c_2 & := & \lambda a : *.\, \lambda f : a{\to}a.\, \lambda x : a.\, f\, (f\, x) & : \ \Pi a : *.\, (a \to a) \to a \to a \\ c_3 & := & \lambda a : *.\, \lambda f : a{\to}a.\, \lambda x : a.\, f\, (f\, (f\, x)) & : \ \Pi a : *.\, (a \to a) \to a \to a \\ \ldots & & & \quad \ldots \end{array}$$

exactly the $\beta\bar{\eta}$ long normal forms of:

impredicative definition of the natural numbers

$$\mathsf{nat} \quad := \quad \Pi a : *.\, (a \to a) \to a \to a$$

$$\mathsf{suc} \quad := \quad \lambda n : \mathsf{nat}.\, \lambda a : *.\, \lambda f : a{\to}a.\, \lambda x : a.\, f\, (n\, a\, f\, x)$$

iteration and primitive recursion

$$\begin{aligned}
\text{iter} &: \Pi a : *.\ a \to (a \to a) \to (\text{nat} \to a) \\
\text{iter} &:= \lambda a : *.\ \lambda x : a.\ \lambda f : a{\to}a.\ \lambda n : \text{nat}.\ n\, a\, f\, x \\
\\
\text{rec} &: \Pi a : *.\ a \to (\text{nat} \to a \to a) \to (\text{nat} \to a) \\
\text{rec} &:= ?
\end{aligned}$$

implementing primitive recursion

$$\text{rec} \quad : \quad \Pi a : *.\, a \to (\text{nat} \to a \to a) \to (\text{nat} \to a)$$

$$\begin{aligned}\text{rec} \quad := \quad & \lambda a : *.\, \lambda x : a.\, \lambda f : \text{nat}{\to}a{\to}a.\, \lambda n : \text{nat}.\\ & \pi_2\,(n\,(\text{nat}{\times}a)\,(\lambda p : \text{nat}{\times}a.\, \langle \text{suc}\,(\pi_1 p), f\,(\pi_1 p)\,(\pi_2 p)\rangle)\langle 0, x\rangle)\end{aligned}$$

$$x \to f\,x \to f\,(f\,x) \to f\,(f\,(f\,x)) \to \ldots$$

$$x \to f\,0\,x \to f\,1\,(f\,0\,x) \to f\,2\,(f\,1\,(f\,0\,x)) \to \ldots$$

$$\langle 0, x\rangle \to \langle 1, f\,0\,x\rangle \to \langle 2, f\,1\,(f\,0\,x)\rangle \to \langle 3, f\,2\,(f\,1\,(f\,0\,x))\rangle \to \ldots$$

$$\cdots \to \langle n, r\rangle \to \langle \text{suc}\,n, f\,n\,r\rangle \to \ldots$$

$$\cdots \to p \to \langle \text{suc}\,(\pi_1 p), f\,(\pi_1 p)\,(\pi_2 p)\rangle \to \ldots$$

predecessor

recursive equations:

$$\begin{aligned} \text{pred}\,0 &= 0 = x \\ \text{pred}\,(\text{suc}\ n) &= n = f\ n\ (\text{pred}\ n) \end{aligned}$$

therefore arguments of rec:

$$\begin{aligned} x &= 0 \\ f &= \lambda n : \text{nat}.\, \lambda p : \text{nat}.\, n \end{aligned}$$

$$\begin{aligned} \text{pred} &:= \text{rec nat}\ x\ f \\ &= \text{rec nat}\ 0\ (\lambda n : \text{nat}.\, \lambda p : \text{nat}.\, n) \end{aligned}$$

induction

$$\begin{aligned}\mathsf{rec} \quad &: \quad \Pi a : *.\\ &\qquad a \to (\mathsf{nat} \to a \to a) \to (\mathsf{nat} \to a)\end{aligned}$$

$$\begin{aligned}\mathsf{ind} \quad &: \quad \Pi a : \mathsf{nat} \to *.\\ &\qquad a\, 0 \to (\Pi n : \mathsf{nat}.\, a\, n \to a\, (\mathsf{suc}\, n)) \to (\Pi n : \mathsf{nat}.\, a\, n)\end{aligned}$$

dependent version of primitive recursion
cannot be defined!

$$p(0) \to (\forall n \in \mathbb{N}.\, p(n) \to p(n+1)) \to \forall n \in \mathbb{N}.\, p(n)$$

mathematical induction

impredicative Cartesian product

$$\text{prod} := \lambda a:*.\,\lambda b:*.\,\Pi c:*.\,(a \to b \to c) \to c$$

$$\text{pair} : \Pi a:*.\,\Pi b:*.\; a \to b \to \text{prod}\; a\, b$$

$$\text{pair} := \lambda a:*.\,\lambda b:*.\,\lambda x:a.\,\lambda y:b.\; \lambda c:*.\,\lambda f:a{\to}b{\to}c.\, f\,x\,y$$

$$\pi_1 : \Pi a:*.\,\Pi b:*.\,\text{prod}\; a\, b \to a$$

$$\pi_1 : \lambda a:*.\,\lambda b:*.\,\lambda p:\text{prod}\; a\, b.\, p\, a\,(\lambda x:a.\,\lambda y:b.\,x)$$

$$\pi_2 : \Pi a:*.\,\Pi b:*.\,\text{prod}\; a\, b \to b$$

$$\pi_2 : \lambda a:*.\,\lambda b:*.\,\lambda p:\text{prod}\; a\, b.\, p\, b\,(\lambda x:a.\,\lambda y:b.\,y)$$

impredicative lists

$$\text{polylist} := \lambda a : *.\, \Pi b : *.\, b \to (a \to b \to b) \to b$$

$$\text{polynil} : \Pi a : *.\, \text{polylist}\ a$$

$$\text{polynil} := \lambda a : *.\ \lambda b : *.\, \lambda x : b.\, \lambda f : a{\to}b{\to}b.\, x$$

$$\text{polycons} : \Pi a : *.\, a \to \text{polylist}\ a \to \text{polylist}\ a$$

$$\text{polycons} := \lambda a : *.\, \lambda h : a.\, \lambda t : \text{polylist}\ a.\ \lambda b : *.\, \lambda x : b.\, \lambda f : a{\to}b{\to}b.\, f\ h\,(t\ b\ x\ f)$$

$$\text{polylist\_rec} : \ldots$$

limitations

with impredicative definitions of $\bot$ and $=$ and nat:

the type of ind is not inhabited

$0 \neq 1$ is not provable

inductive definitions $\rightsquigarrow$ *next lecture*

impredicativity

impredicative definitions

= definitions that quantify over a domain containing
the defined object

comprehension in ZF set theory is impredicative
comprehension in second order logic is impredicative

$$B \;:=\; \{x \in A \mid \phi(x)\}$$

$$\ldots \forall X \ldots$$

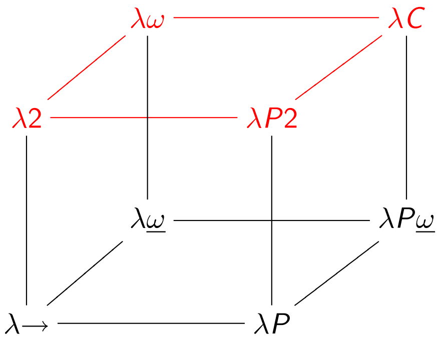
X can also be B

ACA_0 = 'limit' of predicative mathematics
= second order theory of arithmetic
comprehension: only $\phi(x)$ without second order quantifiers

$\leadsto$ *FOM mailing list*

impredicativity in the lambda cube

the four systems in the top plane are impredicative



is impredicativity bad?

- *impredicativity of ZF set theory*

  vague philosophical worry

  might be the cause of ZF being inconsistent
  no actual problems known

- *impredicativity in the lambda cube*

  vague philosophical worry

  is inconsistent when adding *classical axioms*
  only consistent because of intuitionism

  interpreting $A \to B$ as functions does not work
  $\Pi x : A.\, B$ 'is' the *intersection* $\bigcap_{x \in A} B$

variations in type theory

- intuitionistic or classical?

$$A \lor \neg A$$

- predicative or impredicative?
- intensional or extensional?

$$(\forall x.\, f(x) = g(x)) \to f = g$$

- no choice or choice?

$$(\forall x.\, \exists y.\, p(x, y)) \to (\exists f.\, \forall x.\, p(x, f(x)))$$

impredicativity in $\lambda 2$

$$2 : * \vdash \Pi a : *.\, a \to 2 \ : \ \overset{?}{*}$$

Coq notation: $* \ \rightsquigarrow \ \mathsf{Set}$

$$U := \prod_{a \in \mathsf{Set}} \wp(a) \ \in \ \mathsf{Set}$$

$\wp(a) =$ the powerset of a

$$\mathsf{Set} = \{X_0, X_1, \ldots, U, \ldots\}$$

$$U \ = \ \wp(X_0) \times \wp(X_1) \times \cdots \times \wp(U) \times \ldots$$

$\wp(U)$ is too big!

Cantor's diagonalisation

$$U \quad := \quad \Pi a : *.\ a \to 2$$

elements of U are functions u with $u(a) \subseteq a$ for each set a

define $u_\Delta \in U$ by:

$$u_\Delta(a) \ := \begin{cases} \{u \in U \mid u \notin u(U)\} & \text{if } a = U \\ \emptyset & \text{if } a \neq U \end{cases}$$

(intuitionistically we do not have $a = U \vee a \neq U$)

$$u_\Delta \in u_\Delta(U) \iff u_\Delta \in \{u \in U \mid u \notin u(U)\} \iff u_\Delta \notin u_\Delta(U)$$

impredicativity in the Coq proof assistant

pCIC = predicative Calculus of (Co)Inductive Constructions

two variants of $*$:

$*_p$	Prop	impredicative
$*_s$	Set	predicative

$$2 : *_p \vdash \Pi a : *_p .\ a \to 2 \ : \ *_p$$

$$2 : *_s \vdash \Pi a : *_s .\ a \to 2 \ : \ \Box$$

recap

1. second order propositional logic
2. $\lambda 2$
3. polymorphic lists
4. logical operations
5. data types
6. impredicativity