Problems in type theory

Henk Barendregt*

To Lidia on her birthday

February 4, 1998

Abstract

In this paper the reader will be introduced to type theories (predicative and impredicative, with and without inductive types) by a short section giving theoretical background and by another section with exercises about the calculus of constructions and its fine structure the lambda cube.

1. Background

Mathematics consists of defining, reasoning and computing. Type systems are designed to represent mathematics in such a way that it can be verified automatically whether the definitions are well-formed, the derivations are correct and the computations are performed by a simple underlying reduction system. This verification method is expected to be useful for the development of mathematics and the teaching of it. Also for the verification of hardware and software systems. See Barendregt [1996] and [1997] for a discussion and more information.

Type systems as presented are based on work of Russell and Whitehead [1910–13], Church [1940], de Bruijn [1970], Scott [1970], Girard [1972], Martin-Löf [1984], Coquand and Huet [1988] and Barendregt [1992]. The use of type theory for proof-checking has been initiated by N.G. de Bruijn [1970], see also Nederpelt et al. [1994].

Pure Type Systems

A Type Systems (TS) consist of a Pure Type System (PTS) together with Inductive Types (IT). A PTS is completely determined by its *specification* and the allowed notion of reduction $\mathbf{R}$. Usually $\mathbf{R} = \beta\delta$. Here β is the well-known notion of β-reduction

$$(\lambda x{:}A.b)a \rightarrow_\beta b[x{:}=a];$$

*Faculty of Mathematics and Computer Science, Nijmegen University, The Netherlands. E-mail: henk@cs.kun.nl. Author is partially sponsored by the European Community HCM Network ERB CHRX CT 920046 (Typed Lambda Calculus) and Esprit Working Group 21900 (Types) and by the Australian Research Council, grant A49702489 (Type Theory and Illative Combinatory Logic). I thank Milena Stephanova for useful discussions and Mirna Bognar and Freek Wiedijk for corrections.

δ-reduction is unfolding defined notions, so if `square` is defined as $\lambda x{:}\mathsf{Nat}.x^2$, then

$$\mathtt{square} \to_\delta \lambda x{:}\mathsf{Nat}.x^2.$$

Inductive types come together with axiomatically given terms R for both induction and primitive recursion over that type. Induction will be given by the type of R and primitive recursion by the notion of reduction ι for terms containing R. A term P *is in* $\mathbf{R}$ *normal form* ($\mathbf{R}$-nf) iff for no term Q one has $P \to_{\mathbf{R}} Q$. If $P \twoheadrightarrow_{\mathbf{R}} Q$ and Q is in $\mathbf{R}$-nf, then we say that P *has* Q *as* $\mathbf{R}$*-nf*. Here $\twoheadrightarrow_{\mathbf{R}}$ is the transitive reflexive closure of $\to_{\mathbf{R}}$.

PTSs are denoted by $\lambda_{\mathbf{R}}\mathfrak{A}$ where $\mathfrak{A}$ is the specification and $\mathbf{R}$ is the underlying notion of reduction. Usually $\mathbf{R}$ is $\beta\delta$ in which case we write simply $\lambda\mathfrak{A}$ for the system. TSs are denoted by $\lambda_{\mathbf{R}}^{\mathtt{Ind}}\mathfrak{A}$, where usually $\mathbf{R}$ is $\beta\delta\iota$. In this case we write simply $\lambda^{\mathtt{Ind}}\mathfrak{A}$ for the system.

Example of a specification of a PTS.

λP

Sorts	$*, \Box$
Axioms	$* : \Box$
Rules	$(*, *, *), (*, \Box, \Box)$

or in a more compact way

λP

$\mathcal{S}$	$*, \Box$
$\mathcal{A}$	$* : \Box$
$\mathcal{R}$	$(*, *), (*, \Box)$

The way this determines a formal system will be explained now. Pseudo-terms for PTSs are determined by the following abstract grammar.

$$\mathcal{T} ::= \mathtt{var} \mid \mathtt{const} \mid \mathcal{T}\mathcal{T} \mid \lambda\mathtt{var}{:}\mathcal{T}.\mathcal{T} \mid \Pi\mathtt{var}{:}\mathcal{T}.\mathcal{T}$$

Some of the constants are called sorts and are declared in the specification as the set $\mathcal{S}$. An axiom is of the form $c : s$ with c a constant and $s{\in}\mathcal{S}$. The set of axioms is denoted by $\mathcal{A}$. A *rule* is of the form (s_1, s_2, s_3), with $s_i{\in}\mathcal{S}$; the set of rules is denoted by $\mathcal{R}$.

The rules of PTSs consist of general rules valid for all systems and specific ones, depending on the choice of specification and notion of reduction. A *typing statement* is of the form $A : B$ with $A, B{\in}\mathcal{T}$ A *declaration* is a typing statement $x : B$, with x a variable. A *context* is a list of declarations. A statement of a PTS is of the form

$$\Gamma \vdash A : B,$$

with Γ a context. This is pronounced as "Γ yields A in B". If Γ is empty then this is written as $\vdash A : B$.

Now we present the axiom and rules of a PTS specified by $\mathfrak{A} = (\mathcal{S}, \mathcal{A}, \mathcal{R})$.

Axiom	$\vdash c : s$	$c : s{\in}\mathcal{A}$;
Start	$\dfrac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$	$s{\in}\mathcal{S}$, x fresh;
Weakening	$\dfrac{\Gamma \vdash A : s \quad \Gamma \vdash B : C}{\Gamma, x{:}A \vdash B : C}$	$s{\in}\mathcal{S}$, x fresh;
Application	$\dfrac{\Gamma \vdash F : \Pi x{:}A.B \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x := a]}$	
Product	$\dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3}$	$(s_1, s_2, s_3){\in}\mathcal{R}$;
Abstraction	$\dfrac{\Gamma, x{:}A \vdash B : C \quad \Gamma \vdash \Pi x{:}A.C : s}{\Gamma \vdash \lambda x{:}A.B : \Pi x{:}A.C}$	$s{\in}\mathcal{S}$;
Conversion	$\dfrac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}$	$B =_R B'$, $s{\in}\mathcal{S}$

Axiom and rules for $\lambda_{\mathbf{R}}\mathfrak{A}$ with $\mathfrak{A} = (\mathcal{S}, \mathcal{A}, \mathcal{R})$

Some PTSs are having as sorts $\mathcal{S} = \{*, \Box\}$, as axiom set $\mathcal{A} = \{* : \Box\}$ and rules determined by the following table. Here a rule (s_1, s_2) stands for (s_1, s_2, s_2).

System	Rules			
$\lambda\!\to$	$(*, *)$			
$\lambda 2$	$(*, *)$	$(\Box, *)$		
$\lambda\underline{\omega}$	$(*, *)$		$(\Box, \Box)$	
$\lambda\omega$	$(*, *)$	$(\Box, *)$	$(\Box, \Box)$	
λP	$(*, *)$			$(*, \Box)$
$\lambda P2$	$(*, *)$	$(\Box, *)$		$(*, \Box)$
$\lambda P\underline{\omega}$	$(*, *)$		$(\Box, \Box)$	$(*, \Box)$
$\lambda P\omega = \lambda C$	$(*, *)$	$(\Box, *)$	$(\Box, \Box)$	$(*, \Box)$

These systems form in a natural way the so-called λ-cube, see figure 1. This cube is a fine-structure of its strongest system λC, a version of the calculus of constructions, see Barendregt [1992] for more information.

1.1. Example. The system $\lambda 2$ is specified as follows.

$\lambda 2$		
	$\mathcal{S}$	$*, \Box$
	$\mathcal{A}$	$* : \Box$
	$\mathcal{R}$	$(*, *), (\Box, *)$

1.2. Definition. In any of the systems of the λ-cube we define the following. Let Γ be a context and $A{\in}\mathcal{T}$.

$$
\begin{tikzcd}
& \lambda\omega \arrow[rr] & & \lambda\mathrm{P}\omega \\
\lambda 2 \arrow[ur] \arrow[rr] & & \lambda\mathrm{P}2 \arrow[ur] & \\
& \lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \lambda\mathrm{P}\underline{\omega} \arrow[uu] \\
\lambda\to \arrow[uu] \arrow[ur] \arrow[rr] & & \lambda\mathrm{P} \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Figure 1: The λ-cube.

(i) A is a Γ-*type* iff $\Gamma \vdash A : *$.
(ii) A is a Γ-*term* iff $\Gamma \vdash A : B$ for some $B \in \mathcal{T}$
(iii) A is a Γ-*element* iff $\Gamma \vdash A : B$ for some Γ-type B.
(iv) We say that A is Γ-provable, notation $\Gamma \vdash A$, iff $\Gamma \vdash B : A$ for some $B \in \mathcal{T}$.

NOTATION. Write in a system of the λ-cube $\Gamma \vdash A$ iff $\Gamma \vdash p : A$ for some $p \in \mathcal{T}$.

1.3. DEFINITION. A *function-space* is defined by

$$A \to B \equiv \Pi x{:}A.B,$$

with x fresh, i.e. not occurring in B.

NOTATION. We write

$$A_1 \to A_2 \to \ldots \to A_{n-1} \to A_n = A_1 \to (A_2 \to \ldots \to (A_{n-1} \to A_n)..).$$

1.4. EXERCISE. Show that the following rules about function spaces are derivable from the axioms and rules of a PTS.

1. $$\frac{\Gamma \vdash F : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B}$$

2. $$\frac{\Gamma, x{:}A \vdash b : B \mid \Gamma \vdash A \to B : s}{\Gamma \vdash \lambda x{:}A.b : A \to B}$$

Inductive Types

Inductive types are introduced by a typical (but simple) example. There are more involved inductive types that will not be treated here.

1.5. EXAMPLE. Nat^I is the inductive type

$$\mathsf{Nat}^I = (\mathtt{Ind}\ X)(\mathtt{zero} : X | \mathtt{suc} : X{\rightarrow}X).$$

This means that there are constructors $\mathtt{zero} : \mathsf{Nat}^I$ and $\mathtt{suc} : \mathsf{Nat}^I{\rightarrow}\mathsf{Nat}^I$. In intuitive terms Nat^I is freely generated as follows

$$\mathsf{Nat}^I ::= \ \mathtt{zero} \,|\, \mathtt{suc}\ \mathsf{Nat}^I$$

Write $\underline{0} = \mathtt{zero}$ and $\underline{n+1} = \mathtt{suc}\ \underline{n}$, so

$$\vdash \underline{n} : \mathsf{Nat}^I$$

for $n : \mathbb{N}$. This inductive type comes axiomatically with an inhabitant for induction $\mathsf{R} = \mathsf{R}_{\mathsf{Nat}^I}$ satisfying

$$\mathsf{R} : \Pi P{:}\mathsf{Nat}^I{\rightarrow}*.[P\ \mathtt{zero}{\rightarrow}(\forall x{:}\mathsf{Nat}^I.Px{\rightarrow}P(\mathtt{suc}\ x)){\rightarrow}\forall x{:}\mathsf{Nat}^I.Px],$$

and the reduction rules

$$\begin{aligned} \mathsf{R}Pab\,\mathtt{zero} &\rightarrow_\iota a; \\ \mathsf{R}Pab\,(\mathtt{suc}\ x) &\rightarrow_\iota bx(\mathsf{R}Pabx). \end{aligned}$$

The type of R actually exists only in the systems including product rules $(*,\Box),(\Box,*)$, therefore in $\lambda^I{\rightarrow}$ the recursion operator for inductive types have to be interpreted as 'parametric'. For $\mathsf{R} = \mathsf{R}(P[x])$ one has

$$\mathsf{R} : P[\mathtt{zero}]{\rightarrow}(\forall x{:}\mathsf{Nat}^I.P[x]{\rightarrow}P[\mathtt{suc}\ x]){\rightarrow}\forall x{:}\mathsf{Nat}^I.P[x]$$

for parametric $P[x]$ with

$$x : \mathsf{Nat}^I \vdash P[x] : *.$$

Here $P[x]$ denotes a term in which x may occur.

Without proof we state the following.

1.6. PROPOSITION. *Let $\lambda\mathfrak{A}$ be one of the systems of the λ-cube, with or without inductive types. Then the following holds.*

(i) $\Gamma \vdash_{\lambda\mathfrak{A}} P : A \ \Rightarrow\ P$ *and* A *have a* $\beta\delta\iota$*-nf.*

(ii) $\Gamma \vdash_{\lambda\mathfrak{A}} P : A \ \&\ P \twoheadrightarrow_{\beta\delta\iota} Q \ \Rightarrow\ \Gamma \vdash Q : A.$

Application of TSs

Type systems are being applied because they can represent mathematics: definitions, reasoning and computations all can be captured by them in an integrated way. If in informal mathematics one proves a statement A in a situation Γ, then this can be translated into $[\Gamma] \vdash p : [A]$ in the appropriate TS. Here p is to be regarded as the proof of $[A]$. In fact it is a precise expression isomorphic to a natural deduction style proof.

Typing statements in a TS can be verified by a simple program. Indeed,

$$\Gamma \vdash A : B \ \Leftrightarrow \ \mathtt{type}_\Gamma(A) =_{\mathbf{R}} B.$$

Here $\mathtt{type}_\Gamma$ is a simple function so the main expense is in determining the convertibility $\mathtt{type}_\Gamma(A) =_{\mathbf{R}} B$.

2. Exercises on Type systems

The various systems of the λ-cube correspond to various logics. $\lambda\!\rightarrow, \lambda 2$ and $\lambda\omega$ correspond to the intuitionistic first-, second- and higher-order propositional logics. The system $\lambda\underline{\omega}$ is an auxiliary system that is natural from the programming point of view, as it allows constructors building complex types as first class citizens. (For example $F\alpha = \alpha\!\rightarrow\!\alpha\!\rightarrow\!\alpha$.) Actually only the $\{\rightarrow\}$ fragment of these logics can be captured in the type system $\lambda\!\rightarrow$. Using inductive types also the other connectives can be dealt with. Another possibility is to use the second-order definition of other connectives in the systems $\lambda 2, \lambda\omega$.

The systems $\lambda P, \lambda P2$ and λC correspond to first-, second- and higher-order predicate logic, $\lambda P\underline{\omega}$ again being an auxiliary system. The correspondence between ordinary logic and provability in these type systems is not exact: more statements are valid in the TSs. By defining slightly more complicated TSs, the systems of the so-called logic cube (or L-cube), the correspondence becomes exact. We only give one example.

$\lambda\mathtt{Pred2}$	$\mathcal{S}$	$*^p, *^s, *^f, \Box^p, \Box^s$
	$\mathcal{A}$	$*^p : \Box^p, *^s : \Box^s$
	$\mathcal{R}$	$(*^p, *^p), (*^s, *^p), (*^s, \Box^p),$ $(*^s, *^s, *^f), (*^s, *^f, *^f), (\Box^p, *^p)$

This system corresponds exactly to second order logic, see Barendregt [1992] and Fujita and Tonino [1992] (and Geuvers [1993] for a correction).

Remember that all systems $\lambda\mathfrak{A}$ have to be read as $\lambda_{\beta\delta}\mathfrak{A}$ and all systems $\lambda^{\mathtt{Ind}}\mathfrak{A}$ as $\lambda^{\mathtt{Ind}}_{\beta\delta\iota}\mathfrak{A}$.

$\boxed{\lambda\!\rightarrow}$

2.1. Exercise. Find inhabitants.

1. $\alpha{:}* \vdash\, ? : \alpha\!\rightarrow\!\alpha;$

2. $\alpha, \beta{:}* \vdash\, ? : \alpha\!\rightarrow\!\beta\!\rightarrow\!\alpha;$

3. $\alpha,\beta{:}*,\gamma{:}* \vdash ? : (\alpha{\to}\beta{\to}\gamma){\to}(\alpha{\to}\beta){\to}\alpha{\to}\gamma$;

4. $\alpha,\beta{:}* \vdash ? : (\alpha{\to}\alpha{\to}\beta){\to}\alpha{\to}\beta$.

2.2. EXERCISE. Show that for no context Γ and types ? and ?? one has the following
$$\Gamma \vdash \lambda x{:}?.xx : ??.$$

2.3. EXERCISE. For $\alpha{:}*$ define $\sim\alpha = \alpha{\to}\gamma$, for some fresh $\gamma{:}*$. Show the following.

1. $\alpha,\beta{:}*,\gamma{:}* \vdash (\alpha{\to}\beta){\to}(\sim\beta{\to}\sim\alpha)$;

2. $\alpha{:}*,\gamma{:}*,\texttt{exfalso} : \gamma{\to}\alpha \vdash {\sim\sim}\,({\sim\sim}\,\alpha{\to}\alpha)$.

2.4. EXERCISE. Let for $\alpha : *$ the type $\mathsf{Nat}^\alpha \equiv (\alpha{\to}\alpha){\to}(\alpha{\to}\alpha)$ be defined. Note that for $c_n^\alpha \equiv \lambda f{:}(\alpha{\to}\alpha)\lambda x{:}\alpha.f^n x$ the well-known Church numeral one has
$$\alpha{:}* \vdash c_n^\alpha : \mathsf{Nat}^\alpha.$$
Construct a term F such that $\alpha : * \vdash F : \mathsf{Nat}^\alpha{\to}\mathsf{Nat}^\alpha$ and
$$Fc_n^\alpha \twoheadrightarrow_\beta c^\alpha_{n^3+3n^2+1}.$$
[Hint. First construct terms `Plus` and `Times` representing addition and multiplication.]

$\boxed{\lambda^{\texttt{Ind}}{\to}}$

2.5. EXERCISE. Show that using $\mathsf{R} = \mathsf{R}_{\mathsf{Nat}^I}$ one can define for every type σ an R^σ such that
$$\mathsf{R}^\sigma : \sigma{\to}(\mathsf{Nat}^I{\to}\sigma{\to}\sigma){\to}\mathsf{Nat}^I{\to}\sigma$$
satisfying the reduction properties
$$\begin{array}{rcl}\mathsf{R}^\sigma ab\,\texttt{zero} & \twoheadrightarrow_{\beta\iota} & a;\\ \mathsf{R}^\sigma ab\,(\texttt{suc}\ x) & \twoheadrightarrow_{\beta\iota} & bx(\mathsf{R}^\sigma abx).\end{array}$$

2.6. EXERCISE. Let f be the modified Ackermann function (that is not primitive recursive) defined by
$$\begin{array}{rcl}f(0,n) & = & n+1,\\ f(m+1,0) & = & f(m,1),\\ f(m+1,n+1) & = & f(m,f(m+1,n)).\end{array}$$
Find a term F such that $\vdash F : \mathsf{Nat}^I{\to}\mathsf{Nat}^I{\to}\mathsf{Nat}^I$ and
$$F\ \underline{n}\ \underline{m} \twoheadrightarrow_{\beta\iota} \underline{f(n,m)}.$$
Explain why this seems paradoxical but is not really so.

2.7. EXERCISE. Define

$$\mathsf{Bool}^I = (\mathtt{Ind}X)(\mathsf{true}^I : X | \mathsf{false}^I : X).$$

Construct using $\mathsf{R} = \mathsf{R}_{\mathsf{Bool}^I}$ terms representing $\&, \vee, \Rightarrow, \neg$.

2.8. EXERCISE. Define

$$\perp^I = (\mathtt{Ind}\ X)().$$

Prove that

$$\alpha : * \vdash \perp^I \to \alpha.$$

$\boxed{\lambda 2}$

We use the notation $\forall \alpha.A$ to denote $\Pi \alpha{:}*.A$.

2.9. EXERCISE. Prove $\vdash \forall \alpha. \perp^2 \to \alpha$. Here $\perp^2 \equiv \forall \alpha.\alpha$.

2.10. EXERCISE. Define $\neg A \equiv A \to \perp^I$. Show that the following types are inhabited.

(i) $\forall \alpha.\alpha \to \neg\neg\alpha$.

(ii) $\forall \alpha.\neg\neg\neg\alpha \to \neg\alpha$.

(iii) $\forall \alpha.\neg\neg[\neg\neg\alpha \to \alpha]$.

2.11. EXERCISE. Prove the following.

1. $\alpha, \beta{:}* \vdash \alpha \& \beta \to \beta$.

2. $\alpha, \beta{:}* \vdash \alpha \to \alpha \vee \beta$.

Here

$$\begin{aligned}
\alpha \& \beta &= \forall \gamma.(\alpha \to \beta \to \gamma) \to \gamma; \\
\alpha \vee \beta &= \forall \gamma.(\alpha \to \gamma) \to (\beta \to \gamma) \to \gamma.
\end{aligned}$$

These are different from the operators with the same names in 2.7.

2.12. EXERCISE. Find a term P such that

$$\alpha, \beta{:}* \vdash P : \alpha \to \beta \to \alpha \& \beta;$$

Write $[x, y] \equiv Pxy$. Hence

$$x{:}\alpha, y{:}\beta \vdash [x, y] : \alpha \& \beta.$$

2.13. EXERCISE. Find the right expressions for ?.

1. $\beta{:}* \vdash \lambda x{:}(\forall \alpha.(\alpha \to \beta)).x?x : (\forall \alpha.(\alpha \to \beta)) \to \beta$.

2. $\beta{:}* \vdash \lambda x{:}(\forall \alpha.(\alpha \to \beta)).x\beta? : (\forall \alpha.(\alpha \to \beta)) \to \beta$.

2.14. EXERCISE. Define $\mathsf{c}_n \equiv \lambda \alpha{:}*.c_n^\alpha$, where c_n^α is defined in exercise 2.4.

1. Construct a type Nat^2 such that for all $n{:}\mathbb{N}$ one has $\vdash \mathsf{c}_n : \mathsf{Nat}^2$.

2. Construct a recursor R such that

$$\mathsf{R} : \forall\alpha.\alpha{\to}(\mathsf{Nat}^2{\to}\alpha{\to}\alpha){\to}\mathsf{Nat}^2{\to}\alpha$$

satisfying the reduction properties

$$\begin{array}{rcl} \mathsf{R}\alpha ab\mathsf{c}_0 & \twoheadrightarrow_\beta & a; \\ \mathsf{R}\alpha ab\mathsf{c}_{n+1} & \twoheadrightarrow_\beta & b\mathsf{c}_n(\mathsf{R}\alpha ab\mathsf{c}_n). \end{array}$$

3. Construct a term F such that $\vdash F : \mathsf{Nat}^2{\to}\mathsf{Nat}^2$ and

$$F\mathsf{c}_n \twoheadrightarrow_\beta \mathsf{c}_{n \dot{-} 1}.$$

Here $0\dot{-}1 = 0$ and $(n+1)\dot{-}1 = n$.

$\boxed{\lambda^{\mathtt{Ind}}2}$

2.15. Exercise. Construct F, G with $\vdash F : \mathsf{Nat}^I{\to}\mathsf{Nat}^2$ and $\vdash G : \mathsf{Nat}^2{\to}\mathsf{Nat}^I$ such that for all $n{\in}\mathbb{N}$ one has

$$\begin{array}{rcl} F\underline{n} & =_{\beta\iota} & \mathsf{c}_n, \\ G\mathsf{c}_n & =_{\beta\iota} & \underline{n}. \end{array}$$

2.16. Exercise. Define

$$\mathsf{Bool}^2 = \forall\alpha.\alpha{\to}\alpha{\to}\alpha.$$

The two canonical elements of this type are

$$\begin{array}{rcl} \mathsf{true}^2 & = & \lambda\alpha{:}*\,\lambda x, y{:}\alpha.x, \\ \mathsf{false}^2 & = & \lambda\alpha{:}*\,\lambda x, y{:}\alpha.y. \end{array}$$

Find F, G such that $\vdash F : \mathsf{Bool}^I{\to}\mathsf{Bool}^2$ and $\vdash G : \mathsf{Bool}^2{\to}\mathsf{Bool}^I$ preserving the truth values.

$\boxed{\lambda P}$

We use the notation $\forall x{:}A.B$ to denote just $\Pi x{:}A.B$.

2.17. Exercise. Prove $A{:}*, P{:}A{\to}* \vdash \forall x{:}A.Px{\to}Px$.

2.18. Exercise. Prove that every antisymmetric binary relation is irreflexive.

$$A{:}*, R{:}A{\to}A{\to}*, \mathtt{antisym}{:}\forall x, y{:}A.Rxy{\to}\neg Ryx \vdash \forall x{:}A.\neg Rxx.$$

2.19. EXERCISE. Define $\Gamma_{\texttt{ab.group}} = G{:}*, \Gamma_1, \Gamma_2, \Gamma_3$, where

$$\begin{array}{rcl}
\Gamma_1 & = & \texttt{eq} : G \to G \to *, \\
 & & \texttt{refl} : \forall x{:}G . \texttt{eq}\, x\, x, \\
 & & \texttt{sym} : \forall x, y{:}G . \texttt{eq}\, x\, y \to \texttt{eq}\, y\, x, \\
 & & \texttt{trans} : \forall x, y, z{:}G . \texttt{eq}\, x\, y \to \texttt{eq}\, y\, z \to \texttt{eq}\, x\, z. \\
\Gamma_2 & = & \texttt{plus} : G \to G \to G, \\
 & & 0 : G, \\
 & & - : G \to G, \\
 & & \texttt{comm} : \forall x, y{:}G . \texttt{eq}\, (\texttt{plus}\, x\, y)(\texttt{plus}\, y\, x), \\
 & & \texttt{assoc} : \forall x, y, z{:}G . \texttt{eq}\, (\texttt{plus}\, (\texttt{plus}\, x\, y)\, z)(\texttt{plus}\, x\, (\texttt{plus}\, y\, z)), \\
 & & \texttt{inv} : \forall x{:}G . \texttt{eq}\, (\texttt{plus}\, x (-x)) 0, \\
 & & \texttt{neutral} : \forall x{:}G . \texttt{eq}\, (\texttt{plus}\, x\ 0)\ x. \\
\Gamma_3 & = & \texttt{congr} : \forall x, x'{:}G . \texttt{eq}\ x\, x' \to \forall y{:}G . \texttt{eq}\, (\texttt{plus}\, x\, y)(\texttt{plus}\, x'\, y).
\end{array}$$

Show

$$\Gamma_{\texttt{ab.group}} \vdash \forall x{:}G . \texttt{eq}\ (-(-x))\, x.$$

$\boxed{\lambda\underline{\omega}}$

2.20. EXERCISE. Define $C_n = \lambda f{:} * \to * \ \lambda x{:} * . f^n x$. Construct a term F with $\vdash F : \mathsf{Nat}^* \to \mathsf{Nat}^*$ where $\mathsf{Nat}^* = (* \to *) \to (* \to *)$, such that

$$F C_n \twoheadrightarrow_\beta C_{n^3+3n^2+1}.$$

2.21. EXERCISE. Is there a term P such that one of the following hold? [Hint. Use proposition 1.6 and a case analysis.]

(i) $\alpha{:}*, f{:} * \to *, \texttt{ax}_1{:}f\alpha \to \alpha, \texttt{ax}_2{:}\alpha \to f\alpha \vdash P : \alpha \to f(f\alpha)$.

(ii) $\alpha{:}*, f{:} * \to *, \texttt{ax}_1{:}f\alpha \to \alpha, \texttt{ax}_2{:}\alpha \to f\alpha \vdash P : f(f\alpha) \to \alpha$.

$\boxed{\lambda\omega}$

2.22. EXERCISE. Is there a term P such that

$$\alpha{:}*, f{:} * \to *, \texttt{ax}{:}\forall \alpha . f\alpha \to \alpha \vdash P : f(f\alpha) \to \alpha?$$

2.23. EXERCISE (Giannini-Ronchi). Define

$$\begin{array}{rcl}
\mathsf{I} & \equiv & \lambda\alpha{:} * \lambda x{:}\alpha . x; \\
\mathsf{K} & \equiv & \lambda\alpha{:} * \lambda x, y{:}\alpha . x; \\
\Upsilon_\gamma & \equiv & \lambda\alpha{:} * . \alpha \to \gamma\alpha, \ \text{in context } \gamma{:} * \to *; \\
\Phi & \equiv & \lambda\gamma{:} * \to * . \forall\alpha{:} * . \Upsilon_\gamma \alpha; \\
\Theta & \equiv & \forall\gamma{:} * \to * . \Upsilon_\gamma(\Phi\gamma); \\
\omega & \equiv & \lambda\gamma{:} * \to * \ \lambda z{:}\Phi\gamma . z(\Phi\gamma)z; \\
M & \equiv & (\lambda x{:}\Theta . [x(\lambda\alpha{:} * . \alpha)\mathsf{I}, x(\lambda\alpha{:} * . \alpha \to \alpha)\mathsf{K}])\omega.
\end{array}$$

Here $[\ldots, \ldots]$ is defined in exercise 2.12. Find the type of M

$$\vdash M : ?.$$

$\boxed{\lambda^{\mathtt{Ind}}\omega}$

2.24. EXERCISE. Construct F with $\vdash F : \mathsf{Nat}^I \to \mathsf{Nat}^*$, such that for all $n \in \mathbb{N}$ one has

$$F\underline{n} \quad =_{\beta\iota} \quad C_n.$$

Here C_n is defined as in exercise 2.20. [Assume for this exercise that there is a recursor $\mathsf{R}_\Box \equiv \mathsf{R}_\Box^k$ such that for $\vdash k : \Box$ one has

$$\begin{aligned}
&\vdash \mathsf{R}_\Box : k \to (\mathsf{Nat} \to k \to k) \to \mathsf{Nat} \to k;\\
&\mathsf{R}_\Box ab0 \to_\iota a;\\
&\mathsf{R}_\Box ab(\mathtt{suc}\ x) \to_\iota bx(\mathsf{R}_\Box abx).]
\end{aligned}$$

$\boxed{\lambda P2}$

2.25. EXERCISE. Define in context $A{:}*, x, y{:}A$

$$\mathtt{eq}_L xy = \forall P{:}A \to * . Px \to Py.$$

This is Leibniz's equality (on type A). Usually we will write $x =_L y$ for $\mathtt{eq}_L xy$. Show that $=_L$ is an equivalence relation.

2.26. EXERCISE. 1. Express in a group (i.e. in context $\Gamma_{\mathtt{ab.group}}$, see exercise 2.19) the property to have torsion:

$$Tx \;\Leftrightarrow\; \exists n : \mathbb{N}. nx = 0.$$

[Hint. One can express a property $Q_x y$ with intended meaning "y belongs to the smallest set containing x and closed under addition". Then $Tx \Leftrightarrow Q_x 0$.]

2. Suppose that T with $\Gamma_{ab.group} \vdash T : G \to *$ expresses that an element has torsion. Prove:

$$\Gamma_{ab.group} \vdash \forall x{:}G.\mathtt{plus}\ x\ x =_L 0 \to Tx.$$

Here $=_L$ is Leibniz's equality on G.

$\boxed{\lambda^{\mathtt{Ind}} P2}$

2.27. EXERCISE. Define in context $\alpha, \beta{:}*$

$$\alpha \times \beta = (\mathtt{Ind}\ X)(\mathtt{pair} : \alpha \to \beta \to X).$$

1. Define terms $\mathtt{fst}, \mathtt{snd}$ such that $\mathtt{fst} : \alpha \times \beta \to \alpha$, $\mathtt{snd} : \alpha \times \beta \to \beta$ and

$$\mathtt{fst}(\mathtt{pair}\ x\ y) \twoheadrightarrow x, \ \mathtt{snd}(\mathtt{pair}\ x\ y) \twoheadrightarrow y.$$

2. Prove $\alpha, \beta{:}* \vdash \forall z{:}\alpha \times \beta. z =_L \mathtt{pair}(\mathtt{fst}\ z)(\mathtt{snd}\ z)$.

$\boxed{\lambda^{\mathrm{Ind}}C}$

2.28. EXERCISE. 1. Define by primitive recursion on Nat^I

- $A : \mathsf{Nat}^I \to *$ by

$$\begin{array}{rcl} A\,\underline{0} & = & \alpha; \\ A\,(x+1) & = & A\,x \to \alpha. \end{array}$$

- $f : \mathsf{Nat}^I \to \mathsf{Nat}^I$ such that $f\underline{n} = \underline{2n+1}$.
- $B : \mathsf{Nat}^I \to *$ by $B = \lambda x{:}\mathsf{Nat}^I.A\ (f\ x)$.

[The same recursor $\mathsf{R}^k_\square$ needs to be assumed as in exercise 2.24.]

2. Prove $\alpha{:}* \vdash (\forall x{:}\mathsf{Nat}^I.B\ x)$.

2.29. EXERCISE. Give an example of terms A, P, f with $\vdash A : *$, $\vdash P : A \to *$ and $\vdash f : * \to *$ such that

$$\begin{array}{l} \vdash \forall x{:}A.f(Px), \\ \vdash \neg f(\forall x{:}A.Px), \end{array}$$

where we use the definition of $\neg$ as in 2.10. [Hint. Take $A = \mathsf{Bool}^I$ and $P = \lambda x{:}A.\,\mathsf{Bool}^I$.]

References

Barendregt, H.P. [1992]. Lambda calculi with types, *in:* S. Abramsky, D.M. Gabbay and T.S.E. Maibaum (eds.), *Handbook of Logic in Computer Science*, Vol. II, Oxford University Press, pp. 117–309.

Barendregt, H.P. [1996]. The quest for correctness, *Images of SMC Research*, Stichting Mathematisch Centrum, pp. 39–58. Available at location http://www.cs.kun.nl/~henk/papers.html.

Barendregt, H.P. [1997]. The impact of the lambda calculus, *Bulletin of Symbolic Logic* **3**(2), pp. 181–215.

de Bruijn, N.G. [1970]. The mathematical language AUTOMATH, its usage and some of its extensions, *in:* D. Lacombe M. Laudet and M. Schuetzenberger (eds.), *Symposium on Automated Demonstration*, Lecture Notes in Mathematics 125, Springer, Berlin, pp. 29–61.

Church, A. [1940]. A formulation of the simple theory of types, *The Journal of Symbolic Logic* **5**, pp. 56–68.

Coquand, T. and G. Huet [1988]. The calculus of constructions, *Information and Computation* **76**, pp. 95–120. Available at location http://pauillac.inria.fr/~coq/coq-eng.html.

Fujita, K. and H. Tonino [1992]. On the adequacy of representing higher order intuitionistic logic as a pure type system, *Annals of Pure and Applied Logic* **57**, pp. 251–276.

Geuvers, J.H. [1993]. *Logics and Type Systems*, Dissertation, Catholic University of Nijmegen.

Girard, J.-Y. [1972]. Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur, These D'Etat, Universite Paris VII.

Martin-Löf, P. [1984]. *Intuitionistic Type Theory*, Studies in Proof Theory, Bibliopolis, Napoli.

Nederpelt, R.P., J.H. Geuvers and R.C. de Vrijer (eds.) [1994]. *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics **133**, North-Holland, Amsterdam.

Russell, B.A.W. and A.N. Whitehead [1910–13]. *Principia Mathematica Vol 1 and 2*, Cambridge University Press.

Scott, D.S. [1970]. Constructive validity, *in:* D. Lacombe M. Laudet and M. Schuetzenberger (eds.), *Symposium on Automated Demonstration*, Lecture Notes in Mathematics 125, Springer, Berlin, pp. 237–275.