Introduction to Type Theory

August 2007

Types Summer School

Bertinoro, It

Herman Geuvers

Nijmegen NL

Lecture 2: Dependent Type Theory, Logical Framework

For $\lambda\!\rightarrow$:

Direct representation (shallow embedding) of logic in type theory.

- Connectives each have a counterpart in the type theory:
  implication $\sim$ arrow type
- Logical rules have their direct counterpart in type theory
  λ-abstraction $\sim$ implication introduction
  application $\sim$ implication elimination
- Context declares assumptions

Second way of interpreting logic in type theory De Bruijn:

Logical framework encoding or deep embedding of logic in type theory.

- Type theory used as a meta system for encoding ones own logic.
- Choose an appropriate context Γ_L, in which the logic L (including its proof rules) is declared.
- Context used as a signature for the logic.
- Use the type system as the 'meta' calculus for dealing with substitution and binding.

	proof	formula
shallow embedding	$\lambda x{:}A.x$	$A{\rightarrow}A$
deep embedding	imp_intr $A\,A\,\lambda x{:}T\,A.x$	$T(A \Rightarrow A)$

Needed:

$$\begin{array}{rcl} \Rightarrow & : & \mathsf{prop}{\rightarrow}\mathsf{prop}{\rightarrow}\mathsf{prop} \\ \mathsf{T} & : & \mathsf{prop}{\rightarrow}\mathbf{type} \\ \mathsf{imp\_intr} & : & (A, B : \mathsf{prop})(\mathsf{T}\,A \rightarrow \mathsf{T}\,B) \rightarrow \mathsf{T}(A \Rightarrow B) \\ \mathsf{imp\_el} & : & (A, B : \mathsf{prop})\mathsf{T}(A \Rightarrow B) \rightarrow \mathsf{T}\,A \rightarrow \mathsf{T}\,B. \end{array}$$

Close to a Gödel like encoding of predicate logic in $\mathbb{N}$:
Define a coding fuction $\lceil - \rceil$ for formulas and define $\mathsf{Bew}(\lceil\varphi\rceil, n)$
Define mp : $\mathbb{N}{\rightarrow}\mathbb{N}{\rightarrow}\mathbb{N}$ such that

$$\forall x, y.\mathsf{Bew}(\lceil\varphi\rceil, x){\rightarrow}\mathsf{Bew}(\lceil\varphi \Rightarrow \psi\rceil, y){\rightarrow}\mathsf{Bew}(\lceil\psi, \mathsf{mp}(x, y)\rceil)$$

Direct representation	Deep encoding
One type system : One logic	One type system : Many logics
Logical rules $\sim$ type theoretic rules	Logical rules $\sim$ context declarations

Plan:

- First show examples of logics in a logical framework
- Then define precisely the type theory of the logical framework

Use **type** to denote the universe of types.

The encoding of logics in a logical framework is shown by three examples:

1. Minimal proposition logic
2. Minimal predicate logic (just $\{\Rightarrow, \forall\}$)
3. Untyped λ-calculus

Minimal propositional logic

Fix the signature (context) of minimal propositional logic.

$$\begin{aligned} \mathsf{prop} &: \mathbf{type} \\ \mathsf{imp} &: \mathsf{prop}{\rightarrow}\mathsf{prop}{\rightarrow}\mathsf{prop} \end{aligned}$$

Notation: $A \Rightarrow B$ for imp $A\,B$

The type prop is the type of 'names' of propositions.

NB: A term of type prop can't be inhabited (proved), as it's not a type.

We 'lift' a name p : prop to the type of its proofs by introducing the following map:

$$\mathsf{T} \quad : \quad \mathsf{prop}{\rightarrow}\mathbf{type}.$$

Intended meaning of Tp is 'the type of proofs of p'.

We interpret 'p is valid' by 'Tp is inhabited'.

To derive $\mathsf{T}p$ we also encode the logical derivation rules

$$\begin{aligned} \text{imp\_intr} &: \Pi p, q : \text{prop}.(\mathsf{T}p \to \mathsf{T}q) \to \mathsf{T}(p \Rightarrow q), \\ \text{imp\_el} &: \Pi p, q : \text{prop}.\mathsf{T}(p \Rightarrow q) \to \mathsf{T}p \to \mathsf{T}q. \end{aligned}$$

New phenomenon: Π-type:

$$\Pi x{:}A.B(x) \simeq \text{ the type of functions } f \text{ such that } f\,a : B(a) \text{ for all } a{:}A$$

imp_intr takes two (names of) propositions p and q and a term $f : \mathsf{T}\,p \to \mathsf{T}\,q$ and returns a term of type $\mathsf{T}(p \Rightarrow q)$

Indeed $A \Rightarrow A$, becomes valid:

$$\text{imp\_intr} A\, A (\lambda x{:}\mathsf{T}\, A.x) : \mathsf{T}(A \Rightarrow A)$$

Exercise: Construct a term of type $\mathsf{T}(A \Rightarrow (B \Rightarrow A))$

Define

Σ_{PROP} to be the signature for minimal proposition logic, PROP.

Desired properties of the encoding:

- Adequacy (soundness) of the encoding:

  $\vdash_{\mathsf{PROP}} A \quad \Rightarrow \quad \Sigma_{\mathsf{PROP}}, a_1{:}\mathsf{prop}, \ldots, a_n{:}\mathsf{prop} \vdash p : \mathsf{T}\, A$ for some p.

  $\{a, \ldots, a_n\}$ is the set of proposition variables in A.
  Proof by induction on the derivation of $\vdash_{\mathsf{PROP}} A$.

- Faithfulness (or completeness) is the converse. It also holds, but more involved to prove.

Minimal predicate logic over one domain A (just $\Rightarrow$ and $\forall$

Signature:

$$
\begin{array}{rcl}
\mathsf{prop} & : & \mathbf{type}, \\
\mathsf{A} & : & \mathbf{type}, \\
\mathsf{T} & : & \mathsf{prop}{\rightarrow}\mathbf{type} \\
\mathsf{f} & : & \mathsf{A}{\rightarrow}\mathsf{A}, \\
\mathsf{R} & : & \mathsf{A}{\rightarrow}\mathsf{A}{\rightarrow}\mathsf{prop}, \\
\Rightarrow & : & \mathsf{prop}{\rightarrow}\mathsf{prop}{\rightarrow}\mathsf{prop}, \\
\mathsf{imp\_intr} & : & \Pi p, q : \mathsf{prop}.(\mathsf{T}p{\rightarrow}\mathsf{T}q){\rightarrow}\mathsf{T}(p \Rightarrow q), \\
\mathsf{imp\_el} & : & \Pi p, q : \mathsf{prop}.\mathsf{T}(p \Rightarrow q){\rightarrow}\mathsf{T}p{\rightarrow}\mathsf{T}q.
\end{array}
$$

Now encode $\forall$: $\forall$ takes a $P : \mathsf{A}{\rightarrow}\mathsf{prop}$ and returns a proposition, so:

$$\forall : (\mathsf{A}{\rightarrow}\mathsf{prop}){\rightarrow}\mathsf{prop}$$

Minimal predicate logic over one domain A (just $\Rightarrow$ and $\forall$

Signature: Σ_{PRED}

$$\begin{array}{rcl} \mathsf{prop} & : & \mathbf{type}, \\ \mathsf{A} & : & \mathbf{type}, \\ & \vdots & \\ \Rightarrow & : & \mathsf{prop}{\rightarrow}\mathsf{prop}{\rightarrow}\mathsf{prop}, \\ \mathsf{imp\_intr} & : & \Pi p, q : \mathsf{prop}.(\mathsf{T}p{\rightarrow}\mathsf{T}q){\rightarrow}\mathsf{T}(p \Rightarrow q), \\ \mathsf{imp\_el} & : & \Pi p, q : \mathsf{prop}.\mathsf{T}(p \Rightarrow q){\rightarrow}\mathsf{T}p{\rightarrow}\mathsf{T}q. \end{array}$$

Now encode $\forall$: $\forall$ takes a P : A→prop and returns a proposition, so:

$$\forall : (\mathsf{A}{\rightarrow}\mathsf{prop}){\rightarrow}\mathsf{prop}$$

Universal quantification is translated as follows.

$$\forall x{:}A.(Px) \mapsto \forall(\lambda x{:}A.(Px))$$

Intro and elim rules for $\forall$:

$$\begin{aligned} \forall &: (\mathsf{A}{\rightarrow}\mathsf{prop}){\rightarrow}\mathsf{prop}, \\ \forall\_\mathsf{intr} &: \Pi P{:}\mathsf{A}{\rightarrow}\mathsf{prop}.(\Pi x{:}\mathsf{A}.\mathsf{T}(Px)){\rightarrow}\mathsf{T}(\forall P), \\ \forall\_\mathsf{elim} &: \Pi P{:}\mathsf{A}{\rightarrow}\mathsf{prop}.\mathsf{T}(\forall P){\rightarrow}\Pi x{:}\mathsf{A}.\mathsf{T}(Px). \end{aligned}$$

The proof of

$$\forall z{:}A(\forall x, y{:}A.Rxy) \Rightarrow Rzz$$

is now mirrored by the proof-term

$$\begin{aligned} \forall\_\mathsf{intr}[\_](\quad & \lambda z{:}\mathsf{A}.\mathsf{imp\_intr}[\_][\_](\lambda h{:}\mathsf{T}(\forall x, y{:}A.Rxy). \\ & \quad \forall\_\mathsf{elim}[\_](\forall\_\mathsf{elim}[\_]hz)z)\ ) \end{aligned}$$

We have replaced the instantiations of the Π-type by $[\_]$.
This term is of type

$$\mathsf{T}(\forall(\lambda z{:}\mathsf{A}.\mathsf{imp}(\forall(\lambda x{:}\mathsf{A}.(\forall(\lambda y{:}\mathsf{A}.\mathsf{R}xy))))(\mathsf{R}zz)))$$

Intro and elim rules for $\forall$:

$$\begin{aligned} \forall \quad &: \quad (\mathsf{A}\longrightarrow\mathsf{prop})\longrightarrow\mathsf{prop}, \\ \forall\_\mathsf{intr} \quad &: \quad \Pi P{:}\mathsf{A}\rightarrow\mathsf{prop}.(\Pi x{:}\mathsf{A}.\mathsf{T}(Px))\rightarrow\mathsf{T}(\forall P), \\ \forall\_\mathsf{elim} \quad &: \quad \Pi P{:}\mathsf{A}\rightarrow\mathsf{prop}.\mathsf{T}(\forall P)\rightarrow\Pi x{:}\mathsf{A}.\mathsf{T}(Px). \end{aligned}$$

The proof of

$$\forall z{:}A(\forall x, y{:}A.Rxy) \Rightarrow Rzz$$

is now mirrored by the proof-term

$$\begin{aligned} \forall\_\mathsf{intr}[\_](\quad &\lambda z{:}\mathsf{A}.\mathsf{imp\_intr}[\_][\_](\lambda h{:}\mathsf{T}(\forall x, y{:}A.Rxy). \\ &\qquad \forall\_\mathsf{elim}[\_](\forall\_\mathsf{elim}[\_]hz)z)\ ) \end{aligned}$$

Exercise: Construct a proof-term that mirrors the (obvious) proof of

$$\forall x(P\,x \Rightarrow Q\,x) \Rightarrow \forall x.P\,x \Rightarrow \forall x.Q\,x$$

Again one can prove adequacy

$$\vdash_{\text{PRED}} \varphi \quad \Rightarrow \quad \Sigma_{\text{PRED}}, x_1{:}\mathsf{A}, \ldots, x_n{:}\mathsf{A} \vdash p : \mathsf{T}\varphi, \text{ for some } p,$$

where $\{x_1, \ldots, x_n\}$ is the set of free variables in φ.

Faithfulness can be proved as well.

Logical Framework, LF, or λP

Derive judgements of the form

$$\Gamma \vdash M : B$$

- Γ is a context
- M and B are terms
  taken from the set of pseudoterms

$$\mathsf{T} ::= \mathsf{Var} \,|\, \mathbf{type} \,|\, \mathbf{kind} \,|\, \mathsf{TT} \,|\, \lambda x{:}\mathsf{T}.\mathsf{T} \,|\, \Pi x{:}\mathsf{T}.\mathsf{T},$$

Auxiliary judgement

$$\Gamma \vdash$$

denoting that Γ is a correct context.

Derivation rules of LF. (s ranges over $\{\mathbf{type}, \mathbf{kind}\}$.)

(base) $\emptyset \vdash$ (ctxt) $\dfrac{\Gamma \vdash A : \mathbf{s}}{\Gamma, x{:}A \vdash}$ if x not in Γ (ax) $\dfrac{\Gamma \vdash}{\Gamma \vdash \mathbf{type} : \mathbf{kind}}$

(proj) $\dfrac{\Gamma \vdash}{\Gamma \vdash x : A}$ if $x{:}A \in \Gamma$ (Π) $\dfrac{\Gamma, x{:}A \vdash B : \mathbf{s} \quad \Gamma \vdash A : \mathbf{type}}{\Gamma \vdash \Pi x{:}A.B : \mathbf{s}}$

(λ) $\dfrac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : \mathbf{s}}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$ (app) $\dfrac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$

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

Notation: write $A \to B$ for $\Pi x{:}A.B$ if $x \notin \mathsf{FV}(B)$.

- The contexts Σ_{PROP} and Σ_{PRED} are well-formed.
- The Π rule allows to form two forms of function types.

$$(\Pi)\ \frac{\Gamma, x{:}A \vdash B : \mathbf{s} \quad \Gamma \vdash A : \mathbf{type}}{\Gamma \vdash \Pi x{:}A.B : \mathbf{s}}$$

  - With $\mathbf{s} = \mathbf{type}$, we can form D→D and $\Pi x{:}\mathsf{D}.x = x$, etc.
  - With $\mathbf{s} = \mathbf{kind}$, we can form D→D→$\mathbf{type}$ and prop→$\mathbf{type}$.

Untyped λ-calculus. Signature Σ_{lambda}:

$$\begin{array}{rcl} \mathsf{D} & : & \mathbf{type}; \\ \mathsf{app} & : & \mathsf{D}{\to}(\mathsf{D}{\to}\mathsf{D}); \\ \mathsf{abs} & : & (\mathsf{D}{\to}\mathsf{D}){\to}\mathsf{D}. \end{array}$$

Encoding of λ-terms as terms of type D.

- A variable x in λ-calculus becomes $x : \mathsf{D}$ in the type system.
- The translation $[-] : \Lambda \to \mathsf{Term}(\mathsf{D})$ is defined as follows.

$$\begin{array}{rcl} [x] & = & x; \\ [PQ] & = & \mathsf{app}\ [P]\ [Q]; \\ [\lambda x.P] & = & \mathsf{abs}\ (\lambda x{:}\mathsf{D}.[P]). \end{array}$$

Examples: $[\lambda x.xx] := \mathsf{abs}(\lambda x{:}\mathsf{D}.\mathsf{app}\, x\, x)$
$[(\lambda x.xx)(\lambda y.y)] := \mathsf{app}(\mathsf{abs}(\lambda x{:}\mathsf{D}.\mathsf{app}\, x\, x))(\mathsf{abs}(\lambda y{:}\mathsf{D}.y))$.

Introducing β-equality in Σ_{lambda} :

$$\mathsf{eq}{:}\mathsf{D}{\rightarrow}\mathsf{D}{\rightarrow}\mathbf{type}.$$

Notation $P = Q$ for eq $P\ Q$.

Rules for proving equalities.

$$
\begin{array}{rcl}
\mathsf{refl} & : & \Pi x{:}\mathsf{D}.x = x, \\
\mathsf{sym} & : & \Pi x, y{:}\mathsf{D}.x = y \rightarrow y = x, \\
\mathsf{trans} & : & \Pi x, y, z{:}\mathsf{D}.x = y \rightarrow y = z \rightarrow x = z, \\
\mathsf{mon} & : & \Pi x, x', z, z'{:}\mathsf{D}.x = x' \rightarrow z = z' \rightarrow (\mathsf{app}\ z\ x) = (\mathsf{app}\ z'\ x'), \\
\mathsf{xi} & : & \Pi f, g{:}\mathsf{D}{\rightarrow}\mathsf{D}.(\Pi x{:}\mathsf{D}.(fx) = (gx)) \rightarrow (\mathsf{abs}\ f) = (\mathsf{abs}\ g), \\
\mathsf{beta} & : & \Pi f{:}\mathsf{D}{\rightarrow}\mathsf{D}.\Pi x{:}\mathsf{D}.(\mathsf{app}(\mathsf{abs}\ f)x) = (fx).
\end{array}
$$

Adequacy:

$$P =_\beta Q \Rightarrow \Sigma_{\mathsf{lambda}}, x_1{:}\mathsf{D}, \ldots, x_n{:}\mathsf{D} \vdash p : [P] = [Q], \text{ for some } p.$$

Here, $x_1, \ldots, x_n$ are the free variables in PQ

Faithfulness also holds.

Signature Σ_{lambda}:

D	:	**type**	sym	:	$\Pi x, y{:}\text{D}.x = y \rightarrow y = x,$
app	:	$\text{D}\rightarrow(\text{D}\rightarrow\text{D})$	trans	:	$\Pi x, y, z{:}\text{D}.x = y \rightarrow y = z \rightarrow x = z,$
abs	:	$(\text{D}\rightarrow\text{D})\rightarrow\text{D},$	mon	:	$\Pi x, x', z, z'{:}\text{D}.x = x' \rightarrow z = z' \rightarrow (\text{app } z\ x) = (\text{ap}$
eq	:	$\text{D}\rightarrow\text{D}\rightarrow\textbf{type},$	xi	:	$\Pi f, g{:}\text{D}\rightarrow\text{D}.(\Pi x{:}\text{D}.(fx) = (gx))\rightarrow(\text{abs } f) = (\text{a}$
refl	:	$\Pi x{:}\text{D}.x = x,$	beta	:	$\Pi f{:}\text{D}\rightarrow\text{D}.\Pi x{:}\text{D}.(\text{app}(\text{abs } f)x) = (fx).$

Exercise:

- Prove (i.e. find a proof term of the associated type) $(\lambda x.x)y =_\beta y$
- Add an axiom for η-equality ($\lambda x.Px =_\eta P$ if $x \notin \text{FV}(P)$) to the context and the extensionality rule ($\forall N(MN = PN \rightarrow M = N)$)
- Prove that η follows from extensionality.

Properties of λP.

- Uniqueness of types
  If $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$, then $\sigma =_{\beta\eta} \tau$.
- Subject Reduction
  If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_{\beta\eta} N$, then $\Gamma \vdash N : \sigma$.
- Strong Normalization
  If $\Gamma \vdash M : \sigma$, then all $\beta\eta$-reductions from M terminate.

Proof of SN is by defining a reduction preserving map from λP to $\lambda\!\rightarrow$.

Decidability Questions:

$\Gamma \vdash M : \sigma?$	TCP
$\Gamma \vdash M : ?$	TSP
$\Gamma \vdash ? : \sigma$	TIP

For λP:

- TIP is undecidable
- TCP/TSP: simultaneously with Context checking

Type Checking

Define algorithms $\mathrm{Ok}(-)$ and $\mathrm{Type}_{-}(-)$ simultaneously:

- $\mathrm{Ok}(-)$ takes a context and returns 'true' or 'false'
- $\mathrm{Type}_{-}(-)$ takes a context and a term and returns a term or 'false'.

The type synthesis algorithm $\mathrm{Type}_{-}(-)$ is sound if

$$\mathrm{Type}_{\Gamma}(M) = A \quad \Rightarrow \quad \Gamma \vdash M : A$$

for all Γ and M.

The type synthesis algorithm $\mathrm{Type}_{-}(-)$ is complete if

$$\Gamma \vdash M : A \quad \Rightarrow \quad \mathrm{Type}_{\Gamma}(M) =_{\beta\eta} A$$

for all Γ, M and A.

$$\mathrm{Ok}(<>) \quad = \quad \text{`true'}$$

$$\mathrm{Ok}(\Gamma, x{:}A) \quad = \quad \mathrm{Type}_\Gamma(A) \in \{\mathbf{type}, \mathbf{kind}\},$$

$$\mathrm{Type}_\Gamma(x) \quad = \quad \text{if } \mathrm{Ok}(\Gamma) \text{ and } x{:}A \in \Gamma \text{ then } A \text{ else `false'},$$

$$\mathrm{Type}_\Gamma(\mathbf{type}) \quad = \quad \text{if } \mathrm{Ok}(\Gamma)\text{then } \mathbf{kind} \text{ else `false'},$$

$$\begin{aligned}
\mathrm{Type}_\Gamma(MN) \quad = \quad & \text{if } \mathrm{Type}_\Gamma(M) = C \text{ and } \mathrm{Type}_\Gamma(N) = D \\
& \text{then} \quad \text{if } C \twoheadrightarrow_\beta \Pi x{:}A.B \text{ and } A =_\beta D \\
& \qquad\quad\ \text{then } B[N/x] \text{ else `false'} \\
& \text{else} \quad \text{`false'},
\end{aligned}$$

$$\begin{array}{lcl} \mathrm{Type}_\Gamma(\lambda x{:}A.M) & = & \textsf{if } \mathrm{Type}_{\Gamma,x:A}(M) = B \\ & & \quad \textsf{then} \qquad\qquad \textsf{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathbf{type}, \mathbf{kind}\} \\ & & \qquad\qquad\qquad\quad \textsf{then } \Pi x{:}A.B \textsf{ else } \text{`false'} \\ & & \quad \textsf{else } \text{`false'}, \\ \mathrm{Type}_\Gamma(\Pi x{:}A.B) & = & \textsf{if } \mathrm{Type}_\Gamma(A) = \mathbf{type} \textsf{ and } \mathrm{Type}_{\Gamma,x:A}(B) = s \\ & & \textsf{then } s \textsf{ else } \text{`false'} \end{array}$$

Soundness

$$\mathrm{Type}_\Gamma(M) = A \;\Rightarrow\; \Gamma \vdash M : A$$

Completeness

$$\Gamma \vdash M : A \;\Rightarrow\; \mathrm{Type}_\Gamma(M) =_{\beta\eta} A$$

As a consequence:

$$\mathrm{Type}_\Gamma(M) = \text{‘false’} \;\Rightarrow\; M \text{ is not typable in } \Gamma$$

NB 1. Completeness implies that Type terminates on all well-typed terms. We want that Type terminates on all pseudo terms.

NB 2. Completeness only makes sense if we have uniqueness of types (Otherwise: let $\mathrm{Type}_{\_}(-)$ generate a set of possible types)

Termination: we want $\mathrm{Type}_{\_}(-)$ to terminate on all inputs.

Interesting cases: λ-abstraction and application:

$$\mathrm{Type}_\Gamma(\lambda x{:}A.M) \quad = \quad \text{if } \mathrm{Type}_{\Gamma,x:A}(M) = B$$

$$\text{then} \qquad \text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathbf{type}, \mathbf{kind}\}$$

$$\text{then } \Pi x{:}A.B \text{ else 'false'}$$

$$\text{else 'false'},$$

! Recursive call is not on a smaller term!

Replace the side condition

$$\text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathbf{type}, \mathbf{kind}\}$$

by

$$\text{if } \mathrm{Type}_\Gamma(A) \in \{\mathbf{type}\}$$

and prove equivalent.

Termination: we want $\mathrm{Type}_{\_}(-)$ to terminate on all inputs.

Interesting cases: λ-abstraction and application:

$$\begin{array}{rcl} \mathrm{Type}_\Gamma(MN) & = & \text{if } \mathrm{Type}_\Gamma(M) = C \text{ and } \mathrm{Type}_\Gamma(N) = D \\ & & \quad \text{then} \quad \text{if } C \twoheadrightarrow_\beta \Pi x{:}A.B \text{ and } A =_\beta D \\ & & \quad \qquad\quad \text{then } B[N/x] \text{ else 'false'} \\ & & \quad \text{else} \quad \text{'false'}, \end{array}$$

! Need to decide β-reduction and β-equality!

For this case, termination follows from soundness of Type and the decidability of equality on well-typed terms (using SN and CR).

Direct representation (shallow embedding) of PRED into λP

Represent both the domains of the logic and the formulas as types.

$$\begin{array}{lcl} \mathsf{A} & : & \mathbf{type}, \\ P & : & \mathsf{A} \to \mathbf{type}, \\ R & : & \mathsf{A} \to \mathsf{A} \to \mathbf{type}, \end{array}$$

Now $\Rightarrow$ is represented as $\to$ and $\forall$ is represented as Π:

$$\forall x{:}A.P\,x \mapsto \Pi x{:}A.P\,x$$

Intro and elim rules are just λ-abstraction and application

Example

$$\mathsf{A} : \mathbf{type}, R : \mathsf{A} \rightarrow \mathsf{A} \rightarrow \mathbf{type} \quad \vdash \quad \lambda z{:}A.\lambda h{:}(\Pi x, y{:}A.R\,x\,y).h\,z\,z$$
$$: \quad \Pi z{:}A.(\Pi x, y{:}A.R\,x\,y) \rightarrow R\,z\,z$$

Exercise: Find terms of the following types (NB $\rightarrow$ binds strongest)

$$(\Pi x{:}A.P\,x \rightarrow Q\,x) \rightarrow (\Pi x{:}A.P\,x) \rightarrow \Pi x{:}A.Q\,x$$

and

$$(\Pi x{:}A.P\,x \rightarrow \Pi z.R\,z\,z) \rightarrow (\Pi x{:}A.P\,x) \rightarrow \Pi z{:}A.R\,z\,z).$$

Also write down the contexts in which these terms are typed.