Types Summer School
Giens France September 2002

# Type Systems

Herman Geuvers
Nijmegen University, NL

## Lecture 2: Higher Order Logic and Type Theory

The original motivation of Church to introduce simple type theory was:

to define higher order (predicate) logic

In $\lambda\!\!\rightarrow$ we add the following

- prop as a basic type
- $\Rightarrow$ : prop$\rightarrow$prop$\rightarrow$prop
- $\forall_\sigma$ : $(\sigma\rightarrow$prop$)\rightarrow$prop (for each type $\sigma$)

This defines the language of higher order logic $\mathbf{HOL}$.

- Induction

$$\forall_{N\to\mathsf{prop}}(\ \lambda P{:}N\to\mathsf{prop}.(P\,0)$$
$$\Rightarrow (\forall_N(\lambda x{:}N.(Px \Rightarrow P(S\,x)))$$
$$\Rightarrow \forall_N(\lambda x{:}N.Px)))$$

  Notation:

$$\forall P{:}N\to\mathsf{prop}(\ (P\,0)$$
$$\Rightarrow (\forall x{:}N.(Px \Rightarrow P(S\,x)))$$
$$\Rightarrow \forall x{:}N.Px)$$

- Higher order predicates/functions

  transitive closure of a relation $R$

$$\lambda R{:}\ A\to A\to\mathsf{prop}.\lambda x,y{:}A.$$
$$(\forall Q{:}A\to A\to\mathsf{prop}.(\mathsf{trans}(Q) \Rightarrow (R\subseteq Q) \Rightarrow Q\,x\,y))$$

  of type

$$(A\to A\to\mathsf{prop})\to(A\to A\to\mathsf{prop})$$

Derivation rules for Higher Order Logic $\mathbf{HOL}$ (following Church)

- Natural deduction style.
- Rules are 'on top' of the simple type theory.
- Judgements are of the form

$$\Delta \vdash_\Gamma \varphi$$

  - $\Delta = \psi_1, \ldots, \psi_n$
  - $\Gamma$ is a $\lambda{\to}$-context
  - $\Gamma \vdash \varphi$ : prop, $\Gamma \vdash \psi_1$ : prop,. . . , $\Gamma \vdash \psi_n$ : prop
  - $\Gamma$ is usually left implicit: $\Delta \vdash \varphi$

$$\text{(axiom)} \quad \frac{}{\Delta \vdash \varphi} \qquad \text{if } \varphi \in \Delta$$

$$\text{($\Rightarrow$ -introduction)} \quad \frac{\Delta \cup \varphi \vdash \psi}{\Delta \vdash \varphi \Rightarrow \psi}$$

$$\text{($\Rightarrow$ -elimination)} \quad \frac{\Delta \vdash \varphi \Rightarrow \psi \quad \Delta \vdash \varphi}{\Delta \vdash \psi}$$

$$\text{($\forall$-introduction)} \quad \frac{\Delta \vdash \varphi}{\Delta \vdash \forall x{:}\sigma.\varphi} \qquad \text{if } x{:}\sigma \notin \mathsf{FV}(\Delta)$$

$$\text{($\forall$-elimination)} \quad \frac{\Delta \vdash \forall x{:}\sigma.\varphi}{\Delta \vdash \varphi[t/x]} \qquad \text{if } t : \sigma$$

$$\text{(conversion)} \quad \frac{\Delta \vdash \varphi}{\Delta \vdash \psi} \qquad \text{if } \varphi =_\beta \psi$$

Church has additional things that we will not consider now:

- Negation connective with rules
- Classical logic

$$\frac{\Delta \vdash \neg\neg\varphi}{\Delta \vdash \varphi}$$

- Define other connectives in terms of $\Rightarrow, \forall, \neg$ (classically).
- Choice operator $\iota_\sigma : (\sigma \to \mathsf{prop}) \to \sigma$
- Rule for $\iota$:

$$\frac{\Delta \vdash \exists! x{:}\sigma.P\,x}{\Delta \vdash P(\iota_\sigma P)}$$

This (Church' original higher order logic) is basically the logic of the theorem prover HOL (Gordon, Melham, Harrison) and of Isabelle-HOL (Paulson, Nipkow).
We will here restrict to the basic constructive core $(\forall, \Rightarrow)$ of **HOL**.

Inportant in $\mathbf{HOL}$:

Conversion rule:

$$\dfrac{\dfrac{\Delta \vdash \forall P{:}N{\to}\mathsf{prop}.(\ldots Pc \ldots)}{\Delta \vdash (\ldots(\lambda y{:}N.y > 0)c \ldots)}\ \forall\text{-elim}}{\Delta \vdash (\ldots c > 0 \ldots)}\ \text{conv}$$

Definability of other connectives (constructively):

$$\begin{aligned}
\bot &:= \forall\alpha{:}\mathsf{prop}.\alpha \\
\varphi{\wedge}\psi &:= \forall\alpha{:}\mathsf{prop}.(\varphi \Rightarrow \psi \Rightarrow \alpha) \Rightarrow \alpha \\
\varphi{\vee}\psi &:= \forall\alpha{:}\mathsf{prop}.(\varphi \Rightarrow \alpha) \Rightarrow (\psi \Rightarrow \alpha) \Rightarrow \alpha \\
\exists x{:}\sigma.\varphi &:= \forall\alpha{:}\mathsf{prop}.(\forall x{:}\sigma.\varphi \Rightarrow \alpha) \Rightarrow \alpha
\end{aligned}$$

Equality is definable in higher order logic:

$t$ and $q$ terms are equal if they share the same properties (Leibniz equality)

Definition in $\mathbf{HOL}$ (for $t, q : A$):

$$t{=_A}q := \forall P{:}A{\to}\mathsf{prop}.(Pt \Rightarrow Pq)$$

- This equality is reflexive and transitive (easy)

Equality is definable in higher order logic:

$t$ and $q$ terms are equal if they share the same properties (Leibniz equality)

Definition in **HOL** (for $t, q : A$):

$$t{=_A}q := \forall P{:}A{\to}\mathsf{prop}.(Pt \Rightarrow Pq)$$

- This equality is reflexive and transitive (easy)
- It is also symmetric(!) Trick: take $\lambda y{:}A.y =_A t$ for $P$.

$$\dfrac{\dfrac{\dfrac{\Delta \vdash t =_A q}{\Delta \vdash \forall P{:}A{\to}\mathsf{prop}.(Pt \Rightarrow Pq)}}{\Delta \vdash (t =_A t) \Rightarrow (q =_A t)} \qquad \dfrac{\ldots}{\Delta \vdash t =_A t}}{\Delta \vdash q =_A t}$$

$$\text{(axiom)} \quad \frac{}{\Delta \vdash \varphi} \qquad \text{if } \varphi \in \Delta$$

$$\text{($\Rightarrow$ -introduction)} \quad \frac{\Delta \cup \varphi \vdash \psi}{\Delta \vdash \varphi \Rightarrow \psi}$$

$$\text{($\Rightarrow$ -elimination)} \quad \frac{\Delta \vdash \varphi \Rightarrow \psi \quad \Delta \vdash \varphi}{\Delta \vdash \psi}$$

$$\text{($\forall$-introduction)} \quad \frac{\Delta \vdash \varphi}{\Delta \vdash \forall x{:}\sigma.\varphi} \qquad \text{if } x{:}\sigma \notin \mathsf{FV}(\Delta)$$

$$\text{($\forall$-elimination)} \quad \frac{\Delta \vdash \forall x{:}\sigma.\varphi}{\Delta \vdash \varphi[t/x]} \qquad \text{if } t : \sigma$$

$$\text{(conversion)} \quad \frac{\Delta \vdash \varphi}{\Delta \vdash \psi} \qquad \text{if } \varphi =_\beta \psi$$

Why not introduce a $\lambda$-term notation for the derivations?

This gives a type theory $\lambda$HOL

- Let prop be a new 'universe' of propositional types.
- Direct encoding (deep embedding) of $\mathbf{HOL}$ into the type theory $\lambda$HOL

(axiom) $$\frac{}{\Delta \vdash_\Gamma x : \varphi}$$ if $x{:}\varphi \in \Delta$

($\Rightarrow$ -introduction) $$\frac{\Delta, x{:}\varphi \vdash_\Gamma M : \psi}{\Delta \vdash_\Gamma \lambda x{:}\varphi.M : \varphi \Rightarrow \psi}$$

($\Rightarrow$ -elimination) $$\frac{\Delta \vdash_\Gamma M : \varphi \Rightarrow \psi \quad \Delta \vdash_\Gamma N : \varphi}{\Delta \vdash_\Gamma M\ N : \psi}$$

($\forall$-introduction) $$\frac{\Delta \vdash_{\Gamma, x:\sigma} M : \varphi}{\Delta \vdash_\Gamma \lambda x{:}\sigma.M : \forall x{:}\sigma.\varphi}$$ if $x{:}\sigma \notin \mathsf{FV}(\Delta)$

($\forall$-elimination) $$\frac{\Delta \vdash_\Gamma M : \forall x{:}\sigma.\varphi}{\Delta \vdash_\Gamma M\ t : \varphi[t/x]}$$ if $\Gamma \vdash t : \sigma$

(conversion) $$\frac{\Delta \vdash_\Gamma M : \varphi}{\Delta \vdash_\Gamma M : \psi}$$ if $\varphi =_\beta \psi$

Now we have two 'levels' of type theories

- The (simple) type theory describing the language of $\mathbf{HOL}$
- The type theory for the proof-terms of $\mathbf{HOL}$

NB Many rules, many similar rules.

We put these levels together into one type theory $\lambda$HOL.
Pseudoterms:

$$\mathsf{T} ::= \mathsf{Prop} \mid \mathsf{Type} \mid \mathsf{Type}' \mid \mathsf{Var} \mid (\Pi \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid (\lambda \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid \mathsf{TT}$$

$\{\mathsf{Prop}, \mathsf{Type}, \mathsf{Type}'\}$ is the set of sorts, $\mathcal{S}$.

Some of the typing rules are parametrized

(axiom) $\vdash \mathsf{Prop} : \mathsf{Type}$ $\qquad\qquad$ $\vdash \mathsf{Type} : \mathsf{Type}'$

(var) $$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$$ $\qquad$ (weak) $$\frac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C}$$

($\Pi$) $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2}$$ if $(s_1, s_2) \in \{ (\mathsf{Type}, \mathsf{Type}), (\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Prop}) \}$

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

(app) $$\frac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B[N/x]}$$

(conv) $$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$ if $A =_\beta B$

$$(\Pi)\ \frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2} \text{ if } (s_1, s_2) \in \{\ (\text{Type}, \text{Type}),\ (\text{Prop}, \text{Prop}), (\text{Type}, \text{Prop})\ \}$$

- The combination (Type,Type) forms the function types $A \to B$ for $A, B$:Type.
  This comprises the unary predicate types and binary relations types: $A \to$Prop and $A \to A \to$Prop.
  Also: higher order predicate types like $(A \to A \to \text{Prop}) \to$Prop.
  NB A $\Pi$-type formed by (Type,Type) is always an $\to$-type.
- (Prop,Prop) forms the propositional types $\varphi \to \psi$ for $\varphi, \psi$:Prop; implicational formulas.
  NB A $\Pi$-type formed by (Type,Type) is always an $\to$-type.
- (Type,Prop) forms the dependent propositional type $\Pi x{:}A.\varphi$ for $A$:Type, $\varphi$:Prop; universally quantified formulas.

Example: Deriving irreflexivity from anti-symmetry

$$\begin{aligned}
\mathsf{Rel} &:= \lambda X{:}\mathsf{Type}.X \to X \to \mathsf{Prop} \\
\mathsf{AntiSym} &:= \lambda X{:}\mathsf{Type}.\lambda R{:}(\mathsf{Rel}\, X).\forall x, y{:}X.(Rxy) \Rightarrow (Ryx) \Rightarrow \bot \\
\mathsf{Irrefl} &:= \lambda X{:}\mathsf{Type}.\lambda R{:}(\mathsf{Rel}\, X).\forall x{:}X.(Rxx) \Rightarrow \bot
\end{aligned}$$

Derivation in **HOL**:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x^A y^A R\, x\, y \Rightarrow R\, y\, x \Rightarrow \bot}{\forall y^A R\, x\, y \Rightarrow R\, y\, x \Rightarrow \bot}}{R\, x\, x \Rightarrow R\, x\, x \Rightarrow \bot} \quad [R\, x\, x]}{R\, x\, x \Rightarrow \bot} \quad [R\, x\, x]}{\bot}}{R\, x\, x \Rightarrow \bot}}{\forall x^A . R\, x\, x \Rightarrow \bot}$$

Derivation in **HOL**, with terms:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z : \forall x^A y^A R\,x\,y \Rightarrow R\,y\,x \Rightarrow \bot}{zx : \forall y^A R\,x\,y \Rightarrow R\,y\,x \Rightarrow \bot}}{zxx : R\,x\,x \Rightarrow R\,x\,x \Rightarrow \bot} \quad [q : R\,x\,x]}{zxxq : R\,x\,x \Rightarrow \bot} \quad [q : R\,x\,x]}{zxxqq : \bot}}{\lambda q{:}(R\,x\,x).zxxqq : R\,x\,x \Rightarrow \bot}}{\lambda x{:}A.\lambda q{:}(R\,x\,x).zxxqq : \forall x^A.R\,x\,x \Rightarrow \bot}$$

Typing judgement in λHOL:

$$A{:}\mathsf{Type}, R{:}A{\to}A{\to}\mathsf{Prop},\ z : \Pi x, y{:}A.(R\,x\,y{\to}R\,y\,x{\to}\bot) \vdash \lambda x{:}A\lambda q{:}(R\,x\,x).z\,x\,x\,q\,q : (\Pi x{:}A.R\,x\,x{\to}\bot)$$

Question: is the type theory $\lambda$HOL really isomorphic with $\mathbf{HOL}$?

Yes:Disambiguation Lemma Given

$$\Gamma \vdash M : T \text{ in } \lambda\text{HOL}$$

there is a permutation of $\Gamma$: $\Gamma_D, \Gamma_L, \Gamma_P$ such that

1. $\Gamma_D, \Gamma_L, \Gamma_P \vdash M : T$
2. $\Gamma_D$ consists only of declarations $A : \mathsf{Type}$
3. $\Gamma_L$ consists only of declarations $x : \sigma$ with $\Gamma_D \vdash \sigma : \mathsf{Type}$
4. $\Gamma_P$ consists only of declarations $z : \varphi$ with $\Gamma_D, \Gamma_L \vdash \varphi : \mathsf{Prop}$

So, if $\Gamma \vdash M : T$, we also have

$$\underbrace{A_1{:}\mathsf{Type}, \ldots, A_n{:}\mathsf{Type}}_{\Gamma_D}, \underbrace{x{:}\sigma_1, \ldots, x_m{:}\sigma_m}_{\Gamma_L}, \underbrace{z_1{:}\varphi_1, \ldots z_p{:}\varphi_p}_{\Gamma_P} \vdash M : T$$

$\Gamma_D$ domainvar. $\Gamma_L$ termvar. $\Gamma_P$ proofvar.

Properties of $\lambda$HOL.

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

Proof of SN is a higher order extension of the one for $\lambda 2$ (using the saturated sets).

Decidability Questions:

$$\begin{array}{ll} \Gamma \vdash M : \sigma ? & \text{TCP} \\ \Gamma \vdash M : ? & \text{TSP} \\ \Gamma \vdash ? : \sigma & \text{TIP} \end{array}$$

For $\lambda$HOL:

- TIP is undecidable
- TCP/TSP: simultaneously.

## 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'.

$$\mathrm{Ok}(<>) = \text{‘true’}$$

$$\mathrm{Ok}(\Gamma, x{:}A) = \mathrm{Type}_\Gamma(A) \in \{\mathsf{Prop}, \mathsf{Type}\},$$

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

$$\mathrm{Type}_\Gamma(\mathsf{Prop}) = \text{if } \mathrm{Ok}(\Gamma) \text{then Type else ‘false’},$$

$$\mathrm{Type}_\Gamma(\mathsf{Type}) = \text{if } \mathrm{Ok}(\Gamma) \text{then Type}' \text{ else ‘false’},$$

$$\mathrm{Type}_\Gamma(\mathsf{Type}') = \text{‘false’},$$

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

$$
\begin{aligned}
\mathrm{Type}_\Gamma(\lambda x{:}A.M) = \;& \text{if } \mathrm{Type}_{\Gamma,x:A}(M) = B \\
& \text{then} \quad\quad\quad \text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathsf{Prop}, \mathsf{Type}\} \\
& \qquad\quad\quad\quad \text{then } \Pi x{:}A.B \text{ else 'false'} \\
& \text{else 'false'},
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Type}_\Gamma(\Pi x{:}A.B) = \;& \text{if } \mathrm{Type}_\Gamma(A) = \mathsf{Type} \\
& \qquad\quad \text{and } \mathrm{Type}_{\Gamma,x:A}(B) = s \in \{\mathsf{Prop}/\mathsf{Type}\} \\
& \text{then } s \text{ else} \\
& \text{if } \mathrm{Type}_\Gamma(A) = \mathsf{Prop} \text{ and } \mathrm{Type}_{\Gamma,x:A}(B) = \mathsf{Prop} \\
& \text{then Prop else 'false'}
\end{aligned}
$$

Soundness

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

for all $\Gamma$, $M$.

Completeness

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

for all $\Gamma$, $M$ and $A$.

This implies that, if $\mathrm{Type}_\Gamma(M) =$ 'false', then $M$ is not typable in $\Gamma$.

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.
(Not guaranteed by soundness and completness)

Interesting case (1): application:

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

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

## Termination

Interesting case(2): $\lambda$-abstraction:

$$\begin{aligned}\mathrm{Type}_\Gamma(\lambda x{:}A.M) = \ & \text{if } \mathrm{Type}_{\Gamma,x:A}(M) = B \\ & \text{then} \quad \text{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathsf{Prop}, \mathsf{Type}\} \\ & \qquad\qquad \text{then } \Pi x{:}A.B \text{ else 'false'} \\ & \text{else 'false'},\end{aligned}$$

Replace the side condition

$$\mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\mathsf{Prop}, \mathsf{Type}\}$$

by

$$\mathrm{Type}_\Gamma(A) = \mathsf{Prop} \quad \text{and} \quad B \equiv \Pi\vec{y}{:}\vec{C}.D \text{ with } D \neq \mathsf{Prop}/\mathsf{Type}/\mathsf{Type}'$$

or

$$\mathrm{Type}_\Gamma(A) = \mathsf{Type} \quad \text{and} \quad B \equiv \Pi\vec{y}{:}\vec{C}.D \text{ with } D \neq \mathsf{Type}/\mathsf{Type}'.$$