Proofs as Programs Summer School
Eugene Oregon June - July 2002

Type Systems

Herman Geuvers
Nijmegen University, NL

Lecture 4: Higher Order Logic

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

to define higher order (predicate) logic

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

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

This defines the language of higher order logic.

- Induction

$$\forall_{N\to\mathsf{prop}}(\ \lambda P{:}N\to\mathsf{prop}.(P0) \\ \Rightarrow (\forall_N(\lambda x{:}N.(Px \Rightarrow P(S\,x))) \\ \Rightarrow \forall_N(\lambda x{:}N.Px)))$$

Notation:

$$\forall P{:}N\to\mathsf{prop}(\ (P0) \\ \Rightarrow (\forall x{:}N.(Px \Rightarrow P(S\,x))) \\ \Rightarrow \forall x{:}N.Px)$$

- 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 (following Church)

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

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

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

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

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

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

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

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

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

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 ι:

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

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**.

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):

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

Idea:
The definition of a connective is an encoding of the elimination rule.

Existential quantifier

$$\exists x{:}\sigma.\varphi := \forall\alpha{:}\mathsf{prop}.(\forall x{:}\sigma.\varphi \Rightarrow \alpha) \Rightarrow \alpha$$

Derivations for the elimination and introduction rules.

$$\dfrac{\exists x{:}\sigma.\varphi \quad \begin{array}{c}[\varphi]\\ \vdots \\ C\end{array}}{C}\; x \notin \mathsf{FV}(C, \mathsf{ass.})$$

Existential quantifier

$$\exists x{:}\sigma.\varphi := \forall \alpha{:}\mathsf{prop}.(\forall x{:}\sigma.\varphi \Rightarrow \alpha) \Rightarrow \alpha$$

Derivations for the elimination and introduction rules.

$$\frac{\exists x{:}\sigma.\varphi \quad \begin{array}{c}[\varphi]\\ \vdots \\ C\end{array}}{C}\, x \notin \mathsf{FV}(C, \mathsf{ass.}) \qquad \frac{\dfrac{\exists x{:}\sigma.\varphi}{(\forall x{:}\sigma.\varphi \Rightarrow C) \Rightarrow C} \quad \dfrac{\begin{array}{c}[\varphi]\\ \vdots \\ C\end{array}}{\forall x{:}\sigma.\varphi \Rightarrow C}}{C}$$

Existential quantifier

$$\exists x{:}\sigma.\varphi := \forall \alpha{:}\mathsf{prop}.(\forall x{:}\sigma.\varphi \Rightarrow \alpha) \Rightarrow \alpha$$

Derivations for the elimination and introduction rules.

$$\frac{\exists x{:}\sigma.\varphi \quad \begin{array}{c}[\varphi]\\ \vdots\\ C\end{array}}{C}\; x \notin \mathsf{FV}(C, \mathsf{ass.}) \qquad \frac{\dfrac{\exists x{:}\sigma.\varphi}{(\forall x{:}\sigma.\varphi \Rightarrow C) \Rightarrow C} \quad \dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ C\end{array}}{\forall x{:}\sigma.\varphi \Rightarrow C}}{C}$$

$$\frac{\varphi[t/x]}{\exists x{:}\sigma.\varphi} \qquad \dfrac{\dfrac{\dfrac{\varphi[t/x] \quad \dfrac{[\forall x{:}\sigma.\varphi \Rightarrow \alpha]}{\varphi[t/x] \Rightarrow \alpha}}{\alpha}}{(\forall x{:}\sigma.\varphi \Rightarrow \alpha) \Rightarrow \alpha}}{\exists x{:}\sigma.\varphi}$$

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 $t, q : A$):

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

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

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

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

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

(axiom)	$\dfrac{}{\Delta \vdash \varphi}$	if $\varphi \in \Delta$
($\Rightarrow$ -introduction)	$\dfrac{\Delta \cup \varphi \vdash \psi}{\Delta \vdash \varphi \Rightarrow \psi}$	
($\Rightarrow$ -elimination)	$\dfrac{\Delta \vdash \varphi \Rightarrow \psi \quad \Delta \vdash \varphi}{\Delta \vdash \psi}$	
($\forall$-introduction)	$\dfrac{\Delta \vdash \varphi}{\Delta \vdash \forall x{:}\sigma.\varphi}$	if $x{:}\sigma \notin \mathsf{FV}(\Delta)$
($\forall$-elimination)	$\dfrac{\Delta \vdash \forall x{:}\sigma.\varphi}{\Delta \vdash \varphi[t/x]}$	if $t : \sigma$
(conversion)	$\dfrac{\Delta \vdash \varphi}{\Delta \vdash \psi}$	if $\varphi =_\beta \psi$

Why not introduce a λ-term notation for the derivations?

This gives a type theory λHOL

- No 'lifting' of prop to the $\mathbf{type}$ level
- Let prop be a new 'universe' of propositional types.
- Direct encoding (deep embedding) of $\mathbf{HOL}$ into the type theory λHOL

Why not introduce a λ-term notation for the derivations?

This gives a type theory λHOL

- No 'lifting' of prop to the **type** level
- Let prop be a new 'universe' of propositional types.
- Direct encoding (deep embedding) of **HOL** into the type theory λHOL

Example (with $\exists x{:}\sigma.\varphi := \forall \alpha{:}\mathsf{prop}.(\forall x{:}\sigma.\varphi \to \alpha) \to \alpha$):

$$\dfrac{\dfrac{M : \exists x{:}\sigma.\varphi}{M\,C : (\forall x{:}\sigma.\varphi \to C) \to C} \qquad \dfrac{\begin{array}{c}[z : \varphi] \\ \vdots \\ P : C\end{array}}{\lambda x{:}\sigma.\lambda z{:}\varphi.P : \forall x{:}\sigma.\varphi \Rightarrow C}}{M\,C\,(\lambda x{:}\sigma.\lambda z{:}\varphi.P) : C}$$

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

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

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

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

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

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

Now we have two ‘levels’ of type theories

- The (simple) type theory describing the language of **HOL**
- The type theory for the proof-terms of **HOL**

NB Many rules, many similar rules.

We put these levels together into one type theory λHOL.
Pseudoterms:

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

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

Some of the typing rules are parametrized (axiom) $\vdash \mathsf{Prop} : \mathsf{Type}$ $\qquad$ $\vdash \mathsf{Type} : \mathsf{Type}'$

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

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

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

(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 $A \rightarrow B$ for A, B:Type.
  This comprises the unary predicate types and binary relations types: $A \rightarrow$Prop and $A \rightarrow A \rightarrow$Prop.
  Also: higher order predicate types like $(A \rightarrow A \rightarrow \text{Prop}) \rightarrow$Prop.
  NB A Π-type formed by (Type,Type) is always an $\rightarrow$-type.
- (Prop,Prop) forms the propositional types $\varphi \rightarrow \psi$ for φ, ψ:Prop; implicational formulas.
  NB A Π-type formed by (Type,Type) is always an $\rightarrow$-type.
- (Type,Prop) forms the dependent propositional type $\Pi x{:}A.\varphi$ for A:Type, φ: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 $\mathbf{HOL}$:

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

Derivation in $\mathbf{HOL}$, with terms:

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

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 λHOL really isomorphic with $\mathbf{HOL}$?

Yes:Disambiguation Lemma Given

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

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

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

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

Yes: Disambiguation Lemma Given

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

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

1. $\Gamma_D, \Gamma_L, \Gamma_P \vdash M : A$
2. Γ_D consists only of declarations $A : \mathsf{Type}$
3. Γ_L consists only of declarations $x : \sigma$ with $\Gamma_D \vdash \sigma : \mathsf{Type}$
4. Γ_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

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

Properties of λHOL.

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

- TIP is undecidable
- TCP/TSP: simultaneously.
  The type checking algorithm is close to the one for λP. (In λP we had a judgement of correct context; this form of judgement could also be introduced for λHOL)

$$
\begin{aligned}
\mathrm{Type}_{\Gamma,y:B}(x) &= \ \textsf{if}\ \mathrm{Type}_\Gamma(B) \in \{\textsf{Prop}, \textsf{Type}, \textsf{Type}'\}\ \textsf{and}\ x{:}A \in \Gamma \\
&\quad\ \textsf{then}\ A\ \textsf{else}\ \text{‘}\textsf{false}\text{’},
\end{aligned}
$$

$$
\mathrm{Type}_{<>}(\textsf{Prop}) = \textsf{Type}
$$

$$
\mathrm{Type}_{<>}(\textsf{Type}) = \textsf{Type}'
$$

$$
\mathrm{Type}_{\Gamma,y:B}(\textsf{Prop}) = \ \textsf{if}\ \mathrm{Type}_\Gamma(B) \in \{\textsf{Prop}, \textsf{Type}, \textsf{Type}'\}\ \textsf{then Type}
$$

$$
\mathrm{Type}_{\Gamma,y:B}(\textsf{Type}) = \ \textsf{if}\ \mathrm{Type}_\Gamma(B) \in \{\textsf{Prop}, \textsf{Type}, \textsf{Type}'\}\ \textsf{then Type}'
$$

$$
\begin{aligned}
\mathrm{Type}_\Gamma(MN) \;=\;& \textsf{if } \mathrm{Type}_\Gamma(M) = C \textsf{ and } \mathrm{Type}_\Gamma(N) = D\\
& \textsf{then} \quad \textsf{if } C \twoheadrightarrow_\beta \Pi x{:}A.B \textsf{ and } A =_\beta D\\
& \qquad\quad\; \textsf{then } B[N/x] \textsf{ else } \text{‘false’}\\
& \textsf{else} \quad \text{‘false’},\\[1ex]
\mathrm{Type}_\Gamma(\lambda x{:}A.M) \;=\;& \textsf{if } \mathrm{Type}_{\Gamma,x:A}(M) = B\\
& \textsf{then} \qquad\quad \textsf{if } \mathrm{Type}_\Gamma(\Pi x{:}A.B) \in \{\textsf{Prop}, \textsf{Type}, \textsf{Type}'\}\\
& \qquad\qquad\quad\; \textsf{then } \Pi x{:}A.B \textsf{ else } \text{‘false’}\\
& \textsf{else } \text{‘false’},\\
\mathrm{Type}_\Gamma(\Pi x{:}A.B) \;=\;& \textsf{if } \mathrm{Type}_\Gamma(A) = s_1 \textsf{ and } \mathrm{Type}_{\Gamma,x:A}(B) = s_2\\
& \textsf{and } (s_1, s_2) \in \{(\textsf{Type}, \textsf{Type}), (\textsf{Prop}, \textsf{Prop}), (\textsf{Type}, \textsf{Prop})\}\\
& \textsf{then } s\\
& \textsf{else } \text{‘false’}
\end{aligned}
$$