Types Summer School
Gothenburg Sweden August 2005

Type Systems

Herman Geuvers
Radboud University Nijmegen, 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 $\lambda\!\to$ we add the following

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

This defines the language of higher order logic **HOL**.

- Induction

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

Notation:

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

- Higher order predicates/functions

transitive closure of a relation R

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

of type

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

Derivation rules for Higher Order Logic **HOL** (following Church)

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

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

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

(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 $x{:}\sigma \notin$ FV(Δ)

($\forall$-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 $\varphi =_\beta \psi$

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 $\iota_\sigma : (\sigma{\to}\text{prop}){\to}\sigma$
- 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**.

Important in **HOL**:
Conversion rule:

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

Definability of other connectives (constructively):

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

Equality is definable in higher order logic:

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

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

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

- This equality is reflexive and transitive (easy)
- It is also symmetric(!) Trick: find a "smart" predicate P

Exercise: Prove reflexivity, transitivity and symmetry of $=_A$.

Exercise: Proof of symmetry of $=_A$.
(Trick: take $\lambda y{:}A.y =_A t$ for P.)

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

One more exercise on Higher Order Logic
The transitive closure of a binary relation R on A has been defined as follows.

trclos $R := \lambda x, y{:}A.$
$(\forall Q{:}A{\to}A{\to}\text{Prop}.(\text{trans}(Q){\to}(R \subseteq Q){\to}(Q\,x\,y)))$.

1. Prove that the transitive closure is transitive.
2. Prove that the transitive closure of R contains R.

(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 $x{:}\sigma \notin \text{FV}(\Delta)$
($\forall$-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

- Let prop be a new 'universe' of propositional types.
- Direct encoding (deep embedding) of **HOL** into the type theory λHOL

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

Now we have two 'levels' of type theories

- The (simple) type theory describing the language of **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:

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

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

Some of the typing rules are parametrized

(axiom)	$\vdash \mathsf{Prop} : \mathsf{Type}$	$\vdash \mathsf{Type} : \mathsf{Type}'$
(var)	$\dfrac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$	(weak) $\dfrac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C}$
(Π)	$\dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2}$	if $(s_1, s_2) \in \{$ (Type, Type), (Prop, Prop), (Type, Prop) $\}$
(λ)	$\dfrac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : s}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$	
(app)	$\dfrac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash M N : B[N/x]}$	
(conv)	$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B}$	if $A =_{\beta} B$

(Π) $\dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2}$ if $(s_1, s_2) \in \{$ (Type, Type), (Prop, Prop), (Type, 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 $(A{\to}A{\to}\mathsf{Prop}){\to}\mathsf{Prop}$.
  NB A Π-type formed by (Type,Type) is always an $\to$-type.
- (Prop,Prop) forms the propositional types $\varphi{\to}\psi$ for φ, ψ:Prop; implicational formulas.
  NB A Π-type formed by (Type,Type) is always an $\to$-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

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

Derivation in $\mathbf{HOL}$:

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

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

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

Typing judgement in λHOL:

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

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

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

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

Properties of λHOL.

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

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

For λHOL:

- 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'.

$$
\begin{aligned}
\mathrm{Ok}(<>) &= \text{'true'} \\
\mathrm{Ok}(\Gamma, x{:}A) &= \mathrm{Type}_\Gamma(A) \in \{\mathsf{Prop}, \mathsf{Type}\}, \\
\mathrm{Type}_\Gamma(x) &= \text{if } \mathrm{Ok}(\Gamma) \text{ and } x{:}A \in \Gamma \text{ then } A \text{ else 'false'}, \\
\mathrm{Type}_\Gamma(\mathsf{Prop}) &= \text{if } \mathrm{Ok}(\Gamma) \text{then } \mathsf{Type} \text{ else 'false'}, \\
\mathrm{Type}_\Gamma(\mathsf{Type}) &= \text{if } \mathrm{Ok}(\Gamma) \text{then } \mathsf{Type}' \text{ else 'false'}, \\
\mathrm{Type}_\Gamma(\mathsf{Type}') &= \text{'false'},
\end{aligned}
$$

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

Soundness

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

for all Γ, M.

Completeness

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

for all Γ, M and A.
This implies that, if $\text{Type}_\Gamma(M) =$ 'false', then M is not typable in Γ.

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

Termination

We want $\text{Type}_-(-)$ to terminate on all inputs.
(Not guaranteed by soundness and completness)

Interesting case (1): application:

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

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

Termination

Interesting case(2): λ-abstraction:

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

Replace the side condition

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

by

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

or

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