Lambda-Calculus and Type Theory

ISR 2024
Obergurgl, Austria

Herman Geuvers & Niels van der Weide

Radboud University Nijmegen, The Netherlands

Lecture 7

Higher order logic in the Calculus of constructions and in Coq

The Barendregt cube

Barendregt cube: 8 typed λ-calculi, defined in one coherent way.
Generalization: Berardi & Terlouw: Pure Type Systems

framework for defining and studying typed λ-calculi
PTS = pure type system

the PTS rules are basically the λP rules as presented before.

variations on the product rule

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

λP $\quad s_1 = *,\ s_2 \in \{*, \Box\}$

$\quad (s_1, s_2) \in \{(*, *), (*, \Box)\}$

$\lambda{\to}$ $\quad (s_1, s_2) \in \{(*, *)\}$

$\lambda 2$ $\quad (s_1, s_2) \in \{(*, *), (\Box, *)\}$

λC $\quad (s_1, s_2) \in \{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}$

(axiom) $\vdash * : \square$

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

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

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

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

System	$\mathcal{R}$			
$\lambda\!\rightarrow$	$(*, *)$			
$\lambda 2$ (system F)	$(*, *)$	$(\Box, *)$		
λP (LF)	$(*, *)$		$(*, \Box)$	
$\lambda\overline{\omega}$	$(*, *)$			$(\Box, \Box)$
λP2	$(*, *)$	$(\Box, *)$	$(*, \Box)$	
$\lambda\omega$ (system Fω)	$(*, *)$	$(\Box, *)$		$(\Box, \Box)$
$\lambda P\overline{\omega}$	$(*, *)$		$(*, \Box)$	$(\Box, \Box)$
λPω (CC)	$(*, *)$	$(\Box, *)$	$(*, \Box)$	$(\Box, \Box)$

the Barendregt cube

$$
\begin{tikzcd}
 & \lambda\omega \arrow[rr] & & \lambda C \\
\lambda 2 \arrow[ur] \arrow[rr] & & \lambda P2 \arrow[ur] & \\
 & \lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \lambda P\underline{\omega} \arrow[uu] \\
\lambda{\to} \arrow[uu, "{(\square, *)}"] \arrow[ur, "{(\square, \square)}"] \arrow[rr, "{(*, \square)}"] & & \lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Calculus of Constructions

$\lambda\!\to$ in this presentation is equivalent to $\lambda\!\to$ as presented before.
Similarly for $\lambda 2$, λP, … This cube also gives a fine structure for the

Calculus of Constructions, CC (Coquand and Huet)

- Polymorphic data types on the $*$-level,
  e.g. $\Pi\alpha{:}*.\alpha\to(\alpha\to\alpha)\to\alpha : *$.
- Predicate domains on the $\square$-level,
  e.g. $N\to N\to * : \square$
- Logic on the $*$-level,
  e.g. $\varphi\wedge\psi := \Pi\alpha{:}*.(\varphi\to\psi\to\alpha)\to\alpha : *$.
- Universal quantification (first and higher order),
  e.g. $\Pi P{:}N\to *.\Pi x{:}N.Px\to Px : *$.

Examples

- Induction

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

- Defining the smallest subset of A containing $P : A \to *$ and closed under $f : A \to A$.

$$S := \lambda y : A.\\ \quad \forall Q : A\to *.(P \subseteq Q) \to (\forall x : A.Q\,x \to Q\,(f\,x)) \to Q\,y$$

where $P \subseteq Q := \forall x : A.P\,x \to Q\,x$.
To prove:

1. S is closed under f,
2. S contains P,
3. S is the smallest such.

Examples ctd.

- Higher order predicates/functions: transitive closure of a relation R

$$\lambda R : A \to A \to * \,.\, \lambda x, y : A. \\ \quad (\forall Q : A \to A \to * \,.\, (\mathrm{trans}(Q) \to (R \subseteq Q) \to Q\, x\, y))$$

of type

$$(A \to A \to *) \to (A \to A \to *)$$

Example trans clos higher order and inductively

- transitive closure in higher order logic:

$$\lambda R : A \to A \to * \,.\, \lambda x, y : A.$$
$$(\forall Q : A \to A \to * \,.\, (\mathrm{trans}(Q) \to (R \subseteq Q) \to Q\,x\,y))$$

of type

$$(A \to A \to *) \to (A \to A \to *)$$

- transitive closure inductively:

```
Inductive TrclosInd (R : A->A->Prop) : A -> A -> Prop :=
| sub : forall x y : A, R x y -> TrclosInd x y
| trans : forall x y z : A,
        TrclosInd x y -> TrclosInd y z -> TrclosInd x z.
```

Exercise trans clos higher order

Given the transitive closure of a binary relation, defined in higher order logic:

$$\begin{aligned}\text{trclos}\, R \quad &:= \quad \lambda x, y{:}A.\\ &\qquad (\forall Q{:}A{\to}A{\to}*.(\text{trans}(Q){\to}(R \subseteq Q){\to}(Q\, x\, y))).\end{aligned}$$

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

Higher order logic HOL

In higher order logic (originally due to Church[1940]) we have:

- higher order domains: D, $D \to \mathrm{Prop}$, $(D \to \mathrm{Prop}) \to \mathrm{Prop}$, etc (sets of predicates over predicates over ...).
- higher order function domains: $(D \to D) \to D$, $((D \to D) \to D) \to D$, etc.
- $\forall$-quantification over all domains

We can do Higher Order Logic in Coq
In Coq we often have the choice to define sets/predicates/relations inductively or via higher order logic. The Standard Library uses inductive representations.

Definability of other connectives (constructively)

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

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

Existential quantifier

$$\exists x{:}\sigma.\varphi := \forall \alpha{:}*.(\forall x{:}\sigma.\varphi \to \alpha) \to \alpha$$

Derivation of the elimination rule in HOL.

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

Equality

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 $t, q : A$):

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

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

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

CC versus HOL

Question: is the type theory CC really isomorphic with HOL?
No: only if we disambiguate $*$ into Set and Prop (or $*_s$ and $*_p$).
This is the type theory of Coq.

Properties of CC

- 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 $M \to_\beta N$, then $\Gamma \vdash N : A$.
- Strong Normalization
  If $\Gamma \vdash M : A$, then all β-reductions from M terminate.

Proof of SN is a really difficult.

Decidability Questions

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

For CC:

- 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 CC)