Proving with Computer Assistance
Lecture 10

Higher Order Logic and the Calculus of Constructions

Herman Geuvers
For the slides, thanks to: Freek Wiedijk
The Barendregt cube

Barendregt cube: 8 typed \( \lambda \)-calculi, defined in one coherent way.

Generalization: Berardi & Terlouw: Pure Type Systems

\[ \text{framework for defining and studying typed } \lambda \text{-calculi} \]

\[ \text{PTS} = \text{pure type system} \]

the PTS rules are basically the \( \lambda P \) rules as presented before.
variations on the product rule

\[
\begin{array}{c}
\Gamma \vdash A : s_1 \\
\Gamma, x : A \vdash B : s_2 \\
\hline \\
\Gamma \vdash \Pi x : A. B : s_2
\end{array}
\]

\begin{align*}
\lambda P & \quad s_1 = *, s_2 \in \{*, \Box\} \\
(s_1, s_2) & \in \{(*, *), (*, \Box)\} \\
\lambda \rightarrow & \quad (s_1, s_2) \in \{(*, *)\} \\
\lambda 2 & \quad (s_1, s_2) \in \{(*, *), (\Box, *)\} \\
\lambda C & \quad (s_1, s_2) \in \{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}
\end{align*}
(axiom) \[\vdash * : \Box\]

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

(λ) \[ \Gamma, x:A \vdash M : B \quad \Gamma \vdash \Pi x:A.B : s \]
\[ \Gamma \vdash \lambda x:A.M : \Pi x:A.B \]

(app) \[ \Gamma \vdash M : \Pi x:A.B \quad \Gamma \vdash N : A \]
\[ \Gamma \vdash MN : B[N/x] \]

(conv) \[ \Gamma \vdash M : A \quad \Gamma \vdash B : s \quad \text{if} \ A =_\beta B \]
\[ \Gamma \vdash M : B \]
\[(\Pi) \quad \begin{array}{c}
\Gamma \vdash A : s_1 \\
\Gamma, x : A \vdash B : s_2 \\
\Gamma \vdash \Pi x : A.B : s_2
\end{array} \quad \text{if } (s_1, s_2) \in \mathcal{R}\]

<table>
<thead>
<tr>
<th>System</th>
<th>$\mathcal{R}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \to$</td>
<td>$(\ast, \ast)$</td>
</tr>
<tr>
<td>$\lambda 2$ (system F)</td>
<td>$(\ast, \ast) (\Box, \ast)$</td>
</tr>
<tr>
<td>$\lambda P$ (LF)</td>
<td>$(\ast, \ast) (\ast, \Box)$</td>
</tr>
<tr>
<td>$\lambda \omega$</td>
<td>$(\ast, \ast) (\Box, \Box)$</td>
</tr>
<tr>
<td>$\lambda P2$</td>
<td>$(\ast, \ast) (\Box, \ast) (\ast, \Box)$</td>
</tr>
<tr>
<td>$\lambda \omega$ (system $F\omega$)</td>
<td>$(\ast, \ast) (\Box, \ast) (\Box, \Box)$</td>
</tr>
<tr>
<td>$\lambda P \omega$</td>
<td>$(\ast, \ast) (\ast, \Box) (\Box, \Box)$</td>
</tr>
<tr>
<td>$\lambda P \omega$ (CC)</td>
<td>$(\ast, \ast) (\Box, \ast) (\ast, \Box) (\Box, \Box)$</td>
</tr>
</tbody>
</table>
the Barendregt cube

\[ \lambda \omega \rightarrow \lambda C \]

\[ \lambda 2 \rightarrow \lambda P 2 \]

\[ (\Box, *) \rightarrow \lambda P \omega \]

\[ \lambda \rightarrow (\Box, \Box) \rightarrow \lambda P \]
Calculus of Constructions

\( \lambda \to \) in this presentation is equivalent to \( \lambda \to \) as presented before. Similarly for \( \lambda 2 \), \( \lambda P \), \ldots 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 \(\Box\)**-level,
  e.g.  \( N \to N \to * : \Box \)

- **Logic** on the \(*\)**-level,
  e.g.  \( \varphi \land \psi := \Pi \alpha : * . (\varphi \to \psi \to \alpha) \to \alpha : * \).

- **Universal quantification** (first and higher order),
  e.g.  \( \Pi P : N \to * . \Pi x : N . P x \to P x : * \).
Examples

• Induction

\[ \forall P : N \to \ast \ ((P \ 0) \to (\forall x : N. (P x \to P(S \ x)))) \to \forall x : N. P x \]

• Higher order predicates/functions: transitive closure of a relation \( R \)

\[ \lambda R : A \to A \to \ast \cdot \lambda x, y : A. \]
\[ (\forall Q : A \to A \to \ast \cdot (\text{trans}(Q) \to (R \subseteq Q) \to Q x y)) \]

of type

\[ (A \to A \to \ast) \to (A \to A \to \ast) \]
Example trans clos higher order and inductively

- transitive closure in higher order logic:

\[
\lambda R : A \to A \to \ast . \lambda x, y : A. \\
(\forall Q : A \to A \to \ast . (\text{trans}(Q) \to (R \subseteq Q) \to Q x y))
\]

of type

\[
(A \to A \to \ast) \to (A \to A \to \ast)
\]

- transitive closure inductively:

\[
\text{Inductive TrclosInd} (R : A \to A \to \text{Prop}) : A \to A \to \text{Prop} := \\
\text{sub} : \forall x y : A, R x y \to \text{TrclosInd} x y \\
\mid \text{trans} : \forall x y z : A, \\
\quad \text{TrclosInd} x y \to \text{TrclosInd} y z \to \text{TrclosInd} x z.
\]
Exercise trans clos higher order

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

\[
\text{trclos } R := \lambda x, y : A. \\
(\forall Q : A \rightarrow A \rightarrow \ast . (\text{trans}(Q) \rightarrow (R \subseteq Q) \rightarrow (Q x y))).
\]

1. Prove that the transitive closure is transitive.

2. Prove that the transitive closure of \( R \) contains \( R \).
In higher order logic (originally due to Church[1940]) we have:

- higher order domains: \( D, D \rightarrow Prop, (D \rightarrow Prop) \rightarrow Prop, \) etc (sets of predicates over predicates over . . . ).

- higher order function domains: \( (D \rightarrow D) \rightarrow D, ((D \rightarrow D) \rightarrow D) \rightarrow D, \) 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)

\[ \bot := \forall \alpha: \ast. \alpha \]
\[ \varphi \land \psi := \forall \alpha: \ast. (\varphi \rightarrow \psi \rightarrow \alpha) \rightarrow \alpha \]
\[ \varphi \lor \psi := \forall \alpha: \ast. (\varphi \rightarrow \alpha) \rightarrow (\psi \rightarrow \alpha) \rightarrow \alpha \]
\[ \exists x: \sigma. \varphi := \forall \alpha: \ast. (\forall x: \sigma. \varphi \rightarrow \alpha) \rightarrow \alpha \]

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

\[ \exists x: \sigma. \varphi := \forall \alpha: \ast. (\forall x: \sigma. \varphi \rightarrow \alpha) \rightarrow \alpha \]

Derivation of the elimination rule in HOL.

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

\[
\begin{array}{c}
[\varphi] \\
\vdots \\
\exists x: \sigma. \varphi \quad C \\
\hline
\forall x: \sigma. \varphi \rightarrow C \rightarrow C \\
\hline
C \\
\end{array}
\]
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 $t, q : A$):

$$t_A = q := \forall P : A \to \ast. (Pt \to Pq)$$

- 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 $\ast$ into Set and Prop (or $\ast_s$ and $\ast_p$). 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 $M \rightarrow_\beta N$, then $\Gamma \vdash N : A$.

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

Proof of SN is a really difficult.
Decidability Questions

Γ ⊢ M : σ? TCP
Γ ⊢ M : ? TSP
Γ ⊢ ? : σ TIP

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)