Proofs as Programs Summer School
Eugene Oregon June - July 2002

Type Systems

Herman Geuvers
Nijmegen University, NL

Lecture 5: Pure Type Systems

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

(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 $(s_1, s_2) \in \{\, (\mathsf{Type}, \mathsf{Type}), (\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Prop}) \,\}$

(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 $A =_\beta B$

λHOL contains $\lambda 2$ and $\lambda\!\rightarrow$.

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

This rule allows to form

- $\rightarrow$-types on the Type-level (one copie of $\lambda\!\rightarrow$)
- $\rightarrow$-types on the Prop-level (second copie of $\lambda\!\rightarrow$)
- $\Pi\alpha{:}\mathsf{Prop}.\alpha\!\rightarrow\!\alpha$: polymorphic types on the Prop-level (one copie of $\lambda 2$)

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

Why not extend λHOL to include

- Higher order logic over polymorphic domains?
  like $\Pi A : \mathsf{Type}.A \to A$
- Quantification over all domains?
  like in $\Pi A : \mathsf{Type}.\Pi P{:}A \to \mathsf{Prop}.\Pi x{:}A.P\,x \Rightarrow P\,x$

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

Why not extend λHOL to include

- Higher order logic over polymorphic domains?
  like $\Pi A : \mathsf{Type}.A \to A$
- Quantification over all domains?
  like in $\Pi A : \mathsf{Type}.\Pi P{:}A \to \mathsf{Prop}.\Pi x{:}A.P\,x \Rightarrow P\,x$

This can easily be done by allowing in the Π-rule

- $(s_1, s_2) \in \{\ (\mathsf{Type}', \mathsf{Type})\ \}$ to obtain higher order logic over polymorphic domains $\rightsquigarrow$ system λU^-
- $(s_1, s_2) \in \{\ (\mathsf{Type}', \mathsf{Prop})\ \}$ to allow quantification over all domains $\rightsquigarrow$ system λU

Problem:

- λU (λHOL + (Type',Type) and (Type',Prop)) is inconsistent (Girard)
- λU^- (λHOL + (Type',Type)) is inconsistent (Coquand, Hurkens)

NB λHOL + (Type',Prop) is consistent.

Problem:

- λU (λHOL + (Type',Type) and (Type',Prop)) is inconsistent (Girard)
- λU^- (λHOL + (Type',Type)) is inconsistent (Coquand, Hurkens)

NB λHOL + (Type',Prop) is consistent.

Implications

- λU^- can't be used as a logic.
- In λU^-, there is a closed term M with $\vdash M : \bot$
- This M can not be in normal form (syntactic reasoning)
- So, λU^- is not SN

Type Checking in λU^- is still decidable:
All types (terms of type Prop, Type or Type$'$) are strongly normalizing

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

In the type synthesis algorithm we only check equality of types

Variations on the rules of λHOL:

- There are many type system with (slightly) different rules
- Many (proofs of) properties are similar
- Plan: Study these type systems in one general framework:
  - The cube of typed λ-calculi (Barendregt)
  - Pure Type Systems (Terlouw, Berardi)

The cube of typed λ-calculi: (forget about Type$'$ for the moment)
Vary on all possible combinations for

$$\mathcal{R} \subseteq \{\ (\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Type}), (\mathsf{Prop}, \mathsf{Type})\ \}$$

in the Π-rule:

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

We take $(\mathsf{Prop}, \mathsf{Prop})$ in every $\mathcal{R}$

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

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

$\lambda\!\!\rightarrow$ in this presentation is equivalent to $\lambda\!\!\rightarrow$ in the way we've presented before. Similarly for $\lambda 2$, λP, ...

This cube also gives a fine structure for the

Calculus of Constructions, CC (Coquand and Huet)

CC has:

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

One can do higher order predicate logic in CC, in a slightly unusual way:

- 'propositions' and first order 'sets' are both of type Prop
- propositions and sets are completely mixed

Is it faithful to do higher order predicate logic in CC??

Answer: No!
There are non-provable formulas of $\mathbf{HOL}$ that become inhabited in CC

Consider extensionality of propostions:

$$\mathsf{EXT} := \forall \alpha, \beta{:}\mathsf{prop}.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{\mathsf{prop}} \beta)$$

In CC, this becomes $\Pi \alpha, \beta{:}\mathsf{Prop}.(\alpha \leftrightarrow \beta) \rightarrow (\alpha =_{\mathsf{Prop}} \beta)$

Consider extensionality of propostions:

$$\mathsf{EXT} := \forall \alpha, \beta{:}\mathsf{prop}.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{\mathsf{prop}} \beta)$$

In CC, this becomes $\Pi \alpha, \beta{:}\mathsf{Prop}.(\alpha \leftrightarrow \beta) \rightarrow (\alpha =_{\mathsf{Prop}} \beta)$

Suppose two base domains A and B and constants $a : A$, $b : B$.
In $\mathbf{HOL}$, the following formulas are consistent.

- $\varphi := \forall x{:}A.x = a$, $\psi := \forall x{:}B.\exists y{:}B.x \neq y$

Consider extensionality of propostions:

$$\mathsf{EXT} := \forall \alpha, \beta{:}\mathsf{prop}.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{\mathsf{prop}} \beta)$$

In CC, this becomes $\Pi \alpha, \beta{:}\mathsf{Prop}.(\alpha \leftrightarrow \beta) \rightarrow (\alpha =_{\mathsf{Prop}} \beta)$

Suppose two base domains A and B and constants $a : A$, $b : B$.
In $\mathbf{HOL}$, the following formulas are consistent.

- $\varphi := \forall x{:}A.x = a$, $\psi := \forall x{:}B.\exists y{:}B.x \neq y$

But in CC, EXT also applies to the base sets A and B.

$A \leftrightarrow B$ (both are non-empty) so $A =_{\mathsf{Prop}} B$

so property ψ (of A) also applies to B

so $\forall x{:}A.\exists y{:}A.x \neq y$

contradicting φ

So, in CC, φ and ψ are inconsistent

We have to be careful when doing higher order logic in CC.

Or: we may try to improve on this: taking the sets and the propositions apart:
System λPREDω:

- Sorts: Prop, Set, Type^p, Type^s
- Axioms for these sorts: Prop : Type^p, Set : Type^s
- Rules R:
  - (Prop, Prop): implication
  - (Set, Prop): first order quantification
  - (Type^p, Prop): higher order quantification
  - (Set, Set): function types
  - (Set, Type^p): predicate types
  - (Type^p, Type^p): higher order types

Pure Type Systems

Determined by a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ with

- $\mathcal{S}$ the set of sorts
- $\mathcal{A}$ the set of axioms, $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$
- $\mathcal{R}$ the set of rules, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$

If $s_2 = s_3$ in $(s_1, s_2, s_3) \in \mathcal{R}$, we write $(s_1, s_2) \in \mathcal{R}$.

pseudoterms:

$$\mathsf{T} ::= \mathcal{S} \mid \mathsf{Var} \mid (\Pi \mathsf{Var{:}T.T}) \mid (\lambda \mathsf{Var{:}T.T}) \mid \mathsf{TT}.$$

$$(\textsf{sort}) \quad \vdash s_1 : s_2 \quad \text{if } (s_1, s_2) \in \mathcal{A} \qquad (\textsf{var}) \quad \frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A} \quad \text{if } x \notin \Gamma$$

$$(\textsf{weak}) \quad \frac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C} \quad \text{if } x \notin \Gamma$$

$$(\Pi) \quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3} \quad \text{if } (s_1, s_2, s_3) \in \mathcal{R}$$

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

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

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

Examples of PTSs

λPREDω	
$\mathcal{S}$	$\mathsf{Set}, \mathsf{Type}^{\mathcal{S}}, \mathsf{Prop}, \mathsf{Type}$
$\mathcal{A}$	$\mathsf{Set} : \mathsf{Type}^{\mathcal{S}}, \mathsf{Prop} : \mathsf{Type}$
$\mathcal{R}$	$(\mathsf{Set}, \mathsf{Set}), (\mathsf{Set}, \mathsf{Type}), (\mathsf{Type}, \mathsf{Type}), (\mathsf{Prop}, \mathsf{Prop}),$ $(\mathsf{Set}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Prop})$

CC	
$\mathcal{S}$	$\mathsf{Prop}, \mathsf{Type}$
$\mathcal{A}$	$\mathsf{Prop} : \mathsf{Type}$
$\mathcal{R}$	$(\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Prop}, \mathsf{Type}), (\mathsf{Type}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Type})$

λHOL	
$\mathcal{S}$	$\mathsf{Prop}, \mathsf{Type}, \mathsf{Type}'$
$\mathcal{A}$	$\mathsf{Prop} : \mathsf{Type}, \mathsf{Type} : \mathsf{Type}'$
$\mathcal{R}$	$(\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Type}), (\mathsf{Type}, \mathsf{Prop})$

λU	
$\mathcal{S}$	$\mathsf{Prop}, \mathsf{Type}, \mathsf{Type}'$
$\mathcal{A}$	$\mathsf{Prop} : \mathsf{Type}, \mathsf{Type} : \mathsf{Type}'$
$\mathcal{R}$	$(\mathsf{Prop}, \mathsf{Prop}), (\mathsf{Type}, \mathsf{Type}), (\mathsf{Type}', \mathsf{Type}), (\mathsf{Type}', \mathsf{Prop}), (\mathsf{Type}, \mathsf{Prop})$

There are now two type systems for higher order predicate logic:
λPREDω and λHOL.
They are isomorphic.

What is the use of the abstract framework of PTSs?

- Present (the kernel of) systems in a uniform way
- Compare systems (e.g. λHOL, λPREDω, CC) within one framework
- Prove properties for all systems at once.

Properties of PTSs.

- Uniqueness of types
  If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $B =_\beta B$.
  Holds if $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ are functions.
- Subject Reduction
  If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_\beta N$, then $\Gamma \vdash N : \sigma$.
- Substitution property
  If $\Gamma, x : \tau, \Delta \vdash M : \sigma$, $\Gamma \vdash P : \tau$, then $\Gamma, \Delta \vdash M[P/x] : \sigma$.
- Thinning
  If $\Gamma \vdash M : \sigma$ and $\Gamma \subseteq \Delta$, then $\Delta \vdash M : \sigma$.
- Strengthening
  If $\Gamma, x : \tau \vdash M : \sigma$ and $x \notin \mathsf{FV}(M)$, then $\Gamma \vdash M : \sigma$.

Strong Normalization

If $\Gamma \vdash M : \sigma$, then all β-reductions from M terminate.

Holds for some PTSs, and for some not.

SN for CC is proved by a higher order extension of the saturated sets argument (for $\lambda 2$).