Types Summer School
Giens France September 2002

Type Systems

Herman Geuvers
Nijmegen University, NL

Lecture 3: Pure Type Systems

(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 $A =_\beta B$

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)

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:

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

(sort) $\vdash s_1 : s_2$ if $(s_1, s_2) \in \mathcal{A}$

(var) $$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$$ if $x \notin \Gamma$

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

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

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

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

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

(sort) $\quad \vdash s_1 : s_2$ if $(s_1, s_2) \in \mathcal{A}$

(var) $\quad \dfrac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$ if $x \notin \Gamma$

(weak) $\quad \dfrac{\Gamma \vdash A : s \qquad \Gamma \vdash \text{at} : C}{\Gamma, x{:}A \vdash \text{at} : C}$ if $x \notin \Gamma$, for at atomic (var. or sort)

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

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

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

(conv_β) $\quad \dfrac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}\ A =_\beta B$

Examples of PTSs (1)

λHOL contains $\lambda 2$ and $\lambda{\to}$.

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

This rule allows to form

- $\to$-types on the Type-level (one copie of $\lambda{\to}$)
- $\to$-types on the Prop-level (second copie of $\lambda{\to}$)
- polymorphic types on the Prop-level (one copie of $\lambda 2$)

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

$\lambda\rightarrow$	
$\mathcal{S}$	Type, Type$'$
$\mathcal{A}$	Type : Type$'$
$\mathcal{R}$	(Type, Type)

$\lambda 2$	
$\mathcal{S}$	Prop, Type
$\mathcal{A}$	Prop : Type
$\mathcal{R}$	(Prop, Prop), (Type, Prop)

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

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 $\Pi A : \mathsf{Type}.\Pi P{:}A{\to}\mathsf{Prop}.\Pi x{:}A.P\,x{\to}P\,x$

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

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

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

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

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

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 (by some 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

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

In the type synthesis algorithm we only check equality of types

Examples of PTSs (2)

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:

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

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	$\mathcal{R}$			
$\lambda\!\to$	(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)

$\lambda\!\to$ in this presentation is equivalent to $\lambda\!\to$ 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. $\Pi\alpha{:}\mathsf{Prop}.\alpha{\rightarrow}(\alpha{\rightarrow}\alpha){\rightarrow}\alpha$.
- Predicate domains on the Type-level,
  e.g. $N{\rightarrow}N{\rightarrow}\mathsf{Prop}$
- Logic on the Prop-level,
  e.g. $\varphi \wedge \psi := \Pi\alpha{:}\mathsf{Prop}.(\varphi{\rightarrow}\psi{\rightarrow}\alpha){\rightarrow}\alpha$.
- Universal quantification (first and higher order),
  e.g. $\Pi P{:}N{\rightarrow}\mathsf{Prop}.\Pi x{:}N.Px{\rightarrow}Px$.

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 **HOL** that become inhabited in CC

Consider extensionality of propostions:

in $\mathbf{HOL}$: 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:

in $\mathbf{HOL}$: 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 $\mathbf{HOL} +$ EXT, 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:

in $\mathbf{HOL}$: EXT $:= \forall\alpha,\beta{:}\text{prop}.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{\text{prop}} \beta)$

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

Suppose two base domains A and B and constants $a : A$, $b : B$.
In $\mathbf{HOL} +$ EXT, 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 $A =_{\text{Prop}} B$
so property ψ (of B) also applies to A
so $\forall x{:}A.\exists y{:}A.x \neq y$
contradicting φ
So, in CC+EXT, 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

Examples of PTSs

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

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

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

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

$\lambda\star$	
$\mathcal{S}$	$\star$
$\mathcal{A}$	$\star : \star$
$\mathcal{R}$	$(\star, \star)$

A PTS-morphism from $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ to $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ is an $f : \mathcal{S} \to \mathcal{S}'$ that preserves the axioms and rules:

- if $(s_1, s_2) \in \mathcal{A}$ then $(f(s_1), f(s_2)) \in \mathcal{A}'$
- if $(s_1, s_2, s_3) \in \mathcal{R}$ then $(f(s_1), f(s_2), f(s_3)) \in \mathcal{R}'$

f extends to pseudoterms and contexts
Proposition:

If $\Gamma \vdash M : A$ then $f(\Gamma) \vdash f(M) : f(A)$

Examples:

- $f : \lambda(\mathcal{S}, \mathcal{A}, \mathcal{R}) \to \lambda\star$, $f(s) := \star$. ("initial" PTS)
- $g : \lambda\mathsf{PRED}\omega \to \mathsf{CC}$, $g(\mathsf{Prop}) = g(\mathsf{Set}) := \mathsf{Prop}$, $g(\mathsf{Type}^p) = g(\mathsf{Type}^s) := \mathsf{Type}$.

Corollary: SN for CC $\Rightarrow$ SN for $\lambda\mathsf{PRED}\omega$

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

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

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

They are equivalent:

The PTS-morphism $h : \lambda\mathrm{PRED}\omega \to \lambda\mathrm{HOL}$, given by

$$\begin{array}{lcllcl} h(\mathrm{Prop}) & := & \mathrm{Prop} & h(\mathrm{Set}) & := & \mathrm{Type} \\ h(\mathrm{Type}^p) & := & \mathrm{Type} & h(\mathrm{Type}^s) & := & \mathrm{Type}' \end{array}$$

constitutes an isomorphism between the derivable sequents.

CC^∞
$\mathcal{S}$ Prop, $\{\text{Type}_i\}_{i \in \mathbb{N}}$
$\mathcal{A}$ Prop : Type, $\text{Type}_i : \text{Type}_{i+1}$
$\mathcal{R}$ (Prop, Prop), (Prop, Type_i), (Type_i, Prop)
(Type_i, Type_j, $\text{Type}_{\max(i,j)}$)

NB: ($\text{Type}_{i+1}\text{Type}_i$, Type_i)) would be inconsistent.

CC^∞
$\mathcal{S}$ Prop, $\{\mathsf{Type}_i\}_{i\in\mathbb{N}}$ $\mathcal{A}$ Prop : Type, $\mathsf{Type}_i : \mathsf{Type}_{i+1}$ $\mathcal{R}$ (Prop, Prop), (Prop, Type_i), (Type_i, Prop) (Type_i, Type_j, $\mathsf{Type}_{\max(i,j)}$)

NB: ($\mathsf{Type}_{i+1}\mathsf{Type}_i$, Type_i)) would be inconsistent.

The Extended Calculus of Constructions has in addition

- Cumulativity: $\mathsf{Prop} \subseteq \mathsf{Type}_0 \subseteq \mathsf{Type}_1 \subseteq \ldots$
- Σ-types:

$$\frac{\Gamma \vdash A : \mathsf{Prop} \quad \Gamma, x{:}A \vdash B : \mathsf{Prop}}{\Gamma \vdash \Sigma x{:}A.B : \mathsf{Prop}} \qquad \frac{\Gamma \vdash A : \mathsf{Type}_i \quad \Gamma, x{:}A \vdash B : \mathsf{Type}_j}{\Gamma \vdash \Sigma x{:}A.B : \mathsf{Type}_{\max(i,j)}}$$

NB: We have $\Pi A{:}\mathsf{Type}_i.\varphi : \mathsf{Prop}$, but not $\Sigma A{:}\mathsf{Type}_i.\varphi : \mathsf{Prop}$

NB: Coq has in addition Set : Type and rules (Set, Set), (Type_i, Set), (Set, Prop) and inductive types.

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 many systems at once.

Properties of PTSs.

- Uniqueness of types
  If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A =_\beta B$.
  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 $\Gamma \vdash M : A$ and $M \longrightarrow_\beta N$, then $\Gamma \vdash N : A$.
- Substitution property
  If $\Gamma, x : B, \Delta \vdash M : A$, $\Gamma \vdash P : B$, then
  $\Gamma, \Delta[P/x] \vdash M[P/x] : A[P/x]$.
- Thinning
  If $\Gamma \vdash M : A$ and $\Gamma \subseteq \Delta$, Δ well-formed, then $\Delta \vdash M : A$.
- Strengthening
  If $\Gamma, x : \tau, \Delta \vdash M : A$ and $x \notin \mathsf{FV}(M, A, \Delta)$, then
  $\Gamma, \Delta \vdash M : A$.

Strong Normalization:

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

SN holds for some PTSs (all subsystems of CC,...), and for some not (λU^{-}, $\lambda\star$, ...).

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