Pure Type Systems without Explicit Contexts

Robbert Krebbers
Joint work with Herman Geuvers,
James McKinna and Freek Wiedijk

Institute for Computing and Information Science
Faculty of Science, Radboud University Nijmegen
and
Faculty of Mathematics and Computer Science
Eindhoven University of Technology
The Netherlands

July 14, 2010

The main traditions of type theory

- Descendants of **simple** type theory
  - Church's original system
  - Polymorphic λ-calculus, System F
  - HOL's type theory
  - ...

  Traditionally presented without contexts

- **Dependent** type theories (de Bruijn, Martin-Löf)
  - Automath
  - Berardi/Terlouw framework of Pure Type Systems
  - Coq's type theory
  - ...

  Traditionally presented with contexts

Problem

Traditional presentation of dependent type theory

- Terms considered with respect to an explicit context Γ

$$\Gamma \vdash M : A$$

- A **bound** variable is bound **locally** by a λ or Π
- A **free** variable is bound **globally** by Γ

Can we present dependent type theory without contexts?

Motivation

First-order logic and contexts

Predicate logic	**Type theory**
$$\dfrac{\dfrac{A \vdash P(x)}{A \vdash \forall x. P(x)}}{\vdash A \to \forall x. P(x)}$$	$$\dfrac{\dfrac{H : A, x : D \vdash M_3 : P(x)}{H : A \vdash M_2 : \Pi x : D. P(x)}}{\vdash M_1 : A \to \Pi x : D. P(x)}$$
'sea' of free variables	*context of 'free' variables*

What about?

$$(\forall x.\, P(x)) \to (\exists x.\, P(x))$$

Approach

- We simulate the sea of free variables
- Infinitely many variables x^{A} for each type A
- This gives an "infinite context" called Γ_∞
- For example

$$s^{N^* \to N^*}$$

- Variable carries history of how it comes to be well-typed
- Judgments of the shape $A : B$
- Should be imagined as $\Gamma_\infty \vdash A : B$

Approach

Two kinds of variables: **free** and **bound** variables

Curry	$\lambda x.f\,x$
Church	$\lambda x^A.f^{A\to A}x^A$
Barendregt *et al.*	$\lambda x : A.f\,x$
Γ_∞-style	$\lambda\dot{x} : A^*.f^{A^*\to A^*}\dot{x}$

That is

- Γ_∞ extends Church's approach to dependent types
- But Γ_∞ avoids the need to consider substitution in labels of bound variables

$$(\lambda x^A \lambda P^{A\to *}\lambda y^{P^{A\to *}x^A}\dots)a^A \to_\beta \lambda P^{A\to *}\lambda y^{P^{A\to *}a^A}\dots$$

PTS terms

- The set $\mathcal{T}$ of **pseudo-terms** is defined as

$$\mathcal{T} ::= s \mid \mathcal{V} \mid \Pi\mathcal{V} : \mathcal{T}.\mathcal{T} \mid \lambda\mathcal{V} : \mathcal{T}.\mathcal{T} \mid \mathcal{T}\mathcal{T}$$

- For ordinary PTSs the choice of $\mathcal{V}$ does not matter
- For Γ_∞ we have two kinds of variables

$$\begin{array}{rcl} \mathcal{V} & ::= & \dot{\mathcal{X}} \mid \mathcal{X}^{\mathcal{T}} \\ \mathcal{X} & ::= & x \mid y \mid z \mid \ \ldots \ \mid x_0 \mid x_1 \mid x_2 \mid \ \ldots \end{array}$$

- Variables x^A are *intended* to be **free**
- Variables $\dot{x}$ are *intended* to be **bound**

Labelling terms

- Type labels should be considered as **strings**
- Labels are insensitive to α and β-conversion
- That is to say

$$x^{A}[A := B] \not\equiv x^{B}$$

and

$$\begin{array}{ccc} (\lambda \dot{A} : *. \dot{A})\, B^* & =_\beta & B^* \\ x^{(\lambda \dot{A}:*.\dot{A})B^*} & \neq_\beta & x^{B^*} \end{array}$$

- But we **do** have (by type conversion)

$$x^{(\lambda \dot{A}:*.\dot{A})B^*} : B^*$$

Typing rules

Two of the six rules

PTS rules

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\; x \notin \Gamma$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_3}$$

Γ_∞ rules

$$\frac{A : s}{x^A : A}$$

$$\frac{A : s_1 \qquad B : s_2}{\Pi \dot{x} : A.B[y^A := \dot{x}] : s_3}$$

Remark:

- Binding a variable in Γ_∞

  replace a free variable by a bound variable

- No weakening rule

But this does not correspond to PTSs!

Now we would have

$$\frac{x^{A^*} \;:\; A^*}{\lambda \dot{A} : *. x^{A^*} \;:\; \Pi \dot{A} : *. \dot{A}}$$

but, in ordinary PTS-style

$$\frac{A : *, x : A \vdash \; x \;:\; A}{x : A \vdash \; \lambda A : *. x \;:\; \Pi A : *. A}$$

which is nonsense because A^* occurs free in the label of x.

Taking the type annotations seriously

It is not enough to consider the free variables in a type label, but the *hereditarily* free variables of a type label.

$$\frac{A : s_1 \qquad B : s_2}{\Pi \dot{x} : A.B[y^A := \dot{x}] : s_3} \text{ Incorrect} y^A \notin \mathrm{hfvT}(B)$$

$$\frac{M : B \qquad \Pi \dot{x} : A.B[y^A := \dot{x}] : s}{\lambda \dot{x} : A.M[y^A := \dot{x}] : \Pi \dot{x} : A.B[y^A := \dot{x}]} \; y^A \notin \mathrm{hfvT}(M) \cup \mathrm{hfvT}(B)$$

Taking the type annotations seriously

Hereditarily free type-variables are defined as

$$\begin{array}{rcl}
\mathrm{hfvT}(s) = \mathrm{hfvT}(\dot{x}) & = & \emptyset \\
\mathrm{hfvT}(F\ N) & = & \mathrm{hfvT}(F) \cup \mathrm{hfvT}(N) \\
\mathrm{hfvT}(\lambda\dot{x} : A.N) = \mathrm{hfvT}(\Pi\dot{x} : A.N) & = & \mathrm{hfvT}(A) \cup \mathrm{hfvT}(N) \\
\mathrm{hfvT}(x^A) & = & \mathrm{hfv}(A)
\end{array}$$

Where the **hereditarily free variables** are defined as

$$\begin{array}{rcl}
\mathrm{hfv}(s) = \mathrm{hfv}(\dot{x}) & = & \emptyset \\
\mathrm{hfv}(F\ N) & = & \mathrm{hfv}(F) \cup \mathrm{hfv}(N) \\
\mathrm{hfv}(\lambda\dot{x} : A.N) = \mathrm{hfv}(\Pi\dot{x} : A.N) & = & \mathrm{hfv}(A) \cup \mathrm{hfv}(N) \\
\mathrm{hfv}(x^A) & = & \{x^A\} \cup \mathrm{hfv}(A)
\end{array}$$

The correspondence theorems

$$\textbf{derivable PTS judgment} \quad \longleftrightarrow \quad \textbf{derivable } \Gamma_\infty \textbf{ judgment}$$

(α-)rename $\Gamma \vdash M : A$ to $\Gamma' \vdash M' : A'$ such that $\Gamma' \subset \Gamma_\infty$ and

$$\Gamma \vdash M : A \qquad \Longrightarrow \qquad M' : A'$$

for $M : A$ generate a context $\Gamma(M, A)$ such that

$$\Gamma(M, A) \vdash M : A \qquad \Longleftarrow \qquad M : A$$

Type annotated judgments

A **type annotated judgment** is a judgment of the shape

$$x_1{}^{B_1} : B_1, \ldots, x_n{}^{B_n} : B_n \vdash M : A$$

where

1. all free variables in M and A are of the form $x_i{}^{B_i}$
2. all bound variables in B_i, M and A are of the form $\dot{x}$

Type annotated judgments

Lemma

Every judgment $\Gamma \vdash M : A$ in a PTS can be (α-)renamed to a type annotated judgment $\Gamma' \vdash M' : A'$.

For example consider

$$A : *,\ a : A \ \vdash\ (\lambda x : A.\, x)\, a : A$$

This judgment can be (α-)renamed to

$$A^* : *,\ a^{A^*} : A^* \ \vdash\ (\lambda \dot{x} : A^*.\, \dot{x})\, a^{A^*}$$

Theorem

Let $\Gamma' \vdash M' : A'$ be a derivable type annotated judgment. Then $M' : A'$ is derivable in the corresponding Γ_∞-theory.

The reverse implication

Theorem

Let $M : A$ be derivable in Γ_∞. Then $\Gamma(M, A) \vdash M : A$ is derivable in the corresponding PTS.

- Generate a context $\Gamma(M, A)$ by induction over $M : A$
- For Π, λ, app and conv we have to merge contexts
- The merge of Γ and Δ is defined as $\Gamma, (\Delta \setminus \Gamma)$ if

$$\forall x \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta)(\mathrm{type}_\Gamma(x) \equiv \mathrm{type}_\Delta(x))$$

- So merge is a partial function
- Key lemma: for type annotated judgments merge is total

Possible advantages

- Easier typing rules
- Strengthening is implicit
- Some meta theory is easier to prove
- Closer to implementation?

But is the cost of labelling variables too high?

Future work

- Γ_∞ presentation for other type theories
  - Theories with definitions?
- Implementation based on Γ_∞
  - Efficiency?
  - Extra kind of variables x^A that remain free?
- Formalization
  - Already one direction finished
  - Locally nameless approach
  - Suits distinction between variables well