A Dependent Type Theory with Names and Binding

Ulrich Schöpp Ian Stark

Laboratory for Foundations of Computer Science
University of Edinburgh

December 16, 2004

Names and Binding for Abstract Syntax

Example: Reasoning about the λ-calculus.

Barendregt Variable Convention [Barendregt, 1984]:

> 2.1.12. CONVENTION *Terms that are α-congruent are identified. [...]*
>
> 2.1.13. VARIABLE CONVENTION *If $M_1, \ldots, M_n$ occur in a certain mathematical context (e.g. definition, proof) then in these terms all bound variables are chosen to be different from the free variables. [...]*
>
> 2.1.14. MORAL *Using conventions 2.1.12 and 2.1.13 one can work with λ-terms in the naive way.*

Aim: Support this convention in formal reasoning, in particular in dependent type theory.

We build on *lots* of previous work, e.g. [McKinna, Pollack], [Honsell, Miculan, Scagnetto], [Gabbay, Pitts], ...

A Dependent Type Theory with Names and Binding

We give a dependent type theory with

- Type of names $\mathbf{N}$ with decidable equality,
- Additive Types (Σ, Π, ...),
- Multiplicative Types (Σ^*, Π^*, ...).

Main use of multiplicatives is to represent α-equivalence classes.

(Π^*) Weak Higher Order Abstract Syntax style
[Despeyroux, Felty, Hirschowitz], [Honsell, Miculan, Scagnetto], ...

(Σ^*) FM abstraction set style
[Gabbay, Pitts]

A Bunched Dependent Type Theory

We use *bunches* [O'Hearn, Pym] to integrate the additive structure with the multiplicative structure.

$$\frac{}{\diamond \text{ Bunch}} \qquad \frac{\Gamma \text{ Bunch} \qquad \Gamma \vdash A \text{ Type}}{\Gamma,\, x : A \text{ Bunch}}$$

$$\frac{\Gamma \text{ Bunch} \qquad \Delta \text{ Bunch}}{\Gamma * \Delta \text{ Bunch}}$$

Examples:

$$(x : \mathsf{Lam},\, y : \mathsf{Lam}) * z : \mathsf{Lam}$$

$$((x : A * y : B),\, z : C) * u : D$$

Semantic intuition:

$A{*}B = \{\langle x, y\rangle \colon A{\times}B \mid \ x \text{ and } y \text{ can be built using disjoint name sets}\}$

Monoidal Products (Π^*)

The type $\Pi^* x : A.\, B$ consists of partial functions that are defined on arguments containing just fresh names.

$$\frac{\Gamma * x : A \vdash M : B}{\Gamma \vdash \lambda^* x : A.\, M : \Pi^* x : A.\, B} \qquad \frac{\Gamma \vdash M : \Pi^* x : A.\, B \qquad \Delta \vdash N : A}{\Gamma * \Delta \vdash M @ N : B\,[N/x]}$$

Example (Syntax in WHOAS-style)

$$\mathsf{Lam} ::= \mathsf{var\ of}\ \mathbf{N} \mid \mathsf{app\ of}\ \mathsf{Lam} \times \mathsf{Lam} \mid \mathsf{lam\ of}\ \Pi^* n : \mathbf{N}.\, \mathsf{Lam}$$

Substitution of $R : \mathsf{Lam}$ for $m : \mathbf{N}$.

$$\begin{aligned} \mathsf{subst} &: \mathsf{Lam} \to \mathsf{Lam} \\ \mathsf{subst}(\mathsf{var}(n)) &= \text{if } (n = m) \text{ then } R \text{ else } \mathsf{var}(n) \\ \mathsf{subst}(\mathsf{app}(M, N)) &= \mathsf{app}(\mathsf{subst}(M), \mathsf{subst}(N)) \\ \mathsf{subst}(\mathsf{lam}(M)) &= \mathsf{lam}(\lambda^* n.\, \mathsf{subst}(M @ n)) \end{aligned}$$

Freefrom Types

The type $B^{*M:A}$ consists of all the elements of B that are *free from* the names in the term $M : A$.

- Example:

$$\mathsf{remove} : \Pi n : \mathbf{N}.\ \left(\mathsf{L} \to \mathsf{L}^{*n:\mathbf{N}}\right).$$

- Current syntax is incomplete (for a categorical semantics): In $B^{*M:A}$ both A and B have to be closed.

Monoidal Sums (Σ^*)

The type $\Sigma^* x : A.\, B$ consists of pairs $M.N$ in which the names of M are hidden.

$$\frac{x : A \vdash B \text{ Type} \qquad \Gamma \vdash M : A \qquad \Gamma \vdash N : B\,[M/x]}{\Gamma \vdash \mathrm{bind}(M, N) : (\Sigma^* x : A.\, B)^{*\,M:A}}$$

$$M.N \stackrel{\text{def}}{=} (\text{let } \mathrm{bind}(M, N) \text{ be } y^{*x} \text{ in } y) \ : \Sigma^* x : A.\, B$$

$$\frac{\Gamma \vdash M : \Sigma^* x : A.\, B \qquad (\Gamma * x : A),\, y : B \vdash N : C^{*\,x:A}}{\Gamma \vdash \text{let } M \text{ be } x.y \text{ in } N : C}$$

Example (Syntax in FM-style)

$$\mathsf{Lam} ::= \mathsf{var\ of}\ \mathbf{N} \mid \mathsf{app\ of}\ \mathsf{Lam} \times \mathsf{Lam} \mid \mathsf{lam\ of}\ \Sigma^* n : \mathbf{N}.\, \mathsf{Lam}$$

Substitution.

$$\mathsf{subst}(\mathsf{lam}(M)) = \text{let } M \text{ be } n.x \text{ in } (\text{let } \mathrm{bind}(n, \mathsf{subst}(x)) \text{ be } y^{*n} \text{ in } \mathsf{lam}(y)^{*n})$$

Hidden-Name Types

Both $(\Pi^* n : \mathbf{N}.\, \mathrm{Lam})$ and $(\Sigma^* n : \mathbf{N}.\, \mathrm{Lam})$ represent α-equivalence classes. $\Rightarrow$ They are isomorphic.

Introduce *hidden-name types* as a syntax for both types.

$$\Sigma^* n : \mathbf{N}.\, B \quad \cong \quad \mathrm{H} n.\, B \quad \cong \quad \Pi^* n : \mathbf{N}.\, B$$

Rules and equations for H are those from both Σ^* and Π^* plus interaction equations, such as

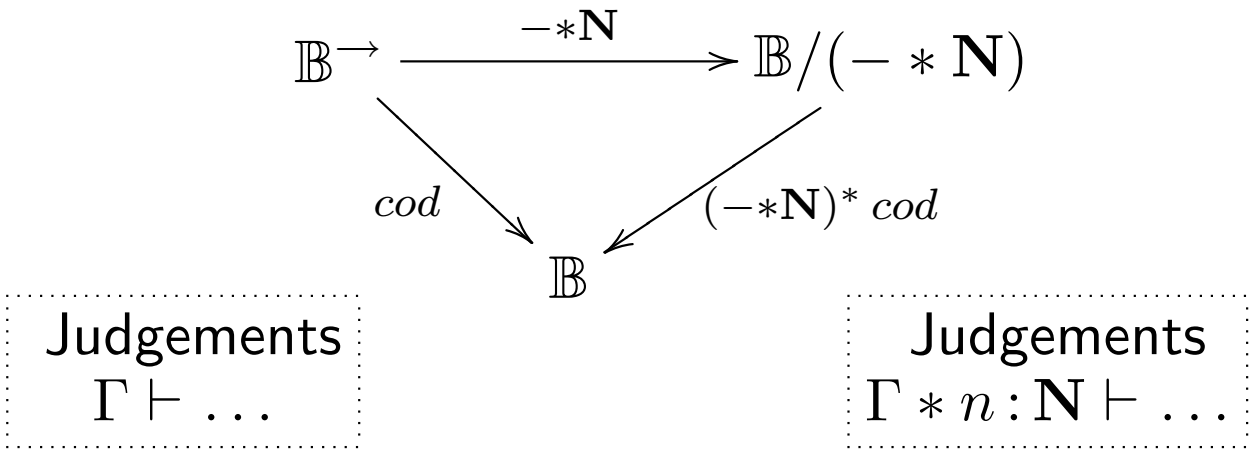
$$\begin{aligned} (n.M)@n &= M \\ n.(N@n) &= N \\ &\ldots \end{aligned}$$

Mixing Σ^* and Π^* is useful:

$$\mathrm{new}\ n.\, M \quad \stackrel{\mathrm{def}}{=} \quad \mathrm{let}\ (\lambda^* n.\, M)\ \mathrm{be}\ n.x\ \mathrm{in}\ x$$

Formalising the Barendregt Variable Convention

- Category $\mathbb{B}$ with pullbacks.
- Names: Object $\mathbf{N}$.
- Freshness: Monoidal structure $*$.
- Working with α-equivalence classes of terms is the same as working with freshly named instances:
  Equivalence of fibrations



Models of this structure: FM-Sets, Realisability version of FM-Sets, Species of Structures.

The equivalence of fibrations

If $- * \mathbf{N} \colon \mathbb{B}^{\to} \to \mathbb{B}/(- * \mathbf{N})$ is an equivalence, then we get two adjunctions

$$\mathrm{H} \dashv (- * \mathbf{N}) \dashv \mathrm{H}$$

related by $\varepsilon' = \eta^{-1}$ and $\eta' = \varepsilon^{-1}$.

H generalises both the abstraction set $[\mathbf{N}]B$ and the freshness quantifier И of FM set theory. Good properties of H, such as

$$\begin{aligned} \mathrm{H}(A + B) &\cong \mathrm{H}A + \mathrm{H}B \\ \mathrm{H}(A \times B) &\cong \mathrm{H}A \times \mathrm{H}B \\ \mathrm{H}(A \to B) &\cong \mathrm{H}A \to \mathrm{H}B, \end{aligned}$$

are immediate, since H is an equivalence.

The equivalence of fibrations

The adjunctions

$$\mathrm{H} \dashv (- * \mathbf{N}) \dashv \mathrm{H}$$

are a special case of

$$\Sigma_A^* \dashv (- * A) \dashv \Pi_A^*,$$

on which the rules for Σ^* and Π^* are modelled.
Freefrom types are the syntax for $- * A$.

We get

$$\Sigma_{\mathbf{N}}^* \cong \mathrm{H} \cong \Pi_{\mathbf{N}}^*.$$

The equations $\varepsilon' = \eta^{-1}$ and $\eta' = \varepsilon^{-1}$ are the interaction equations for hidden-name types: $(n.M)@n = M$, ...

Conclusion and Further Work

Conclusion

Σ^*, H and Π^* and are good for names!

Further Work

- Better freefrom types.
- Decidability questions.
- Practicable?
- Semantics
  - Other models?
  - Free construction of H?
  - Model ∇?
- ...