Slide 1

Project TOSCA

TASK HOAS_LF

Towards a general framework for metareasoning on HOAS encodings

Anna Bucalo, Martin Hofmann, Furio Honsell,
Marino Miculan, Ivan Scagnetto

Università di Udine

Slide 2

How to represent binding operators?

Scenario: we have to represent formally (*encode*) an object language (e.g., π-calculus) in some logical framework for doing formal (meta)reasoning

Problem: how to render binding operators (e.g, ν) efficiently?

- **First-order abstract syntax**

```
nu : Name -> Proc -> Proc
```

  Needs lots of machinery about α-equivalence, substitution, …

- **de Bruijn indexes**

```
nu : Proc -> Proc
```

  Good at α-equivalence, but not immediate to understand and needs even more machinery for capture-avoiding substitution than FOAS

Slide 3

Higher-order abstract syntax [Harper, Honsell, Plotkin 87]

```
nu : (Name -> Proc) -> Proc
```

♡ it delegates successfully many aspects of names management to the metalanguage (α-conversion, capture-avoiding substitution, generation of fresh names,...) $\Rightarrow$ widely used in most logical frameworks

♠ if `Name` is defined as inductive then *exotic terms* (= not corresponding to any real process of the object language) will arise!

```
weird = nu [x:Nat](Cases x of 0 => P
                             | _ => P|Q   end).
```

♠ usually structural induction over higher-order terms (*contexts*, terms with *holes*) is not provided $\Rightarrow$ metatheoretic analysis is difficult/impossible

Slide 4

A methodology for HOAS metareasoning

We propose a general methodology for dealing with metatheoretic properties of contexts in HOAS-based encodings.

Let Υ be a framework metalanguage corresponding to a theory of Simple Types/Classical Higher-Order Logic *à la Church*. Types:

$\tau ::=$	o	propositions
$\mid$	υ	names
$\mid$	ι	object language terms (e.g., processes)
$\mid$	$\tau \to \tau$	

Two judgements: type assignments and validity derivations

$$\Gamma \vdash_\Sigma M : \tau \qquad \Gamma \vdash_\Sigma P$$

Slide 5

The logical framework Υ: Syntax

Two basic logical connectives:

$$\Rightarrow : o \to o \to o \qquad \forall_\tau : (\tau \to o) \to o$$

Other logical connectives and Leibniz equality are defined as abbreviations, as usual

$$\begin{array}{rcl rcl}
\forall x^\tau . p & \stackrel{\text{def}}{=} & \forall_\tau(\lambda x^\tau . p) & p \wedge q & \stackrel{\text{def}}{=} & \neg(p \Rightarrow \neg q) \\
\bot & \stackrel{\text{def}}{=} & \forall r^o . r & p \vee q & \stackrel{\text{def}}{=} & \neg p \Rightarrow q \\
\neg p & \stackrel{\text{def}}{=} & p \Rightarrow \bot & p \Leftrightarrow q & \stackrel{\text{def}}{=} & (p \Rightarrow q) \wedge (q \Rightarrow p) \\
\exists x^\tau . p & \stackrel{\text{def}}{=} & \neg \forall x^\tau . \neg p & M =^\tau N & \stackrel{\text{def}}{=} & \forall R^{\tau \to o} .\, RM \Rightarrow RN
\end{array}$$

Slide 6

The logical framework Υ: Typing and Logical rules

Typing rules:

$$\frac{-}{\Gamma, x : \tau \vdash_\Sigma x : \tau} \qquad \text{(VAR)}$$

$$\frac{-}{\Gamma \vdash_\Sigma M : \tau}(M : \tau) \in \Sigma \qquad \text{(CONST)}$$

$$\frac{\Gamma \vdash_\Sigma M : \tau' \to \tau \quad \Gamma \vdash_\Sigma N : \tau'}{\Gamma \vdash_\Sigma MN : \tau} \qquad \text{(APP)}$$

$$\frac{\Gamma, x : \tau' \vdash_\Sigma M : \tau}{\Gamma \vdash_\Sigma \lambda x^{\tau'} . M : \tau' \to \tau} \qquad \text{(ABS)}$$

Logical rules (next slide)

Slide 7

$$\frac{\Gamma \vdash_\Sigma p : o \quad \Gamma \vdash_\Sigma q : o \quad \Gamma \vdash_\Sigma r : o}{\Gamma \vdash_\Sigma (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow p \Rightarrow r} \quad \text{(S)}$$

$$\frac{\Gamma \vdash_\Sigma p : o \quad \Gamma \vdash_\Sigma q : o}{\Gamma \vdash_\Sigma p \Rightarrow q \Rightarrow p} \quad \text{(K)}$$

$$\frac{\Gamma \vdash_\Sigma P : \tau \to o \quad \Gamma \vdash_\Sigma M : \tau}{\Gamma \vdash_\Sigma \forall_\tau(P) \Rightarrow PM} \quad (\forall\text{-E})$$

$$\frac{\Gamma \vdash_\Sigma p : o}{\Gamma \vdash_\Sigma \neg\neg p \Rightarrow p} \quad \text{(DN)}$$

$$\frac{\Gamma, x : \tau \vdash_\Sigma M : \sigma \quad \Gamma \vdash_\Sigma N : \tau}{\Gamma \vdash_\Sigma (\lambda x^\tau.M)N =^\sigma M[N/x]} \quad (\beta)$$

$$\frac{\Gamma \vdash_\Sigma M : \tau \to \sigma}{\Gamma \vdash_\Sigma \lambda x^\tau.Mx =^{\tau\to\sigma} M} x \notin FV(M) \quad (\eta)$$

$$\frac{\Gamma, x : \sigma \vdash_\Sigma M =^\tau N}{\Gamma \vdash_\Sigma \lambda x^\sigma.M =^{\sigma\to\tau} \lambda x^\sigma.N} \quad (\xi)$$

$$\frac{\Gamma \vdash_\Sigma p \Rightarrow q \quad \Gamma \vdash_\Sigma p}{\Gamma \vdash_\Sigma q} \quad \text{(MP)}$$

$$\frac{\Gamma \vdash_\Sigma p : o \quad \Gamma, x : \tau \vdash_\Sigma p \Rightarrow q}{\Gamma \vdash_\Sigma p \Rightarrow \forall x^\tau.q} \quad \text{(Gen)}$$

Slide 8

An example encoding Σ in the logical framework Υ

Example of object language $\mathcal{L}$:

$$P ::= 0 \mid \tau.P \mid P_1 | P_2 \mid [x \neq y]P \mid (\nu x)P$$

Corresponding signature Σ:

$$\begin{array}{rcl} 0 & : & \iota \\ \tau & : & \iota \to \iota \\ | & : & \iota \to \iota \to \iota \\ [\cdot \neq \cdot]\cdot & : & \upsilon \to \upsilon \to \iota \to \iota \\ \nu & : & (\upsilon \to \iota) \to \iota \end{array} \qquad \begin{array}{rcl} Rec^\iota_\tau & : & \tau \to (\tau \to \tau) \to \\ & & (\tau \to \tau \to \tau) \to \\ & & (\upsilon \to \upsilon \to \tau \to \tau) \to \\ & & ((\upsilon \to \tau) \to \tau) \\ & & \to \iota \to \tau \end{array}$$

Slide 9

Σ^{++}: A Theory of Contexts on Σ

The theory of contexts Σ^{++} is obtained by adding to Σ the following three axioms:

$$\frac{\Gamma \vdash_\Sigma P : \iota}{\Gamma \vdash_\Sigma \exists x^\upsilon . x \not\in P} \quad (\mathrm{Unsat}^\upsilon_\iota)$$

$$\frac{\Gamma \vdash_\Sigma P : \upsilon^n \to \iota \qquad \Gamma \vdash_\Sigma x : \upsilon}{\Gamma \vdash_\Sigma \exists Q^{\upsilon^{n+1}\to\iota} . x \not\in^{n+1} Q \wedge P =^{\upsilon^n \to \iota} (Q\, x)} \quad (\beta\_exp^{\upsilon^n\to\iota})$$

$$\frac{\Gamma \vdash_\Sigma P : \upsilon^{n+1} \to \iota \qquad \Gamma \vdash_\Sigma Q : \upsilon^{n+1} \to \iota \qquad \Gamma \vdash_\Sigma x : \upsilon}{\Gamma \vdash_\Sigma x \not\in^{n+1} P \Rightarrow x \not\in^{n+1} Q \Rightarrow (P\, x) =^{\upsilon^n\to\iota} (Q\, x) \Rightarrow P =^{\upsilon^{n+1}\to\iota} Q} \quad (\mathrm{Ext}^{\upsilon^{n+1}\to\iota})$$

Questions:

- Expressivity: are these axioms *really* useful? $\Rightarrow$ case studies
- Soundness: are these axioms consistent? $\Rightarrow$ a model for HOAS
- Completeness: in what sense?

Slide 10

Case studies: π-calculus

Full language, with recursion and mismatch.

- Encoded the full theory (transition system, strong late bisimulation)
- Proved all main results in *A calculus of mobile processes* by Milner, Parrow, Walker (algebraic laws and Lemmata 1–7).

  In particular: For p, q processes, x, y names, $x \not\in p$:

  Lemma 3 if $p \xrightarrow{\alpha} q$ then $p[x/y] \xrightarrow{\alpha[x/y]} q[x/y]$

  Lemma 6 if $p \sim q$ then $p[x/y] \sim q[x/y]$

  Both are instances of the general property (cf. Cardelli)

  If $\Gamma \vdash_\Sigma P$ then for all h injective: $\Gamma[h] \vdash_\Sigma P[h]$

  Both these name replacements are readily encoded by applications of higher-order terms to names.

Slide 11

Case studies: λ-calculus

Both call-by-name and call-by-value, simply typed:

- full theory: substitution, small step and big step semantics, typing system
- functionality of substitution relation (totality and determinism)
- equivalence of small step and big step semantics
- confluence of big step semantics
- subject reduction

Slide 12

Substitution of the λ-calculus as a (functional) relation

```
Inductive subst [N:tm] : (var->tm) -> tm -> Prop :=
     subst_var  : (subst N isvar N)
   | subst_void : (y:var)(subst N [_:var]y y)
   | subst_App  : (M1,M2:var->tm)(M1',M2':tm)
           (subst N M1 M1') -> (subst N M2 M2') ->
           (subst N [y:var](App (M1 y) (M2 y)) (App M1' M2'))
   | subst_Lam  : (M:var->var->tm)(M':var->tm)
           ((z:var)(subst N [y:var](M y z) (M' z))) ->
           (subst N [y:var](Lam (M y)) (Lam M')).
```

Axioms used for proving

- Determinism: Ext^{ι}, $\mathrm{Ext}^{\upsilon\to\iota}$, $\mathrm{Unsat}^{\upsilon}_{\iota}$
- Totality: higher-order recursion

Slide 13

Other case studies (minor/work in progress)

First Order Logic full theory: validity judgement, substitution;
metatheory: functionality of substitution.

spi calculus full theory; metatheory: some algebraic laws

ν**-calculus** theory

$\lambda\sigma$**-calculus** theory; some metatheoretic result

Slide 14

Towards a categorical model

In order to interpret a HOAS signature in a model based on functor categories, we adopt the following protocol [Hof99]:

- the metalanguage is interpreted in a suitable functor category $\check{\mathcal{V}} \stackrel{\text{def}}{=} \mathcal{S}et^{\mathcal{V}}$ such that
  - if a constructor type contains a negative occurrence of a given type, the latter must have a representable interpretation (e.g. since we have $\nu : (\upsilon \to \iota) \to \iota$, $[\![\upsilon]\!]$ must be representable, i.e., $[\![\upsilon]\!] \cong \check{\mathcal{Y}}(X)$ for some X);
- the structure of functional types will be unraveled by means of the equation $\check{\mathcal{Y}}(X) \Rightarrow A \cong A^X$, where $A^X_Y \stackrel{\text{def}}{=} A_{X \uplus Y}$.

Slide 15

The model $\mathcal{U}$

The ambient category is $\check{\mathcal{V}} \stackrel{\text{def}}{=} \mathcal{S}et^{\mathcal{V}}$, where $\mathcal{V}$ is defined as follows:

- objects are finite sets of variables;
- morphisms are substitution of variables for variables.

The model $\mathcal{U}$ of Υ is defined by means of the following protocol:

- types and contexts are interpreted as covariant presheaves: $[\![\tau]\!] \in Obj(\check{\mathcal{V}})$ and $[\![\Gamma]\!] \in Obj(\check{\mathcal{V}})$;
- terms are interpreted as natural transformations: $[\![\Gamma \vdash_\Sigma M : \tau]\!] \in \check{\mathcal{V}}([\![\Gamma]\!], [\![\tau]\!])$;

Slide 16

Interpreting basic datatypes

- $[\![\upsilon]\!] \stackrel{\text{def}}{=} Var : \mathcal{V} \longrightarrow \mathcal{S}et$ defined as follows:

$$Var_X \stackrel{\text{def}}{=} X \qquad Var_h(x) \stackrel{\text{def}}{=} h(x), \quad \text{for } x \in X, h \in \mathcal{V}(X,Y)$$

  Hence, it is isomorphic to the representable functor $\check{\mathcal{Y}}(\{\star\})$.

- $[\![\iota]\!] \stackrel{\text{def}}{=} Proc : \mathcal{V} \longrightarrow \mathcal{S}et$ defined as follows:

$$Proc_X \stackrel{\text{def}}{=} \{P \mid FV(P) \subseteq X\}$$

$$Proc_h(P) \stackrel{\text{def}}{=} P[h], \quad \text{for } P \in Proc_X, h \in \mathcal{V}(X,Y)$$

  $Proc$ is not representable

Prop.: For all n, $Var^n \Rightarrow Proc$ is an initial algebra for a suitable functor.

Slide 17

Toposes are not enough

Being $\check{\mathcal{V}} \stackrel{\text{def}}{=} \mathcal{S}et^{\mathcal{V}}$ a topos, we could use the canonical interpretation for the propositions type:

$$\begin{aligned} [\![o]\!]_X &\stackrel{\text{def}}{=} Sub(\check{\mathcal{Y}}(X)) = Sub(\mathcal{V}(X, \_)) \\ [\![o]\!]_f(S) &\stackrel{\text{def}}{=} \{g \in Arr(\mathcal{V}) \mid dom(g) = Y \text{ and } g \circ f \in S\} \end{aligned}$$

(where $f : X \longrightarrow Y$ and $S \in [\![o]\!]_X$)

However, this does not work because the axiom of unique choice would be validated:

$$\begin{aligned} AC!_{\sigma,\tau} \quad : \quad & (\forall a^\sigma. \exists ! b^\tau. \, R(a, b)) \Rightarrow \\ & \exists ! f^{\sigma \to \tau}. \, \forall a^\sigma. \, R(a, f(a)) \end{aligned}$$

Slide 18

Toposes are not enough (cont.)

whence:

- $AC!$ allows to derive the characteristic function of the equality over names $eq : \upsilon \to \upsilon \to nat$ (defined by $\forall x, y : \upsilon.\, x = y \Leftrightarrow eq(x, y) = 1$, where $=$ is Leibniz equality);
- $Q \stackrel{\text{def}}{=} \lambda x^\upsilon.\, if\ eq(x, y)\ then\ p\ else\ q$ (where $y : \upsilon$ e $p, q : \iota$);
- using $Ext^{\upsilon \to \iota}$ one can prove that $Q =^{\upsilon \to \iota} \lambda x^\upsilon.\, q$;
- hence it is possible to show that all processes are syntactically equal (absurd).

Slide 19

Interpreting o in $\check{\mathcal{V}}$

Given $F \in \check{\mathcal{V}}$, predicates over F (**Pred**(F)) are $\mathcal{V}$-indexed familes of sets $\{P_X\}_{X\in\mathcal{V}}$ such that:

1. $P_X \subseteq F_X$ where $X \in \mathcal{V}$;
2. for all $h \in \mathcal{I}(X, Y)$, if $f \in P_X$ then $F_h(f) \in P_Y$;
3. if $f \in F_X$ and $F_h(f) \in P_Y$ for some $h \in \mathcal{I}(X, Y)$, then $f \in P_X$.

Then we can define $[\![o]\!] \stackrel{\text{def}}{=} Prop$, where $Prop_X \stackrel{\text{def}}{=} \textbf{Pred}(\check{\mathcal{Y}}(X))$

For each X, $Prop_X$ is a Boolean algebra, where order is given by (pointwise) inclusion.

Slide 20

Interpreting o in $\check{\mathcal{V}}$: the formal justification

This approach can be explained by the existence of the adjunction $(\cdot)^r \dashv (\cdot)^*$, where $(\cdot)^r : \check{\mathcal{V}} \longrightarrow \check{\mathcal{I}}$ and $\check{\mathcal{I}} \stackrel{\text{def}}{=} \mathcal{S}et^{\mathcal{I}}$:

- objects of $\mathcal{I}$ are finite sets of variables;
- morphisms of $\mathcal{I}$ are *injective* substitution of variables for variables.

Indeed we have the following:

$$\textbf{Pred}(F) \stackrel{\text{def}}{=} \textbf{Pred}_{\check{\mathcal{I}}}(F^r) \cong \check{\mathcal{I}}(F^r, \Omega) \cong \check{\mathcal{V}}(F, \Omega^*)$$

Hence, choosing $F = \check{\mathcal{Y}}(X)$, we have

$$\textbf{Pred}(\check{\mathcal{Y}}(X)) \cong \check{\mathcal{V}}(\check{\mathcal{Y}}(X), \Omega^*) \cong \Omega^*_X$$

This suggests to take $[\![o]\!] \stackrel{\text{def}}{=} Prop$, where $Prop_X \stackrel{\text{def}}{=} \textbf{Pred}(\check{\mathcal{Y}}(X))$.

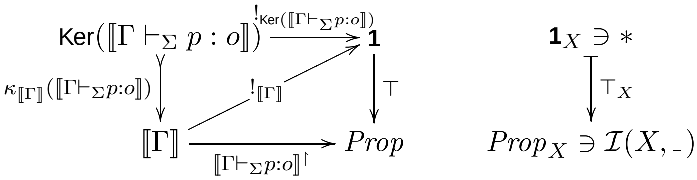
Interpreting the truth judgment

Slide 21

$\Gamma \vdash_\Sigma p$ holds iff for all $X \in \mathcal{V}$ and $\eta \in [\![\Gamma]\!]_X$ we have

$$[\![\Gamma \vdash_\Sigma p : o]\!]_X(\eta) \geq \mathcal{I}(X, \_).$$

Intuitive meaning: proposition p holds on (a tuple of) terms η if it is preserved at least by all injective substitutions ($\mathcal{I}(X, \_)$).



$$[\![\Gamma]\!]_X \ni \eta \longmapsto [\![\Gamma \vdash_\Sigma p : o]\!]_X(\eta) \wedge \mathcal{I}(X, \_)$$

Forcing

Slide 22

Given $X \in \mathcal{V}$, Γ, $\eta \in [\![\Gamma]\!]_X$, and p such that $\Gamma \vdash_\Gamma p : o$ the *forcing judgment*

$$X \Vdash_{\Gamma,\eta} p$$

stands for $\eta \in \kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!])_X$ (i.e., $[\![\Gamma \vdash_\Sigma p : o]\!]_X(\eta) \geq \mathcal{I}(X, \_)$).

The forcing judgment is a powerful tool allowing to streamline the computation of the truth value of propositions:

$\Gamma \vdash_\Sigma p$ is valid iff for all $X \in \mathcal{V}$ and $\eta \in [\![\Gamma]\!]_X$ we have $X \Vdash_{\Gamma,\eta} p$.

Slide 23

Some properties derived by means of forcing

- $X \Vdash_{\Gamma,\eta} \forall x^\tau.p$ iff for all Y, $h \in \mathcal{I}(X,Y)$, $a \in [\![\tau]\!]_Y$ we have $Y \Vdash_{(\Gamma,x:\tau),\ \langle[\![\Gamma]\!]_h(\eta),a\rangle}\ p$;
- $X \Vdash_{\Gamma,\eta} p \Rightarrow q$ iff $X \Vdash_{\Gamma,\eta} p$ implies $X \Vdash_{\Gamma,\eta} q$;
- it is never the case that $X \Vdash_{\Gamma,\eta} \bot$.
- $X \Vdash_{\Gamma,\eta} \neg p$ iff it is never the case that $X \Vdash_{\Gamma,\eta} p$;
- $X \Vdash_{\Gamma,\eta} p \wedge q$ iff $X \Vdash_{\Gamma,\eta} p$ and $X \Vdash_{\Gamma,\eta} q$;
- $X \Vdash_{\Gamma,\eta} p \vee q$ iff $X \Vdash_{\Gamma,\eta} p$ or $X \Vdash_{\Gamma,\eta} q$;
- $X \Vdash_{\Gamma,\eta} \exists x^\tau.p$ iff there exist Y, $h \in \mathcal{I}(X,Y)$, $a \in [\![\tau]\!]_Y$ s.t. $Y \Vdash_{(\Gamma,x:\tau),\ \langle[\![\Gamma]\!]_h(\eta),a\rangle}\ p$.
- For all Γ, M, N, X and $\eta \in [\![\Gamma]\!]_X$:

$$X \Vdash_{\Gamma,\eta} M =^\tau N \quad\Longleftrightarrow\quad [\![\Gamma \vdash_\Sigma M : \tau]\!]_X(\eta) = [\![\Gamma \vdash_\Sigma N : \tau]\!]_X(\eta)$$

Slide 24

The model validates the theory of contexts

Using forcing, all HOAS axioms have been verified.

Unsat$^\upsilon_\iota$: if $\Gamma \vdash_\Sigma P : \iota$, then for all $X, \eta \in [\![\Gamma]\!]_X$: $X \Vdash_{\Gamma,\eta} \exists x^\upsilon.x \notin P$.

Ext$^{\upsilon\to\iota}$: if $\Gamma \vdash_\Sigma P : \upsilon \to \iota$, $\Gamma \vdash_\Sigma Q : \upsilon \to \iota$ and $\Gamma \vdash_\Sigma x : \upsilon$, then for all X, $\eta \in [\![\Gamma]\!]_X$: $X \Vdash_{\Gamma,\eta} x \notin^1 P \Rightarrow x \notin^1 Q \Rightarrow (P\,x) =^\iota (Q\,x) \Rightarrow P = Q$.

β_exp^ι: if $\Gamma \vdash_\Sigma P : \iota$ and $\Gamma \vdash_\Sigma x : \upsilon$, then for all $X, \eta \in [\![\Gamma]\!]_X$: $X \Vdash_{\Gamma,\eta} \exists Q^{\upsilon\to\iota}.x \notin^1 Q \wedge P =^\iota (Q\,x)$.

Closure under injective substitutions

If $\Gamma \vdash_\Sigma P$ then for all h injective: $\Gamma[h] \vdash_\Sigma P[h]$

corresponds to monotonicity of forcing:

if $X \Vdash_{\Gamma,\eta} P$ then for all $Y, h \in \mathcal{I}(X,Y)$: $Y \Vdash_{\Gamma,\eta[h]} P$.

Slide 25

AC! is not validated by $\mathcal{U}$

Suppose AC! true in the model $\mathcal{U}$. Since

$$y : \upsilon \vdash \forall x^{\upsilon} \exists ! n^{nat}. x =^{\upsilon} y \iff n =^{nat} 1$$

holds, by AC! we have

$$y : \upsilon \vdash \exists f^{\upsilon \to nat}. x =^{\upsilon} y \iff (f\ x) =^{nat} 1$$

that is: for all Y and $y' \in Y$, there exists Z, $h \in \mathcal{I}(Y, Z)$ and $g \in (\mathit{Var} \Rightarrow nat)_Z$ such that for all X, $h' \in \mathcal{I}(Z, X)$, $x' \in X$:

$$Y \Vdash_{y,f,x;y'[h;h'],g[h'],x'} x =^{\upsilon} y \iff (f\ x) = 1$$

But this condition means that g is not a natural transformation $\Rightarrow$ contradiction.

Slide 26

Recursion and induction principles over $\upsilon^n \to \iota$ are validated

$$\frac{\Gamma \vdash_\Sigma R : (\upsilon^n \to \iota) \to o}{\begin{array}{l}\Gamma \vdash_\Sigma (R\, \lambda \vec{x}^{\upsilon}.0) \Rightarrow (\forall P^{\upsilon^n \to \iota}.(R\, P) \Rightarrow (R\, \lambda \vec{x}^{\upsilon}.(\tau.(P\, \vec{x})))) \Rightarrow \\ \quad (\forall P^{\upsilon^n \to \iota}.(R\, P) \Rightarrow \forall Q^{\upsilon^n \to \iota}.(R\, Q) \Rightarrow (R\, \lambda \vec{x}^{\upsilon}.(P\, \vec{x}) | (Q\, \vec{x}))) \Rightarrow \\ \quad (\forall y^{\upsilon}.\forall z^{\upsilon}.\forall P^{\upsilon^n \to \iota}.(R\, P) \Rightarrow \\ \quad\ (R\, \lambda \vec{x}^{\upsilon}.[x_1 \neq x_1](P\, \vec{x})) \wedge \cdots \wedge (R\, \lambda \vec{x}^{\upsilon}.[x_i \neq x_j](P\, \vec{x})) \wedge \cdots \\ \quad\ \cdots \wedge (R\, \lambda \vec{x}^{\upsilon}.[x_n \neq x_n](P\, \vec{x})) \wedge \\ \quad\ (R\, \lambda \vec{x}^{\upsilon}.[y \neq x_1](P\, \vec{x})) \wedge \cdots \wedge (R\, \lambda \vec{x}^{\upsilon}.[y \neq x_n](P\, \vec{x})) \wedge \\ \quad\ (R\, \lambda \vec{x}^{\upsilon}.[x_1 \neq z](P\, \vec{x})) \wedge \cdots \wedge (R\, \lambda \vec{x}^{\upsilon}.[x_n \neq z](P\, \vec{x})) \wedge \\ \quad\ (R\, \lambda \vec{x}^{\upsilon}.[y \neq z](P\, \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^{n+1} \to \iota}.(\forall y^{\upsilon}.(R\, \lambda \vec{x}^{\upsilon}.(P\, \vec{x}\, y))) \Rightarrow (R\, \lambda \vec{x}^{\upsilon}.\nu(P\, \vec{x}))) \Rightarrow \\ \quad \forall P^{\upsilon^n \to \iota}.(R\, P)\end{array}} \ (Ind^{\upsilon^n \to \iota})$$

Slide 27

Related work

- $FO\lambda^{\Delta N}$ by McDowell-Miller (LICS'97) is a metalogic where induction principles are derived from induction over natural numbers.
- Gabbay and Pitts (LICS'99) introduced a language of contexts based on permutative renaming. "New" quantifier, similar to both $\forall$ and $\exists$

$$\frac{\Gamma, y\#\vec{x} \vdash \phi}{\Gamma \vdash \text{И}y.\phi} \qquad \frac{\Gamma \vdash \text{И}y.\phi \quad \Gamma, \phi, y\#\vec{x} \vdash \psi}{\Gamma \vdash \psi}$$

In the theory of contexts, $\text{И}y.\phi$ is definable as

$$\text{И}y.\phi \quad \equiv \quad \forall y^{\upsilon}.y \not\in^1 (\lambda y^{\upsilon}.\phi) \Rightarrow \phi \quad \equiv \quad \exists y^{\upsilon}.y \not\in^1 (\lambda y^{\upsilon}.\phi) \wedge \phi$$

and the rules above are derivable.

Slide 28

Conclusions and future work

The proposed theory of context is quite expressive, sound and modular.

The model is the basis for future extensions

Future work:

- Expressivity: more case studies (ambient calculus)
- Extending the model to dependent types (useful for dealing with higher-order proof objects, e.g., natural deduction derivations)
- Extending the model to capture-avoiding substitutions of terms for variables
- Realizability semantics (constructive logic)