

# Typage et déduction dans le calcul de réécriture

Benjamin Wack

Encadrants : C. Kirchner, L. Liquori



# Deduction and computation

- $\lambda$ -calculus [**Church 40**] is a simple and powerful computational model
  - ▶ Explicit notions of function, application, binding
  - ▶ Turing equivalent

# Deduction and computation

- $\lambda$ -calculus [**Church 40**] is a simple and powerful computational model
  - ▶ Explicit notions of function, application, binding
  - ▶ Turing equivalent
- Simply typed  $\lambda$ -calculus [**Church 40, Curry 34**]
  - ▶ Ensures strong normalization
  - ▶ Isomorphism with natural deduction for intuitionistic logic [**Curry, Howard, de Bruijn**]

# Deduction and computation

- $\lambda$ -calculus [**Church 40**] is a simple and powerful computational model
  - ▶ Explicit notions of function, application, binding
  - ▶ Turing equivalent
- Simply typed  $\lambda$ -calculus [**Church 40, Curry 34**]
  - ▶ Ensures strong normalization
  - ▶ Isomorphism with natural deduction for intuitionistic logic [**Curry, Howard, de Bruijn**]
- Various extensions [**de Bruijn 70, Girard 72, Coquand 85, Berardi 88, Paulin 90**]
  - ▶ To broaden the expressiveness of the logic
  - ▶ To ease the definition of elaborated functions

## More computational power ?

- Explicit introduction of rewriting in the system [Breazu-Tannen, Jouannaud, Okada *et al.*]
  - ▶ Term rewriting
  - ▶ Higher-order rewriting

## More computational power ?

- Explicit introduction of rewriting in the system [**Breazu-Tannen, Jouannaud, Okada *et al.***]
  - ▶ Term rewriting
  - ▶ Higher-order rewriting
- Removal of computational arguments from formal proofs
  - ▶ Poincaré principle [**Barendregt & Barendsen**]
  - ▶ Deduction modulo [**Dowek, Hardin, Kirchner, Werner**]

## More computational power ?

- **Explicit introduction of rewriting in the system [Breazu-Tannen, Jouannaud, Okada *et al.*]**
  - ▶ Term rewriting
  - ▶ Higher-order rewriting
- **Removal of computational arguments from formal proofs**
  - ▶ Poincaré principle [Barendregt & Barendsen]
  - ▶ Deduction modulo [Dowek, Hardin, Kirchner, Werner]
- **The rewriting calculus [Cirstea, Kirchner, Liquori *et al.*]**
  - ▶ Designed as a semantics for rule-based languages
  - ▶ Embeds the  $\lambda$ -calculus and various aspects of rewriting

# Contents

1. Untyped rewriting calculus
2. Type systems for programming
  - ▶ Properties and type inference
  - ▶ Typed encoding of term rewriting systems
3. Pure Pattern Type Systems
  - ▶ Strong normalization in  $\rho_{\rightarrow}$  and  $\rho P$
4. Using the  $\rho$ -calculus for deduction
  - ▶  $P^2TS$ -proof terms for deduction modulo
  - ▶ Generalized Natural Deduction



## The Untyped Syntax

$\mathcal{P} \subseteq \mathcal{T}$	Patterns
$\mathcal{T} ::= \mathcal{X} \mid \mathcal{K} \mid \lambda \mathcal{P}.\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \mathcal{T} \wr \mathcal{T}$	Terms

1.  $\lambda P.A$  denotes an *abstraction* with pattern  $P$  and body  $A$   
... the free variables of  $P$  are bound in  $A$
2. The terms can also be *structures* built using the symbol “ $\wr$ ”
3. We work modulo  *$\alpha$ -conversion* and Barendregt’s *hygiene-convention*

## Some $\rho$ -terms

$(\lambda x.x x) (\lambda x.x x)$

the  $\lambda$ -term  $(\omega\omega)$

$(\lambda(f x y).(g y x)) (f a b)$

the application of a rewrite rule

$(\lambda a.b \ \lambda a.c) a$

the parallel application of two rules

## Some $\rho$ -terms

$(\lambda x.x x) (\lambda x.x x)$

the  $\lambda$ -term  $(\omega\omega)$

$(\lambda(f x y).(g y x)) (f a b)$

the application of a rewrite rule

$(\lambda a.b \ \lambda a.c) a$

the parallel application of two rules

## Some $\rho$ -terms

$(\lambda x.x x) (\lambda x.x x)$

the  $\lambda$ -term  $(\omega\omega)$

$(\lambda(f x y).(g y x)) (f a b)$

the application of a rewrite rule

$(\lambda a.b \wr \lambda a.c) a$

the parallel application of two rules

## The Small-step Reduction Semantics

$$(\lambda P.A) B \rightarrow_{\rho} A\theta \quad \text{if } P\theta \equiv B$$

$$(A \wr B) C \rightarrow_{\delta} AC \wr BC$$

## Some $\rho$ -reductions

$(\lambda x.x x) (\lambda x.x x)$

$(\lambda(f x y).g y x) (f a b)$

$(\lambda a.b \ \lambda a.c) a$

## Some $\rho$ -reductions

$$(\lambda x.x x) (\lambda x.x x) \quad \mapsto_{\rho} \{\omega \omega\} \mapsto_{\rho\delta} \dots$$

$$(\lambda(f x y).g y x) (f a b)$$

$$(\lambda a.b \ \lambda a.c) a$$

## Some $\rho$ -reductions

$$(\lambda x.x x) (\lambda x.x x) \quad \mapsto_{\rho} \{\omega \omega\} \mapsto_{\rho\delta} \dots$$

$$(\lambda(f x y).g y x) (f a b) \quad \mapsto_{\rho} g b a$$

$$(\lambda a.b \ \lambda a.c) a$$



## Some $\rho$ -reductions

$$(\lambda x.x x) (\lambda x.x x) \mapsto_{\rho} \{\omega \omega\} \mapsto_{\rho\delta} \dots$$

$$(\lambda(f x y).g y x) (f a b) \mapsto_{\rho} g b a$$

$$(\lambda a.b \wr \lambda a.c) a \mapsto_{\delta} (\lambda a.b) a \wr (\lambda a.c) a \mapsto_{\rho} b \wr c$$

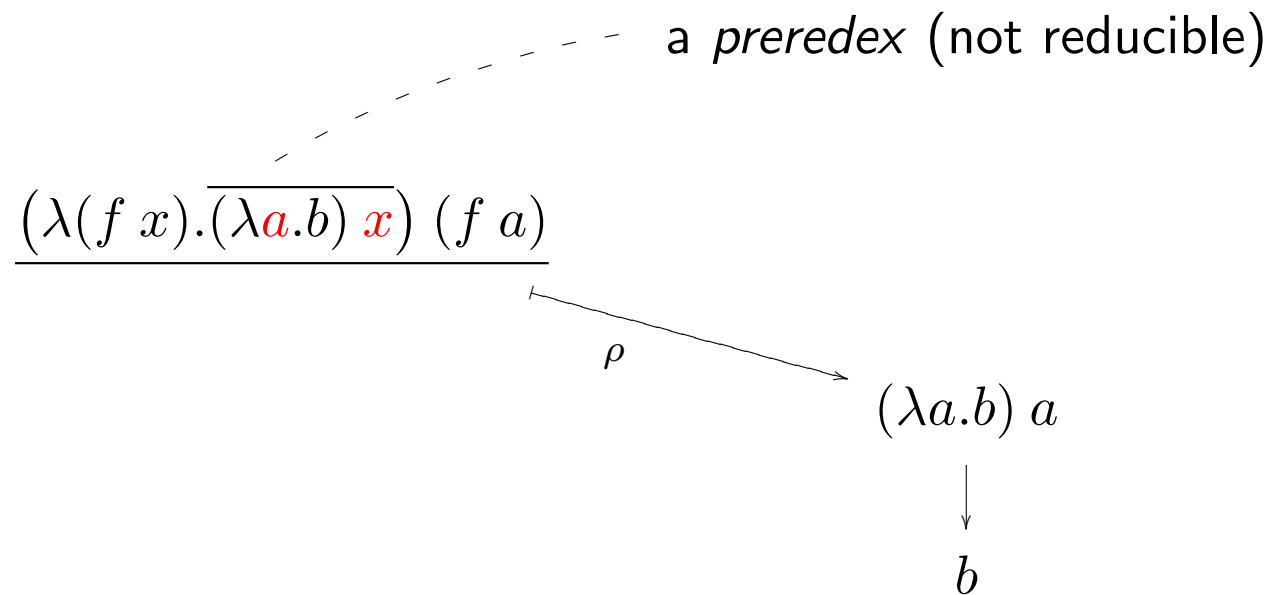
## About preredexes

$$\underline{(\lambda(f\ x).\overline{(\lambda a.b)\ x})\ (f\ a)}$$

## About preredexes

$\underline{(\lambda(f\ x).\overline{(\lambda a.b)\ x})\ (f\ a)}$  a *preredex* (not reducible)

## About preredexes



# Ensuring confluence

- Strategies
  - ▶ Call by value...
  - ▶ Suitable for operational semantics but not adapted for logics
- Restrictions on patterns [**van Oostrom 90**]
  - ▶ Algebraic and linear
  - ▶ More restrictive but stable by reduction

## About the expressiveness of the $\rho$ -calculus

- The  $\lambda$ -calculus is fully embedded in the  $\rho$ -calculus [Cirstea & Kirchner 98]
  - ▶  $\beta$ -reductions are faithfully mimicked
  - ▶ a  $\lambda$ -term  $\rho$ -reduces to  $\lambda$ -terms only
- Various aspects of rewriting can be represented [Cirstea & Kirchner 98]
  - ▶ Rewriting paths
  - ▶ Rewriting systems
  - ▶ Rewriting strategies
- Various object calculi can be encoded [Cirstea, Kirchner & Liquori 01]

# Contents

1. Untyped rewriting calculus
2. Type systems for programming
  - ▶ Properties and type inference
  - ▶ Typed encoding of term rewriting systems
3. Pure Pattern Type Systems
  - ▶ Strong normalization in  $\rho_{\rightarrow}$  and  $\rho P$
4. Using the  $\rho$ -calculus for deduction
  - ▶  $P^2TS$ -proof terms for deduction modulo
  - ▶ Generalized Natural Deduction

## A Simple Type System $\rho_1$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_{\Sigma} x : \sigma} \text{ (Var)} \qquad \frac{f : \sigma \in \Sigma}{\Gamma \vdash_{\Sigma} f : \sigma} \text{ (Const)}$$



## A Simple Type System $\rho_1$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_{\Sigma} x : \sigma} \text{ (Var)} \quad \frac{f : \sigma \in \Sigma}{\Gamma \vdash_{\Sigma} f : \sigma} \text{ (Const)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \rightarrow \tau \quad \Gamma \vdash_{\Sigma} B : \sigma}{\Gamma \vdash_{\Sigma} A B : \tau} \text{ (Appl)}$$

## A Simple Type System $\rho_1$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_{\Sigma} x : \sigma} \text{ (Var)} \quad \frac{f : \sigma \in \Sigma}{\Gamma \vdash_{\Sigma} f : \sigma} \text{ (Const)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \rightarrow \tau \quad \Gamma \vdash_{\Sigma} B : \sigma}{\Gamma \vdash_{\Sigma} A B : \tau} \text{ (Appl)}$$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : \sigma \quad \Gamma, \Delta \vdash_{\Sigma} A : \tau}{\Gamma \vdash_{\Sigma} \lambda(P : \Delta). A : \sigma \rightarrow \tau} \text{ (Abs)}$$

$$\text{Dom}(\Delta) = \mathcal{FV}(P)$$

## A Simple Type System $\rho_1$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_{\Sigma} x : \sigma} \text{ (Var)} \quad \frac{f : \sigma \in \Sigma}{\Gamma \vdash_{\Sigma} f : \sigma} \text{ (Const)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \rightarrow \tau \quad \Gamma \vdash_{\Sigma} B : \sigma}{\Gamma \vdash_{\Sigma} A B : \tau} \text{ (Appl)}$$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : \sigma \quad \Gamma, \Delta \vdash_{\Sigma} A : \tau}{\Gamma \vdash_{\Sigma} \lambda(P : \Delta). A : \sigma \rightarrow \tau} \text{ (Abs)}$$

$$\text{Dom}(\Delta) = \mathcal{FV}(P)$$

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \Gamma \vdash_{\Sigma} B : \sigma}{\Gamma \vdash_{\Sigma} A \wr B : \sigma} \text{ (Struct)}$$

# Polymorphic extensions

*à la Church*

*à la Curry*

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_{\Sigma} \lambda\alpha.A : \forall\alpha.\sigma} \text{ (Abs}\forall\text{)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma}{\Gamma \vdash_{\Sigma} A\tau : \sigma[\alpha := \tau]} \text{ (App}\forall\text{)}$$

# Polymorphic extensions

*à la Church*

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_{\Sigma} \lambda\alpha.A : \forall\alpha.\sigma} \text{ (Abs}\forall\text{)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma}{\Gamma \vdash_{\Sigma} A\tau : \sigma[\alpha := \tau]} \text{ (App}\forall\text{)}$$

*à la Curry*

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma} \text{ (Abs}\forall\text{)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma}{\Gamma \vdash_{\Sigma} A : \sigma[\alpha := \tau]} \text{ (App}\forall\text{)}$$

## Polymorphic extensions

*à la Church*

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_{\Sigma} \lambda\alpha. A : \forall\alpha.\sigma} \quad (Abs\forall)$$

$$\frac{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma}{\Gamma \vdash_{\Sigma} A\tau : \sigma[\alpha := \tau]} \quad (App\forall)$$

*à la Curry*

$$\frac{\Gamma \vdash_{\Sigma} A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma} \quad (Abs\forall)$$

$$\frac{\Gamma \vdash_{\Sigma} A : \forall\alpha.\sigma}{\Gamma \vdash_{\Sigma} A : \sigma[\alpha := \tau]} \quad (App\forall)$$

$$\forall (f:\sigma) \in \Sigma, \quad \sigma \equiv \forall\bar{\alpha}(\sigma_1 \rightarrow \dots \iota(\bar{\beta}))$$

where  $\bar{\beta} = \mathcal{BV}(\sigma)$

# Typing properties

## Well-typed matching

If  $P\theta \equiv A$ , then  $\forall x \in P, \quad \Gamma \vdash_{\Sigma} x : \sigma \Rightarrow \Gamma \vdash_{\Sigma} x\theta : \sigma$

## Subject Reduction [Cirstea, Liquori & Wack 03]

If  $\Gamma \vdash_{\Sigma} A : \sigma$  and  $A \mapsto_{\rho\delta} B$ , then  $\Gamma \vdash_{\Sigma} B : \sigma$

## Uniqueness [Cirstea, Liquori & Wack 03]

In systems *à la* Church, if  $\Gamma \vdash_{\Sigma} A : \sigma$  and  $\Gamma \vdash_{\Sigma} A : \tau$ , then  $\tau =_{\alpha} \sigma$

## Decidability [Liquori & Wack 04]

In systems *à la* Church,  $\left. \begin{array}{l} \text{(typechecking)} \quad \Gamma \vdash_{\Sigma} \mathcal{T} : \sigma ? \\ \text{(type reconstruction)} \quad \Gamma \vdash_{\Sigma} \mathcal{T} : ? \end{array} \right\}$  are decidable

In systems *à la* Curry, both are undecidable

# Type inference

- In systems *à la* Church, type inference is fully guided by syntax
- The type system *à la* Curry has to be restricted
  - ▶ The only legal types are **type-schemes**  $\forall \bar{\alpha}. \tau$  where  $\tau$  is a simple type
  - ▶ Polymorphism is restricted to a new construction  $[P \ll A]B$   
(similar to `let ... in`)
  - ▶ Inference works in the style of the **Damas-Milner** algorithm



## Normalization failure

$$\omega \triangleq \lambda x . x \quad x$$

$$\begin{aligned} \omega \quad \omega &\equiv (\lambda x . x \quad x) \quad \omega \\ &\xrightarrow{\rho} \omega \quad \omega \\ &\xrightarrow{\rho} \dots \end{aligned}$$

## Normalization failure

$$\Gamma = x : \alpha \rightarrow \alpha, \quad \omega \triangleq \lambda x . x \quad x$$

$$\omega \quad \omega \equiv (\lambda x . x \quad x) \quad \omega$$

$$\rightarrow_{\rho} \omega \quad \omega$$

$$\rightarrow_{\rho} \dots$$

## Normalization failure

$f : (\alpha \rightarrow \alpha) \rightarrow \alpha$  and  $\Gamma = x : \alpha \rightarrow \alpha$ ,  $\omega \triangleq \lambda(f\ x).x\ (f\ x)$

$$\omega\ (f\ \omega) \equiv (\lambda(f\ x).x\ (f\ x))\ (f\ \omega)$$

$$\rightarrow_{\rho} \omega\ (f\ \omega)$$

$$\rightarrow_{\rho} \dots$$

## Normalization failure (cont'd)

$f : (\alpha \rightarrow \alpha) \rightarrow \alpha$  and  $\Gamma = x : \alpha \rightarrow \alpha$ ,  $\omega \triangleq \lambda f x.x (f x)$

$$\frac{\Gamma \vdash_{\Sigma} f : (\alpha \rightarrow \alpha) \rightarrow \alpha \quad \Gamma \vdash_{\Sigma} x : \alpha \rightarrow \alpha}{\Gamma \vdash_{\Sigma} f x : \alpha} \quad \frac{\Gamma \vdash_{\Sigma} x : \alpha \rightarrow \alpha \quad \Gamma \vdash_{\Sigma} f x : \alpha}{\Gamma \vdash_{\Sigma} x (f x) : \alpha}$$

$$\vdash_{\Sigma} \omega (f \omega) : \alpha$$

## Encoding rewriting systems in the $\rho$ -calculus

Addition over Peano integers:  $\Sigma = \{0, S, rec, add\}$

$$plus \triangleq \lambda rec z . \left( \begin{array}{l} \lambda(add\ 0\ y) . y \\ \lambda(add(S\ x)\ y) . S((z\ (rec\ z))\ (add\ x\ y)) \end{array} \right)$$

## Encoding rewriting systems in the $\rho$ -calculus

Addition over Peano integers:  $\Sigma = \{0, S, rec, add\}$

$$plus \triangleq \lambda rec z . \left( \begin{array}{l} \lambda(add\ 0\ y) . y \\ \wr \lambda(add(S\ x)\ y) . S((z\ (rec\ z))\ (add\ x\ y)) \end{array} \right)$$

$(plus\ (rec\ plus))\ (add\ N\ M)$

$$\mapsto_{\rho\delta} (\lambda 0.M)\ N \wr (\lambda 0.\widetilde{M+1})\ \widetilde{N-1} \dots (\lambda 0.\widetilde{M+N})\ 0 \wr (\lambda(S\ x).\dots)\ 0$$

## Encoding rewriting systems in the $\rho$ -calculus

Addition over Peano integers:  $\Sigma = \{0, S, rec, add\}$

$$plus \triangleq \lambda rec z . \left( \begin{array}{l} \lambda(add\ 0\ y) . y \\ \wr \lambda(add(S\ x)\ y) . S((z\ (rec\ z))\ (add\ x\ y)) \end{array} \right)$$

$(plus\ (rec\ plus))\ (add\ N\ M)$

$$\mapsto_{\rho\delta} (\lambda 0.M)\ N \wr (\lambda 0.\widetilde{M+1})\ \widetilde{N-1} \dots (\lambda 0.\widetilde{M+N})\ 0 \wr (\lambda(S\ x).\dots)\ 0$$

$$\stackrel{?}{\mapsto} \widetilde{M+N}$$

## Detecting matching failures: the symbol $stk$

- The relation  $P \not\sqsubseteq A$  detects (some) definitive matching failures



## Detecting matching failures: the symbol $\text{stk}$

- The relation  $P \not\sqsubseteq A$  detects (some) definitive matching failures
- The relation  $\mapsto_{\text{stk}}$  treats matching failures uniformly:

$$(\lambda P:\Delta.A) B \mapsto_{\text{stk}} \text{stk} \quad \text{if } P \not\sqsubseteq B$$

$$\text{stk} \wr A \mapsto_{\text{stk}} A$$

$$A \wr \text{stk} \mapsto_{\text{stk}} A$$

$$\text{stk} A \mapsto_{\text{stk}} \text{stk}$$

## Detecting matching failures: the symbol $\text{stk}$

- The relation  $P \not\sqsubseteq A$  detects (some) definitive matching failures
- The relation  $\mapsto_{\text{stk}}$  treats matching failures uniformly:

$$(\lambda P:\Delta.A) B \mapsto_{\text{stk}} \text{stk} \quad \text{if } P \not\sqsubseteq B$$

$$\text{stk} \wr A \mapsto_{\text{stk}} A$$

$$A \wr \text{stk} \mapsto_{\text{stk}} A$$

$$\text{stk} A \mapsto_{\text{stk}} \text{stk}$$

- **Theorem [Cirstea, Liquori & Wack 03]** The reduction  $\mapsto_{\rho\delta}^{\text{stk}}$  is confluent

## Systematic encoding

- There exists a  $\rho$ -term *first* (using **stk**) such that

$$\begin{array}{l} (first\ A_1\ A_2\ \dots\ A_n)\ B \quad \mapsto_{\rho\delta}^{stk} \quad A_{i+1}\ B \\ \text{if} \quad \begin{array}{l} A_{i+1}\ B \quad \not\mapsto_{\rho\delta}^{stk} \quad \mathbf{stk} \\ \forall j \leq i, \ A_j\ B \quad \mapsto_{\rho\delta}^{stk} \quad \mathbf{stk} \end{array} \end{array}$$

## Systematic encoding

- There exists a  $\rho$ -term *first* (using **stk**) such that

$$\begin{array}{l}
 (first\ A_1\ A_2\ \dots\ A_n)\ B \quad \mapsto_{\rho}^{stk} \quad A_{i+1}\ B \\
 \text{if} \quad \begin{array}{l}
 A_{i+1}\ B \quad \not\mapsto_{\rho}^{stk} \quad \mathbf{stk} \\
 \forall j \leq i, \ A_j\ B \quad \mapsto_{\rho}^{stk} \quad \mathbf{stk}
 \end{array}
 \end{array}$$

- The Term Rewrite System  $\mathcal{R} = \{l_i \rightarrow r_i\}$  with signature  $\{a_j\}$  is encoded by:

$$\llbracket \mathcal{R} \rrbracket = \lambda(rec\ z) . first \left( \begin{array}{l}
 \lambda l_1 . z (rec\ z) r_1 \\
 \dots \\
 \lambda(a_1\ \bar{x}) . z (rec\ z) a_1(\overline{z (rec\ z) x}) \\
 \dots
 \end{array} \right)$$

## Properties of the encoding

### Theorem [Cirstea, Liquori & Wack 03]

This encoding is **sound** for left-linear TRS  
**complete** for convergent TRS  
**typable** if the TRS is well-typed

### Remark [Cirstea, Kirchner, Liquori & Wack 03]

Various strategies can be encoded

## Other cases of non termination under typing

- In CaML,  $\omega$  can be written

```
type t = F of (t -> t);;
```

```
let omega x = match x with (F y) -> y (F y);;
```

- In CIC, type constructors must fulfill a **positiveness** condition [Mendler 87]

## Logical inconsistency

- In this type system, the Curry-Howard isomorphism is not valid:

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : \sigma \quad \Gamma, \Delta \vdash_{\Sigma} A : \tau}{\Gamma \vdash_{\Sigma} \lambda P : \Delta . A : \sigma \rightarrow \tau} \text{ (Abs)}$$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} \sigma \quad \Gamma, \Delta \vdash_{\Sigma} \tau}{\Gamma \vdash_{\Sigma} \sigma \rightarrow \tau} \text{ (}\rightarrow I\text{)}$$

## Logical inconsistency

- In this type system, the Curry-Howard isomorphism is not valid:

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : \sigma \quad \Gamma, \Delta \vdash_{\Sigma} A : \tau}{\Gamma \vdash_{\Sigma} \lambda P : \Delta . A : \sigma \rightarrow \tau} \text{ (Abs)} \qquad \frac{\Gamma, \Delta \vdash_{\Sigma} \sigma \quad \Gamma, \Delta \vdash_{\Sigma} \tau}{\Gamma \vdash_{\Sigma} \sigma \rightarrow \tau} (\rightarrow I)$$

- How to fix it ?

$$\frac{\Gamma, X_i : \sigma_i \vdash_{\Sigma} A : \tau}{\Gamma \vdash_{\Sigma} \lambda P . A : (\bigwedge \sigma_i) \rightarrow \tau} \text{ (Abs)} \quad , \quad \mathcal{FV}(P) = \{X_i\}$$

But how to type applications ?



# Contents

1. Untyped rewriting calculus
2. Type systems for programming
  - ▶ Properties and type inference
  - ▶ Typed encoding of term rewriting systems
3. Pure Pattern Type Systems
  - ▶ Strong normalization in  $\rho_{\rightarrow}$  and  $\rho P$
4. Using the  $\rho$ -calculus for deduction
  - ▶  $P^2TS$ -proof terms for deduction modulo
  - ▶ Generalized Natural Deduction

## Dependent type discipline in $P^2TS$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} B : C \quad \Gamma \vdash_{\Sigma} \Pi P:\Delta. C : s}{\Gamma \vdash_{\Sigma} \lambda P:\Delta. B : \Pi P:\Delta. C} \text{ (Abs)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \Pi P:\Delta. C \quad \Gamma \vdash_{\Sigma} [P \ll_{\Delta} B]C : s}{\Gamma \vdash_{\Sigma} A B : [P \ll_{\Delta} B]C} \text{ (Appl)}$$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : A \quad \Gamma \vdash_{\Sigma} B : A \quad \Gamma, \Delta \vdash_{\Sigma} A : s_1 \quad \Gamma, \Delta \vdash_{\Sigma} C : s_2}{\Gamma \vdash_{\Sigma} [P \ll_{\Delta} B]C : s_2} \text{ (Match)}$$

## Dependent type discipline in $P^2TS$

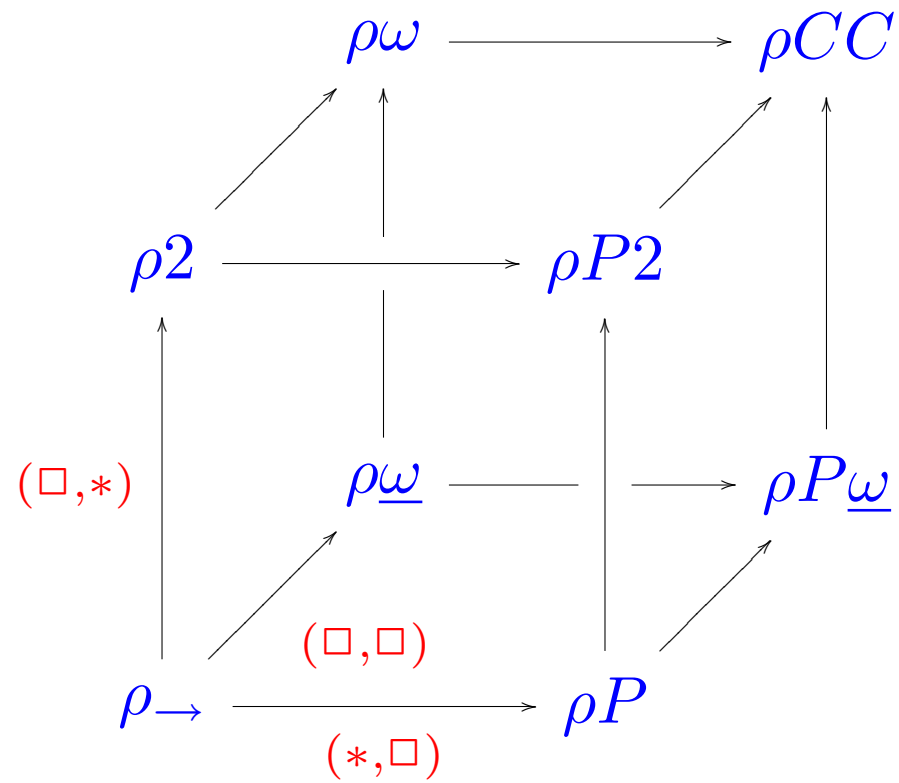
$$\frac{\Gamma, \Delta \vdash_{\Sigma} B : C \quad \Gamma \vdash_{\Sigma} \Pi P:\Delta. C : s}{\Gamma \vdash_{\Sigma} \lambda P:\Delta. B : \Pi P:\Delta. C} \text{ (Abs)}$$

$$\frac{\Gamma \vdash_{\Sigma} A : \Pi P:\Delta. C \quad \Gamma \vdash_{\Sigma} [P \ll_{\Delta} B]C : s}{\Gamma \vdash_{\Sigma} A B : [P \ll_{\Delta} B]C} \text{ (Appl)}$$

$$\frac{\Gamma, \Delta \vdash_{\Sigma} P : A \quad \Gamma \vdash_{\Sigma} B : A \quad \Gamma, \Delta \vdash_{\Sigma} A : s_1 \quad \Gamma, \Delta \vdash_{\Sigma} C : s_2}{\Gamma \vdash_{\Sigma} [P \ll_{\Delta} B]C : s_2} \text{ (Match)}$$

With  $\Delta = \{x:\iota, l:list\}$  we have  $\vdash_{\Sigma} \lambda(\text{cons } x \ l):\Delta. x : \Pi(\text{cons } x \ l):\Delta. \iota$

# The $\rho$ -cube



# Typing properties

[Barthe, Cirstea, Kirchner & Liquori 03]

**Subject reduction:**  $\Gamma \vdash_{\Sigma} A : C \wedge A \mapsto_{\rho\delta} B \Rightarrow \Gamma \vdash_{\Sigma} B : C$

**Correctness:**  $\Gamma \vdash_{\Sigma} A : B \Rightarrow \Gamma \vdash_{\Sigma} B : s \vee B \equiv s$

**Consistency:**  $A \in \text{Nf}(\rho\delta) \Rightarrow \not\vdash_{\Sigma} A : \perp \quad (\triangleq \forall x: * .x)$

**Uniqueness:**  $\Gamma \vdash_{\Sigma} A : B \wedge \Gamma \vdash_{\Sigma} A : B' \Rightarrow B \equiv_{\rho\delta} B'$

**Conservativity:**  $\Gamma \vdash_{PTS} A : B \Leftrightarrow \Gamma \vdash_{P^2TS} A : B$

## Typing is more restrictive

Here, with  $\Delta \equiv \{x : \Pi z:\alpha.\alpha\}$ :

$$\vdash_{\Sigma} \omega \triangleq \lambda(f x):\Delta.x (f x) : \Pi(f x):\Delta.\alpha$$

And:

$$\vdash_{\Sigma} f : \Pi(y : \Pi z:\alpha.\alpha).\alpha$$

But to type  $f \omega$  the pattern  $y$  and the argument  $\omega$  must have a common type  $\sigma$

## Strong normalization : sketch of the proof

### Theorem [Wack 04]:

In  $\rho_{\rightarrow}$  and  $\rho_P$ , if  $\Gamma \vdash_{\Sigma} A : C$  then  $A$  and  $C$  are SN

## Strong normalization : sketch of the proof

### Theorem [Wack 04]:

In  $\rho_{\rightarrow}$  and  $\rho_P$ , if  $\Gamma \vdash_{\Sigma} A : C$  then  $A$  and  $C$  are SN

1. Find a translation  $\llbracket \cdot \rrbracket : P^2TS \rightarrow \lambda\omega$  correct w.r.t. reductions

If  $A \mapsto_{\rho\delta} B$ , then  $\llbracket A \rrbracket \xrightarrow{\beta} \llbracket B \rrbracket$  in at least one step



## Strong normalization : sketch of the proof

### Theorem [Wack 04]:

In  $\rho_{\rightarrow}$  and  $\rho_P$ , if  $\Gamma \vdash_{\Sigma} A : C$  then  $A$  and  $C$  are SN

1. Find a translation  $\llbracket \cdot \rrbracket : P^2TS \rightarrow \lambda\omega$  correct w.r.t. reductions

If  $A \mapsto_{\rho\delta} B$ , then  $\llbracket A \rrbracket \xrightarrow{\beta} \llbracket B \rrbracket$  in at least one step

2. Typability of the translated terms

$$\Sigma, \Gamma \vdash_{\Sigma} A : C \quad \Rightarrow \quad \exists \tau, \quad \llbracket \Gamma \rrbracket \vdash_{\lambda\omega} \llbracket A \rrbracket : \tau$$

## Strong normalization : sketch of the proof

### Theorem [Wack 04]:

In  $\rho_{\rightarrow}$  and  $\rho P$ , if  $\Gamma \vdash_{\Sigma} A : C$  then  $A$  and  $C$  are SN

1. Find a translation  $\llbracket \cdot \rrbracket : P^2TS \rightarrow \lambda\omega$  correct w.r.t. reductions

If  $A \mapsto_{\rho\delta} B$ , then  $\llbracket A \rrbracket \xrightarrow{\beta} \llbracket B \rrbracket$  in at least one step

2. Typability of the translated terms

$$\Sigma, \Gamma \vdash_{\Sigma} A : C \quad \Rightarrow \quad \exists \tau, \quad \llbracket \Gamma \rrbracket \vdash_{\lambda\omega} \llbracket A \rrbracket : \tau$$

3. Usual techniques can be adapted to reduce SN in  $\rho P$  to SN in  $\rho_{\rightarrow}$

## Correctness of reductions

- $\llbracket (\lambda(f\ x).x) (f\ a) \rrbracket = (\lambda u.(u(\lambda x.x))) ((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)) \mapsto_{\beta} \lambda v.v = \llbracket a \rrbracket$

## Correctness of reductions

- $\llbracket (\lambda(f\ x).x) (f\ a) \rrbracket = (\lambda u.(u(\lambda x.x))) ((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)) \mapsto_{\beta} \lambda v.v = \llbracket a \rrbracket$
- The  $\rho$ -term  $(\lambda y.(\lambda(f\ x).x) y) (f\ a)$  features a preredex

## Correctness of reductions

- $\llbracket (\lambda(f\ x).x) (f\ a) \rrbracket = (\lambda u.(u(\lambda x.x))) ((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)) \mapsto_{\beta} \lambda v.v = \llbracket a \rrbracket$
- The  $\rho$ -term  $(\lambda y.(\lambda(f\ x).x) y) (f\ a)$  features a preredex
- Thus, the reductions of the  $\lambda$ -term  $\llbracket (\lambda y.(\lambda(f\ x).x) y) (f\ a) \rrbracket$  must mimick first an external  $\rho$ -reduction

## Correctness of reductions

- $\llbracket (\lambda(f\ x).x) (f\ a) \rrbracket = (\lambda u.(u(\lambda x.x))) ((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)) \mapsto_{\beta} \lambda v.v = \llbracket a \rrbracket$
- The  $\rho$ -term  $(\lambda y.(\lambda(f\ x).x) y) (f\ a)$  features a preredex
- Thus, the reductions of the  $\lambda$ -term  $\llbracket (\lambda y.(\lambda(f\ x).x) y) (f\ a) \rrbracket$  must mimick first an external  $\rho$ -reduction
- Remark: a term produced by the translation may have additional reductions

## The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} \llbracket f B \rrbracket \quad : \quad (\sigma \rightarrow \beta) \rightarrow \beta$$

$$\vdash_{\lambda\omega} \llbracket \lambda(f x).A \rrbracket \quad : \quad ((\sigma \rightarrow \tau) \rightarrow \gamma) \rightarrow \gamma$$

where  $\tau$  is the type of  $\llbracket A \rrbracket$

## The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} \llbracket f B \rrbracket \quad : \quad (\sigma \rightarrow \beta) \rightarrow \beta$$

$$\vdash_{\lambda\omega} \llbracket \lambda(f x).A \rrbracket \quad : \quad ((\sigma \rightarrow \tau) \rightarrow \gamma) \rightarrow \gamma$$

where  $\tau$  is the type of  $\llbracket A \rrbracket$

$$(\sigma \rightarrow \tau) \rightarrow \gamma \quad = \quad (\sigma \rightarrow \beta) \rightarrow \beta \quad \text{thus} \quad \tau = \beta = \gamma$$



## The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} \llbracket f B \rrbracket \quad : \quad (\sigma \rightarrow \beta) \rightarrow \beta$$

$$\vdash_{\lambda\omega} \llbracket \lambda(f x).A \rrbracket \quad : \quad ((\sigma \rightarrow \tau) \rightarrow \gamma) \rightarrow \gamma$$

where  $\tau$  is the type of  $\llbracket A \rrbracket$

$$(\sigma \rightarrow \tau) \rightarrow \gamma \quad = \quad (\sigma \rightarrow \beta) \rightarrow \beta \quad \text{thus} \quad \tau = \beta = \gamma$$

- The actual translation features **terms depending on types**

$$\llbracket f B \rrbracket \quad : \quad \forall \beta. (\sigma \rightarrow \beta \rightarrow \beta)$$

$$\llbracket \lambda(f x).A \rrbracket \quad : \quad \forall \beta. (\sigma \rightarrow \beta \rightarrow \beta) \rightarrow \tau$$

# The type of a translated variable

- Naive translation

$$x : \Pi y:\iota.\iota \vdash_{\Sigma} x \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . y \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . a \quad : \quad \Pi y:\iota . \iota$$

# The type of a translated variable

- Naive translation

$$x : \Pi y:\iota.\iota \vdash_{\Sigma} x \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . y \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . a \quad : \quad \Pi y:\iota . \iota$$

$$\Gamma \vdash_{\lambda\omega} \lambda y:\beta_y.y \quad : \quad \beta_y \rightarrow \beta_y$$

$$\Gamma \vdash_{\lambda\omega} \lambda y:\beta_y.[[a]] \quad : \quad \beta_y \rightarrow \forall\alpha.(\alpha \rightarrow \alpha)$$

# The type of a translated variable

- Naive translation

$$x : \Pi y:\iota.\iota \vdash_{\Sigma} x \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . y \quad : \quad \Pi y:\iota . \iota$$

$$\vdash_{\Sigma} \lambda y:\iota . a \quad : \quad \Pi y:\iota . \iota$$

$$\Gamma \vdash_{\lambda\omega} \lambda y:\beta_y.y \quad : \quad \beta_y \rightarrow \beta_y$$

$$\Gamma \vdash_{\lambda\omega} \lambda y:\beta_y. \llbracket a \rrbracket \quad : \quad \beta_y \rightarrow \forall \alpha. (\alpha \rightarrow \alpha)$$

- Use of **types depending on types**

$$\beta_x : * \rightarrow *, \beta_y : * \vdash_{\lambda\omega} \llbracket x \rrbracket : \beta_y \rightarrow \beta_x \beta_y$$

# Contents

1. Untyped rewriting calculus
2. Type systems for programming
  - ▶ Properties and type inference
  - ▶ Typed encoding of term rewriting systems
3. Pure Pattern Type Systems
  - ▶ Strong normalization in  $\rho_{\rightarrow}$  and  $\rho P$
4. Using the  $\rho$ -calculus for deduction
  - ▶  $P^2TS$ -proof terms for deduction modulo
  - ▶ Generalized Natural Deduction

## A linear representation of NDM proofs

- A proof in Natural Deduction Modulo: the congruence states that  $e$  is the neutral element of a group:  $e * x \cong x$

$$\begin{array}{c}
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} \forall y.(y * e' = y)} \quad (Ax) \\
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} \forall y.(y * e' = y)} \quad (\forall E) \\
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} e * e' = e} \quad (\cong) \quad \text{with } e * e' \cong e' \\
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} e' = e} \quad (\Rightarrow I) \\
 \frac{}{\vdash_{\cong} \forall y.(y * e' = y) \Rightarrow e' = e}
 \end{array}$$

## A linear representation of NDM proofs

- A proof in Natural Deduction Modulo: the congruence states that  $e$  is the neutral element of a group:  $e * x \cong x$

$$\begin{array}{c}
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} \forall y.(y * e' = y)} \quad (Ax) \\
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} e * e' = e} \quad (\forall E) \\
 \frac{}{\forall y.(y * e' = y) \vdash_{\cong} e' = e} \quad (\cong) \quad \text{with } e * e' \cong e' \\
 \frac{}{\vdash_{\cong} \forall y.(y * e' = y) \Rightarrow e' = e} \quad (\Rightarrow I)
 \end{array}$$

- $\lambda$ -calculus is sufficient to write witnesses [**Dowek & Werner 03**]

$\lambda\alpha.(\alpha e)$

- ▶ the witness is short and focuses on reasoning
- ▶ but proof reconstruction can be tedious

## A more explicit representation

- Using  $P^2TS$ , conversions can be accounted for by dedicated constructs in the style of Leibniz's equality :

$$\vdash_{\Sigma} \text{Rew } \phi t (\lambda l.r) \pi : \phi((\lambda l.r) t)$$



## A more explicit representation

- Using  $P^2TS$ , conversions can be accounted for by dedicated constructs in the style of Leibniz's equality :

$$\vdash_{\Sigma} \text{Rew } \phi t (\lambda l.r) \pi : \phi((\lambda l.r) t)$$

- The new proof term for our example is

$$\lambda \alpha. \left( \text{Rew } (\lambda y.(y=e)) (e*e') (\lambda(e*x).x) (\alpha e) \right)$$

## A more explicit representation

- Using  $P^2TS$ , conversions can be accounted for by dedicated constructs in the style of Leibniz's equality :

$$\vdash_{\Sigma} \text{Rew } \phi t (\lambda l.r) \pi : \phi((\lambda l.r) t)$$

- The new proof term for our example is

$$\lambda \alpha. \left( \text{Rew } (\lambda y.(y=e)) (e*e') (\lambda(e*x).x) (\alpha e) \right)$$

- **Proposition:** For conversion on propositions, application of rewrite rules at top-level is sufficient

# A Curry-Howard-de Bruijn correspondence

**Theorem [Wack 05]:**

✓ Full proof representation

# A Curry-Howard-de Bruijn correspondence

## Theorem [Wack 05]:

- ✓ Full proof representation
- ✗ Incomplete proof reduction
  - ✓ Every redex represents a cut
  - ✗ But some cuts are obfuscated by conversion rules

$$\begin{array}{c} (\Rightarrow I) \frac{p \vdash_{\cong} p}{\vdash_{\cong} p \Rightarrow p} \quad \vdots \\ (\cong) \frac{\vdash_{\cong} p \Rightarrow p}{\vdash_{\cong} q \Rightarrow p} \quad \frac{\vdash_{\cong} q}{\vdash_{\cong} q} \\ (\Rightarrow E) \frac{\vdash_{\cong} q \Rightarrow p \quad \vdash_{\cong} q}{\vdash_{\cong} p} \end{array}$$

? Conjecture : additional *fold-unfold* reduction rules allow to reduce every cut

## Main benefits

- Proof checking reduces to type checking and **matching**
- Construction of the **conversion steps can be delegated** to an efficient rewriting-based software
- A  **$\lambda$ -proof term** can always be extracted from a  $\rho$ -proof term
- The set of used **rewrite rules** can also be extracted

## A simple proof in Natural Deduction...

The theory  $\mathcal{T}$  contains at least  $\left\{ \begin{array}{l} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \perp) \end{array} \right.$

## A simple proof in Natural Deduction...

The theory  $\mathcal{T}$  contains at least  $\left\{ \begin{array}{l} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \perp) \end{array} \right.$

$$\mathcal{T} \vdash \emptyset \subseteq A$$

## A simple proof in Natural Deduction...

The theory  $\mathcal{T}$  contains at least  $\left\{ \begin{array}{l} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \perp) \end{array} \right.$

$$\begin{array}{c}
 (Ax) \frac{}{\mathcal{T}, x \in \emptyset \vdash \forall x(x \in \emptyset \Rightarrow \perp)} \\
 (\forall E) \frac{}{\mathcal{T}, x \in \emptyset \vdash x \in \emptyset \Rightarrow \perp} \\
 (\Rightarrow E) \frac{}{\mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A) \Rightarrow \emptyset \subseteq A} \\
 \\
 (\perp E) \frac{\mathcal{T}, x \in \emptyset \vdash \perp}{\mathcal{T}, x \in \emptyset \vdash x \in A} \\
 (\Rightarrow I) \frac{}{\mathcal{T} \vdash x \in \emptyset \Rightarrow x \in A} \\
 (\forall I) \frac{}{\mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A)} \\
 \mathcal{T} \vdash \dots \\
 (Ax) \frac{}{\mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A) \Rightarrow \emptyset \subseteq A} \\
 (\Rightarrow E) \frac{}{\mathcal{T} \vdash \emptyset \subseteq A}
 \end{array}$$



**... shorter in deduction modulo**

In NDM the context is empty and  $\mathcal{R} = \left\{ \begin{array}{l} X \subseteq Y \rightarrow \forall x(x \in X \Rightarrow x \in Y) \\ x \in \emptyset \rightarrow \perp \end{array} \right.$

## ... shorter in deduction modulo

In NDM the context is empty and  $\mathcal{R} = \left\{ \begin{array}{l} X \subseteq Y \rightarrow \forall x(x \in X \Rightarrow x \in Y) \\ x \in \emptyset \rightarrow \perp \end{array} \right.$

$$\begin{array}{l}
 (\text{Ax}) \frac{}{x \in \emptyset \vdash_{\cong} \perp} \quad x \in \emptyset \cong \perp \\
 (\perp E) \frac{}{x \in \emptyset \vdash_{\cong} x \in A} \\
 (\Rightarrow I) \frac{}{\vdash_{\cong} x \in \emptyset \Rightarrow x \in A} \\
 (\forall I) \frac{}{\vdash_{\cong} \emptyset \subseteq A} \quad \emptyset \subseteq A \cong \dots
 \end{array}$$

## ... shorter in deduction modulo

In NDM the context is empty and  $\mathcal{R} = \left\{ \begin{array}{l} X \subseteq Y \rightarrow \forall x(x \in X \Rightarrow x \in Y) \\ x \in \emptyset \rightarrow \perp \end{array} \right.$

$$\begin{array}{l}
 (\text{Ax}) \frac{}{x \in \emptyset \vdash_{\cong} \perp} \quad x \in \emptyset \cong \perp \\
 (\perp E) \frac{}{x \in \emptyset \vdash_{\cong} x \in A} \\
 (\Rightarrow I) \frac{}{\vdash_{\cong} x \in \emptyset \Rightarrow x \in A} \\
 (\forall I) \frac{}{\vdash_{\cong} \emptyset \subseteq A} \quad \emptyset \subseteq A \cong \dots
 \end{array}$$

The proof is shorter but not very informative

## A generalization of Natural Deduction

We consider some new rules about predicate symbols:

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y} \quad x \notin \mathcal{FV}(\Gamma) \qquad (\emptyset E) \frac{\Gamma \vdash x \in \emptyset}{\Gamma \vdash \phi}$$

## A generalization of Natural Deduction

We consider some new rules about predicate symbols:

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y} \quad x \notin \mathcal{FV}(\Gamma) \qquad (\emptyset E) \frac{\Gamma \vdash x \in \emptyset}{\Gamma \vdash \phi}$$

$$\vdash \emptyset \subseteq A$$

## A generalization of Natural Deduction

We consider some new rules about predicate symbols:

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y} \quad x \notin \mathcal{FV}(\Gamma)$$

$$(\emptyset E) \frac{\Gamma \vdash x \in \emptyset}{\Gamma \vdash \phi}$$

$$(\subseteq I) \frac{x \in \emptyset \vdash x \in A}{\vdash \emptyset \subseteq A}$$

## A generalization of Natural Deduction

We consider some new rules about predicate symbols:

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y} \quad x \notin \mathcal{FV}(\Gamma)$$

$$(\emptyset E) \frac{\Gamma \vdash x \in \emptyset}{\Gamma \vdash \phi}$$

$$\begin{array}{l} (Ax) \frac{}{x \in \emptyset \vdash x \in \emptyset} \\ (\emptyset E) \frac{}{x \in \emptyset \vdash x \in A} \\ (\subseteq I) \frac{}{\vdash \emptyset \subseteq A} \end{array}$$

## A generalization of Natural Deduction

We consider some new rules about predicate symbols:

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y} \quad x \notin \mathcal{FV}(\Gamma) \qquad (\emptyset E) \frac{\Gamma \vdash x \in \emptyset}{\Gamma \vdash \phi}$$

$$\begin{array}{l} (Ax) \frac{}{x \in \emptyset \vdash x \in \emptyset} \\ (\emptyset E) \frac{}{x \in \emptyset \vdash x \in A} \\ (\subseteq I) \frac{}{\vdash \emptyset \subseteq A} \end{array}$$

The proof is even shorter than in NDM and bears some resemblance with an “old-school” mathematic style



## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

Example:  $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$  gives

## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

Example:  $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$  gives

$$\frac{\frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash x \in X \Rightarrow x \in Y}}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}$$

## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

Example:  $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$  gives

$$\frac{\frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash x \in X \Rightarrow x \in Y}}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}$$

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y}$$

## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

Example:  $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$  gives

$$\frac{\frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash x \in X \Rightarrow x \in Y}}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}$$

$$\frac{\frac{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}{\Gamma \vdash t \in X \Rightarrow t \in Y} \quad \Gamma \vdash t \in X}{\Gamma \vdash t \in Y}$$

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y}$$

## Systematic generation of the new inference rules

For each *defined* predicate  $P$  (i.e. there is a rewrite rule  $P \rightarrow \phi$ ):

- decompose  $\phi$  along the connectives  $\wedge$  and  $\Rightarrow$  and  $\forall$
- gather all the assumptions and side conditions to build a new rule

Example:  $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$  gives

$$\frac{\frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash x \in X \Rightarrow x \in Y}}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}$$

$$\frac{\frac{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y)}{\Gamma \vdash t \in X \Rightarrow t \in Y} \quad \Gamma \vdash t \in X}{\Gamma \vdash t \in Y}$$

$$(\subseteq I) \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y}$$

$$(\subseteq E) \frac{\Gamma \vdash X \subseteq Y \quad \Gamma \vdash t \in X}{\Gamma \vdash t \in Y}$$

## Conservativity *w.r.t* first-order logic

- **Theorem:** Every defined predicate is provably equivalent to its definition
- Thus, a GND system is correct and complete if and only if the corresponding NDM system is correct and complete

## Cut elimination

A new notion of cut appears for each defined predicate:

$$(\subseteq E) \frac{\frac{\frac{\vdots^{\mathcal{D}_2}}{\Gamma \vdash t \in X}}{\Gamma, x \in X \vdash x \in Y} \quad (\subseteq I) \frac{\frac{\vdots^{\mathcal{D}_1}}{\Gamma, x \in X \vdash x \in Y}}{\Gamma \vdash X \subseteq Y} \quad (x \notin \mathcal{FV}(\Gamma))}{\Gamma \vdash t \in Y}}$$



## Cut elimination

A new notion of cut appears for each defined predicate:

$$(\subseteq E) \frac{\frac{\frac{\vdots^{\mathcal{D}_2}}{\Gamma \vdash t \in X}}{\Gamma, x \in X \vdash x \in Y} \quad (\subseteq I) \quad \frac{\frac{\vdots^{\mathcal{D}_1}}{\Gamma \vdash X \subseteq Y}}{(x \notin \mathcal{FV}(\Gamma))}}{\Gamma \vdash t \in Y}}$$

reduces to

$$\frac{\frac{\frac{\vdots^{\mathcal{D}_2}}{\Gamma \vdash t \in X} \quad \dots \quad \frac{\vdots^{\mathcal{D}_2}}{\Gamma \vdash X \subseteq Y}}{\Gamma \vdash X \subseteq Y} \quad (\subseteq I)}{\Gamma \vdash t \in Y}$$

## Cut elimination

A new notion of cut appears for each defined predicate:

$$(\subseteq E) \frac{\frac{\vdots^{\mathcal{D}_2}}{\Gamma \vdash t \in X} \quad (\subseteq I) \frac{\frac{\vdots^{\mathcal{D}_1}}{\Gamma, x \in X \vdash x \in Y} \quad (x \notin \mathcal{FV}(\Gamma))}{\Gamma \vdash X \subseteq Y}}{\Gamma \vdash t \in Y}$$

reduces to

$$\frac{\frac{\vdots^{\mathcal{D}_2} \quad \dots \quad \vdots^{\mathcal{D}_2}}{\vdots^{\mathcal{D}_1}}}{\Gamma \vdash t \in Y}$$

**Theorem:** Cut elimination holds whenever it holds in the corresponding NDM system

# Proof terms

Definition of proof terms for Generalized Natural Deduction

- Add *ad-hoc* constructions in the language
- Use the  $\lambda$ -abstraction and store multiple assumptions and witnesses in patterns

## Proof terms

Definition of proof terms for Generalized Natural Deduction

- Add *ad-hoc* constructions in the language
- Use the  $\lambda$ -abstraction and store multiple assumptions and witnesses in patterns

$$(\subseteq I) \frac{\Gamma, \alpha : x \in X \vdash \pi : x \in Y}{\Gamma \vdash \lambda(\subseteq x \alpha).\pi : X \subseteq Y}$$

$$(\subseteq E) \frac{\Gamma \vdash \pi : X \subseteq Y \quad \Gamma \vdash \pi' : t \in X}{\Gamma \vdash \pi(\subseteq t \pi') : t \in Y}$$

## Proof terms

Definition of proof terms for Generalized Natural Deduction

- Add *ad-hoc* constructions in the language
- Use the  $\lambda$ -abstraction and store multiple assumptions and witnesses in patterns

$$\begin{array}{c} (\subseteq I) \frac{\Gamma, \alpha : x \in X \vdash \pi : x \in Y}{\Gamma \vdash \lambda(\subseteq x \alpha).\pi : X \subseteq Y} \quad (\subseteq E) \frac{\Gamma \vdash \pi : X \subseteq Y \quad \Gamma \vdash \pi' : t \in X}{\Gamma \vdash \pi (\subseteq t \pi') : t \in Y} \end{array}$$

The reduction  $(\lambda(\subseteq x \alpha).\pi) (\subseteq t \pi') \mapsto \pi[x := t, \alpha := \pi']$  models cut elimination

## Proof terms

Definition of proof terms for Generalized Natural Deduction

- Add *ad-hoc* constructions in the language
- Use the  $\lambda$ -abstraction and store multiple assumptions and witnesses in patterns

$$\begin{array}{c} (\subseteq I) \frac{\Gamma, \alpha : x \in X \vdash \pi : x \in Y}{\Gamma \vdash \lambda(\subseteq x \alpha).\pi : X \subseteq Y} \quad (\subseteq E) \frac{\Gamma \vdash \pi : X \subseteq Y \quad \Gamma \vdash \pi' : t \in X}{\Gamma \vdash \pi(\subseteq t \pi') : t \in Y} \end{array}$$

The reduction  $(\lambda(\subseteq x \alpha).\pi) (\subseteq t \pi') \mapsto \pi[x := t, \alpha := \pi']$  models cut elimination

► A collection of new type systems for the  $\rho$ -calculus, to be studied

# Contributions

- Types for programming
  - ▶ Properties and applications of these systems
  - ▶ Type inference
- $P^2TS$ 
  - ▶ Detailed study of the usual properties
  - ▶ Strong normalization in  $\rho_{\rightarrow}$  and  $\rho P$
- Rewriting calculus and deduction
  - ▶ Rich proof terms for deduction modulo
  - ▶ A new way of embedding domain-specific information in the logic

# Perspectives

- Types
  - ▶ Strong normalization in the remaining of the  $\rho$ -cube
  - ▶ Conjunction types for structures
  - ▶ Generalized Natural Deduction seen as a collection of type systems



# Perspectives

- Types
  - ▶ Strong normalization in the remaining of the  $\rho$ -cube
  - ▶ Conjunction types for structures
  - ▶ Generalized Natural Deduction seen as a collection of type systems
- Generalized Natural Deduction
  - ▶ Further decomposition of the propositions in the generation of new rules
  - ▶ Tests on broader classes of rewrite rules

# Perspectives

- Types
  - ▶ Strong normalization in the remaining of the  $\rho$ -cube
  - ▶ Conjunction types for structures
  - ▶ Generalized Natural Deduction seen as a collection of type systems
- Generalized Natural Deduction
  - ▶ Further decomposition of the propositions in the generation of new rules
  - ▶ Tests on broader classes of rewrite rules
- Implementation of proof assistants
  - ▶ based on Natural Deduction Modulo, using  $\rho$ -proof terms
  - ▶ based on Generalized Natural Deduction

**Thanks for your attention**

## Deduction modulo

Let  $\mathcal{R}$  be a rewriting system which rewrites:

- terms to terms (e.g.  $0 + x \rightarrow x$ )
- atomic propositions to propositions (e.g.  $x * y = 0 \rightarrow x = 0 \vee y = 0$ )

## Deduction modulo

Let  $\mathcal{R}$  be a rewriting system which rewrites:

- terms to terms (e.g.  $0 + x \rightarrow x$ )
- atomic propositions to propositions (e.g.  $x * y = 0 \rightarrow x = 0 \vee y = 0$ )

Let  $\cong$  be the congruence closure of  $\rightarrow_{\mathcal{R}}$

## Deduction modulo

Let  $\mathcal{R}$  be a rewriting system which rewrites:

- terms to terms (e.g.  $0 + x \rightarrow x$ )
- atomic propositions to propositions (e.g.  $x * y = 0 \rightarrow x = 0 \vee y = 0$ )

Let  $\cong$  be the congruence closure of  $\rightarrow_{\mathcal{R}}$

Every deduction rule is considered modulo  $\cong$  :

$$(\Rightarrow E) \frac{\Gamma \vdash_{\cong} \vartheta \quad \Gamma \vdash_{\cong} \phi}{\Gamma \vdash_{\cong} \psi} \quad \vartheta \cong \phi \Rightarrow \psi$$

## Deduction modulo

Let  $\mathcal{R}$  be a rewriting system which rewrites:

- terms to terms (e.g.  $0 + x \rightarrow x$ )
- atomic propositions to propositions (e.g.  $x * y = 0 \rightarrow x = 0 \vee y = 0$ )

Let  $\cong$  be the congruence closure of  $\rightarrow_{\mathcal{R}}$

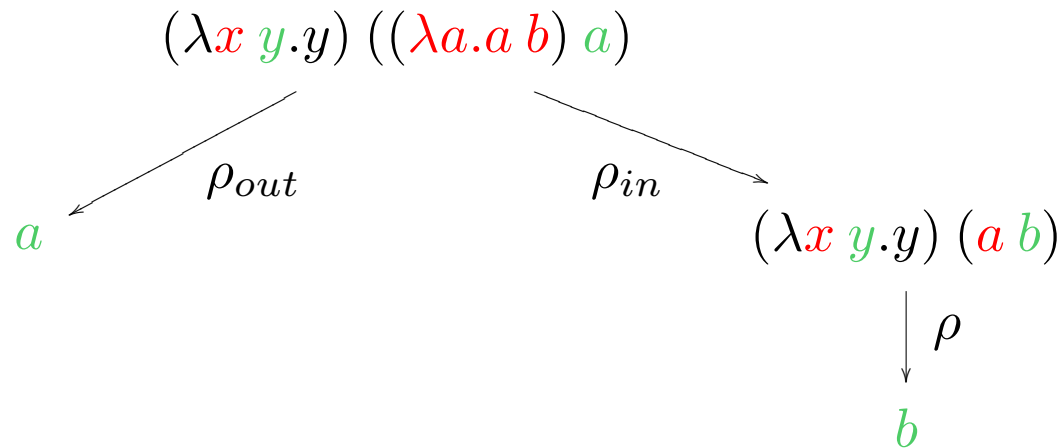
Every deduction rule is considered modulo  $\cong$  :

$$(\Rightarrow E) \frac{\Gamma \vdash_{\cong} \vartheta \quad \Gamma \vdash_{\cong} \phi}{\Gamma \vdash_{\cong} \psi} \quad \vartheta \cong \phi \Rightarrow \psi$$

A large part of the theory can (or should) be represented in  $\mathcal{R}$

## (Non-)Confluence of the $\rho$ -calculus

- *Active* variables are troublesome



- This kind of pattern (as well as abstractions) should be treated with higher-order matching



## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

- ▶  $C$  such that  $C \mapsto_{\rho\delta} \lambda y.(\lambda(dx x).e)(dy(C y))$
- ▶  $A$  such that  $A \mapsto_{\rho\delta} C A$

$A$

## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

►  $C$  such that  $C \mapsto_{\rho\delta} \lambda y.(\lambda(dx x).e)(dy(C y))$

►  $A$  such that  $A \mapsto_{\rho\delta} C A$

$$A \twoheadrightarrow C A$$

## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

►  $C$  such that  $C \mapsto_{\rho\delta} \lambda y.(\lambda(dx x).e)(dy(C y))$

►  $A$  such that  $A \mapsto_{\rho\delta} C A$

$$A \twoheadrightarrow C A \longrightarrow (\lambda(d z z).e)(d A(C A))$$

## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

▶  $C$  such that  $C \mapsto_{\rho\delta} \lambda y. (\lambda(dx x).e) (d y (C y))$

▶  $A$  such that  $A \mapsto_{\rho\delta} C A$

$$\begin{array}{ccc} A \twoheadrightarrow C A & \longrightarrow & (\lambda(d z z).e) (d A (C A)) \\ & & \downarrow \\ & & (\lambda(d z z).e) (d (C A) (C A)) \end{array}$$

## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

►  $C$  such that  $C \mapsto_{\rho\delta} \lambda y.(\lambda(dx x).e)(dy(C y))$

►  $A$  such that  $A \mapsto_{\rho\delta} C A$

$$\begin{array}{c} A \twoheadrightarrow C A \longrightarrow (\lambda(d z z).e)(d A(C A)) \\ \downarrow \\ (\lambda(d z z).e)(d(C A)(C A)) \\ \downarrow \\ e \end{array}$$

## (Non-)Confluence of the $\rho$ -calculus – part II

Non-linear patterns do not mix well with non-termination **[Klop 80]**

►  $C$  such that  $C \mapsto_{\rho\delta} \lambda y.(\lambda(dx x).e)(dy(C y))$

►  $A$  such that  $A \mapsto_{\rho\delta} C A$

$$\begin{array}{ccc}
 A \longrightarrow C A & \longrightarrow & (\lambda(d z z).e)(d A(C A)) \\
 \downarrow & & \downarrow \\
 C e & & (\lambda(d z z).e)(d(C A)(C A)) \\
 & & \downarrow \\
 & & e
 \end{array}$$

# Expressiveness

1. **Embedding the  $\lambda$  into the  $\rho$ .**  $\varphi : \lambda \Rightarrow \rho$

(a)  $\varphi(x) = x$

(b)  $\varphi(\lambda x.M) = \lambda x.\varphi(M)$

(c)  $\varphi(M N) = \varphi(M) \varphi(N)$

Theorem: If  $M \mapsto_{\beta} N$ , then  $\varphi(M) \mapsto_{\rho} \varphi(N)$

# Expressiveness

## 1. Embedding the $\lambda$ into the $\rho$ . $\varphi : \lambda \Rightarrow \rho$

(a)  $\varphi(x) = x$

(b)  $\varphi(\lambda x.M) = \lambda x.\varphi(M)$

(c)  $\varphi(M N) = \varphi(M) \varphi(N)$

Theorem: If  $M \mapsto_{\beta} N$ , then  $\varphi(M) \mapsto_{\rho} \varphi(N)$

## 2. Encoding Rewriting

(a) A rewrite system  $\mathcal{R}$  can be represented as a structure containing all the rules

(b) Reduction paths can be encoded

If  $t_1 \mapsto_{\mathcal{R}} t_2$ , then  $\exists A$  such that  $A \bullet t_1 \mapsto_{\rho\delta} t_2$



## Normalization failure

$f : (\alpha \rightarrow \alpha) \rightarrow \alpha$  and  $\Gamma = x : \alpha \rightarrow \alpha$ ,  $\omega \triangleq \lambda f x.x (f x)$

$$\frac{\frac{\Gamma \vdash_{\Sigma} f : (\alpha \rightarrow \alpha) \rightarrow \alpha \quad \Gamma \vdash_{\Sigma} x : \alpha \rightarrow \alpha}{\Gamma \vdash_{\Sigma} f x : \alpha} \quad \frac{\Gamma \vdash_{\Sigma} x : \alpha \rightarrow \alpha \quad \overline{\Gamma \vdash_{\Sigma} f x : \alpha}}{\Gamma \vdash_{\Sigma} x (f x) : \alpha}}{\vdash_{\Sigma} \omega \equiv \lambda f x.x (f x) : \alpha \rightarrow \alpha}$$

$$\vdash_{\Sigma} \omega (f \omega) : \alpha$$

## The relation $\subseteq$ and *first*

$$\begin{array}{l}
 f \bar{P} \not\subseteq \lambda Q.B \\
 f \bar{P} \not\subseteq g \bar{B} \quad \text{if } f \neq g \vee \exists i, P_i \not\subseteq B_i \\
 P \not\subseteq (\lambda Q.A) B \quad \text{if } Q \not\subseteq B \vee P \not\subseteq A
 \end{array}$$

$$\begin{array}{l}
 \text{first}(A_1, A_2, \dots, A_n) \triangleq X \rightarrow ((\text{stk} \rightarrow A_n X \wr I) (\dots (\text{stk} \rightarrow A_2 X \wr I) (A_1 X))) \\
 \text{first}(A_1, A_2, \dots, A_n) B \mapsto_{\rho\delta} \text{first}(A_2, \dots, A_n) B
 \end{array}$$

## Encoding of TRSs

$$\begin{aligned}
 \llbracket \mathcal{R} \rrbracket &= \lambda \text{rec } z . \text{first} \left( \begin{array}{l} \lambda l_1 . z (\text{rec } z) r_1, \\ \dots, \\ \lambda a_1 \bar{x} . z (\text{rec } z) a_1 (\overline{z (\text{rec } z) x}), \\ \dots \end{array} \right) \\
 &\wr \lambda \text{Rec } z . \text{first} \left( \begin{array}{l} \lambda l_1 . z (\text{rec } z) r_1, \\ \dots, \\ \lambda y . y \end{array} \right)
 \end{aligned}$$

## Positiveness

In CIC, the constructor  $F : (x_1 : A_1) \dots (x_n : A_n).R$  is accepted only if  $R$  is *positive* in each  $A_i$ :

1.  $R$  is positive in  $T$  if  $R$  does not occur in  $T$
2.  $R$  is positive in  $(R\vec{t})$  if  $R$  does not occur in  $\vec{t}$
3.  $R$  is positive in  $(x : A)C$  if  $R$  does not occur in  $A$  and  $R$  is positive in  $C$

## Encoding the $P^2TS$ into $\lambda$ -calculus

$$\llbracket x \rrbracket \triangleq x$$

$$\llbracket f \rrbracket \triangleq \lambda x_1 \dots \lambda x_{\alpha_f}. (\lambda z. (z x_1 \dots x_{\alpha_f}))$$

$$\llbracket f B_1 \dots B_{\alpha_f} \rrbracket \triangleq \lambda z. (z B_1 \dots B_{\alpha_f})$$

## Encoding the $P^2TS$ into $\lambda$ -calculus

$$\llbracket x \rrbracket \triangleq x$$

$$\llbracket f \rrbracket \triangleq \lambda x_1 \dots \lambda x_{\alpha_f}. (\lambda z. (z x_1 \dots x_{\alpha_f}))$$

$$\llbracket f B_1 \dots B_{\alpha_f} \rrbracket \triangleq \lambda z. (z B_1 \dots B_{\alpha_f})$$

$$\llbracket \lambda(f P_1 \dots P_p). A \rrbracket \triangleq \lambda u. (u x_{\perp} \dots x_{\perp} \llbracket \lambda P_1 \dots \lambda P_p. \lambda x'_{p+1} \dots \lambda x'_{\alpha_f}. A \rrbracket )$$

## Encoding the $P^2TS$ into $\lambda$ -calculus

$$\llbracket x \rrbracket \triangleq x$$

$$\llbracket f \rrbracket \triangleq \lambda x_1 \dots \lambda x_{\alpha_f}. (\lambda z. (z x_1 \dots x_{\alpha_f}))$$

$$\llbracket f B_1 \dots B_{\alpha_f} \rrbracket \triangleq \lambda z. (z B_1 \dots B_{\alpha_f})$$

$$\llbracket \lambda (f P_1 \dots P_p). A \rrbracket \triangleq \lambda u. (u x_{\perp} \dots x_{\perp} \llbracket \lambda P_1 \dots \lambda P_p. \lambda x'_{p+1} \dots \lambda x'_{\alpha_f}. A \rrbracket )$$

$$\llbracket \lambda x. A \rrbracket \triangleq \lambda x. \llbracket A \rrbracket$$

$$\llbracket A B \rrbracket \triangleq \llbracket A \rrbracket \llbracket B \rrbracket$$

## Encoding the $P^2TS$ into $\lambda$ -calculus

$$\llbracket x \rrbracket \triangleq x$$

$$\llbracket f \rrbracket \triangleq \lambda x_1 \dots \lambda x_{\alpha_f}. (\lambda z. (z x_1 \dots x_{\alpha_f}))$$

$$\llbracket f B_1 \dots B_{\alpha_f} \rrbracket \triangleq \lambda z. (z B_1 \dots B_{\alpha_f})$$

$$\llbracket \lambda(f P_1 \dots P_p). A \rrbracket \triangleq \lambda u. (u x_{\perp} \dots x_{\perp} \llbracket \lambda P_1 \dots \lambda P_p. \lambda x'_{p+1} \dots \lambda x'_{\alpha_f}. A \rrbracket )$$

$$\llbracket \lambda x. A \rrbracket \triangleq \lambda x. \llbracket A \rrbracket$$

$$\llbracket A B \rrbracket \triangleq \llbracket A \rrbracket \llbracket B \rrbracket$$

$$\llbracket A \wr B \rrbracket \triangleq \lambda x_1 \dots \lambda x_{\alpha}. \left( (\lambda z. (\llbracket A \rrbracket x_1 \dots x_{\alpha})) (\llbracket B \rrbracket x_1 \dots x_{\alpha}) \right)$$



## An example of translated term

$$\frac{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket}{\llbracket \lambda (f x). x \rrbracket} \left( \overbrace{(\lambda x_1. \lambda z. (z x_1))}^{\llbracket f \rrbracket} \overbrace{(\lambda v. v)}^{\llbracket a \rrbracket} \right)$$

*Note: In the original image, the term  $(\lambda u. (u(\lambda x. x)))$  is highlighted in blue.*

## An example of translated term

$$\begin{array}{l}
 \overbrace{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket} \\
 \underbrace{\llbracket \lambda (f x). x \rrbracket} \quad \overbrace{\llbracket f \rrbracket} \quad \overbrace{\llbracket a \rrbracket} \\
 (\lambda y. \left( \underbrace{(\lambda u. (u(\lambda x. x)))}_{\llbracket \lambda u. (u(\lambda x. x)) \rrbracket} y \right)) \left( \underbrace{(\lambda x_1. \lambda z. (z x_1))}_{\llbracket f \rrbracket} \underbrace{(\lambda v. v)}_{\llbracket a \rrbracket} \right) \\
 \mapsto_{\beta} (\lambda y. (y(\lambda x. x))) \left( (\lambda x_1. \lambda z. (z x_1)) (\lambda v. v) \right)
 \end{array}$$

## An example of translated term

$$\begin{array}{l}
 \overbrace{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket} \\
 \underbrace{\llbracket \lambda (f x). x \rrbracket} \quad \underbrace{\llbracket f \rrbracket} \quad \underbrace{\llbracket a \rrbracket} \\
 (\lambda y. \left( \underbrace{(\lambda u. (u(\lambda x. x)))}_{\text{blue}} y \right)) \left( \underbrace{(\lambda x_1. \lambda z. (z x_1))}_{\text{blue}} \underbrace{(\lambda v. v)}_{\text{blue}} \right) \\
 \mapsto_{\beta} (\lambda y. (y(\lambda x. x))) \left( (\lambda x_1. \lambda z. (z x_1)) (\lambda v. v) \right) \\
 \mapsto_{\beta} (\lambda y. (y(\lambda x. x))) (\lambda z. (z(\lambda v. v)))
 \end{array}$$

## An example of translated term

$$\begin{aligned}
 & \overbrace{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket} \\
 & \quad \overbrace{\llbracket \lambda (f x). x \rrbracket} \quad \overbrace{\llbracket f \rrbracket} \quad \overbrace{\llbracket a \rrbracket} \\
 & (\lambda y. \left( \overbrace{(\lambda u. (u(\lambda x. x)))} y \right) \left( \overbrace{(\lambda x_1. \lambda z. (z x_1))} \overbrace{(\lambda v. v)} \right)) \\
 \mapsto_{\beta} & (\lambda y. (y(\lambda x. x))) \left( (\lambda x_1. \lambda z. (z x_1)) (\lambda v. v) \right) \\
 \mapsto_{\beta} & (\lambda y. (y(\lambda x. x))) (\lambda z. (z(\lambda v. v))) \\
 \mapsto_{\beta} & (\lambda z. (z(\lambda v. v))) (\lambda x. x)
 \end{aligned}$$

## An example of translated term

$$\begin{aligned}
 & \overbrace{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket} \\
 & \quad \overbrace{\llbracket \lambda (f x). x \rrbracket} \quad \overbrace{\llbracket f \rrbracket} \quad \overbrace{\llbracket a \rrbracket} \\
 & \quad (\lambda y. \left( \overbrace{(\lambda u. (u(\lambda x. x)))} y \right) \left( \overbrace{(\lambda x_1. \lambda z. (z x_1))} \overbrace{(\lambda v. v)} \right)) \\
 \mapsto_{\beta} & \quad (\lambda y. (y(\lambda x. x))) \left( (\lambda x_1. \lambda z. (z x_1)) (\lambda v. v) \right) \\
 \mapsto_{\beta} & \quad (\lambda y. (y(\lambda x. x))) (\lambda z. (z(\lambda v. v))) \\
 \mapsto_{\beta} & \quad (\lambda z. (z(\lambda v. v))) (\lambda x. x) \\
 \mapsto_{\beta} & \quad (\lambda x. x) (\lambda v. v)
 \end{aligned}$$

## An example of translated term

$$\begin{aligned}
 & \overbrace{\llbracket \lambda y. (\lambda (f x). x) y \rrbracket} \\
 & \quad \overbrace{\llbracket \lambda (f x). x \rrbracket} \quad \overbrace{\llbracket f \rrbracket} \quad \overbrace{\llbracket a \rrbracket} \\
 & \quad (\lambda y. \left( \overbrace{(\lambda u. (u(\lambda x. x)))} y \right) \left( \overbrace{(\lambda x_1. \lambda z. (z x_1))} \overbrace{(\lambda v. v)} \right)) \\
 & \mapsto_{\beta} (\lambda y. (y(\lambda x. x))) \left( (\lambda x_1. \lambda z. (z x_1)) (\lambda v. v) \right) \\
 & \mapsto_{\beta} (\lambda y. (y(\lambda x. x))) (\lambda z. (z(\lambda v. v))) \\
 & \mapsto_{\beta} (\lambda z. (z(\lambda v. v))) (\lambda x. x) \\
 & \mapsto_{\beta} (\lambda x. x) (\lambda v. v) \\
 & \mapsto_{\beta} (\lambda v. v) \\
 & = \llbracket a \rrbracket
 \end{aligned}$$

## The type of a translated constant

Supposing  $\vdash_{\Sigma} f : \Pi x:\iota.\iota$

$$\vdash_{\lambda\omega} \llbracket f \rrbracket = \lambda x_1.\lambda z.(z x_1) : \sigma \rightarrow (\sigma \rightarrow \beta) \rightarrow \beta$$

$$\vdash_{\lambda\omega} \llbracket f B \rrbracket : (\sigma \rightarrow \beta) \rightarrow \beta$$

## Enhanced translation

$$\bigwedge \sigma_1, \dots, \sigma_\alpha \triangleq \Pi(\beta : *) . ((\sigma_1 \rightarrow \dots \sigma_\alpha \rightarrow \beta) \rightarrow \beta)$$

$$\llbracket f \rrbracket \triangleq \lambda x_1 . \lambda(\beta : *) (\lambda z . (z x_1)) : \sigma \rightarrow \bigwedge \sigma$$

$$\llbracket \lambda f x . A \rrbracket \triangleq \lambda u . (u \tau \lambda x . \llbracket A \rrbracket) : (\bigwedge \sigma) \rightarrow \tau$$

where  $\llbracket \Gamma \rrbracket \vdash_{\lambda\omega} \llbracket A \rrbracket : \tau$

$$\vdash x_\perp : \perp \triangleq \Pi(\beta : *) . \beta$$



## Use of types depending on types

$$\vdash_{\Sigma} \quad x \quad : \quad \Pi y:\iota . \iota$$

$$\beta_x : * \rightarrow *, \beta_y : * \quad \vdash_{\lambda\omega} \quad [[x]] \quad : \quad \beta_y \rightarrow \beta_x \beta_y$$

$\lambda y.y$	$\beta_x := \lambda\beta : *. \beta$
$\lambda y.a$	$\beta_x := \lambda\beta : *. \bigwedge \emptyset$
$f$	$\beta_x := \lambda\beta : *. \bigwedge \beta$

## Disjunctive connectors

When dealing with  $\vee$  and  $\exists$ , some part of the definition can not be decomposed properly

## Disjunctive connectors

When dealing with  $\vee$  and  $\exists$ , some part of the definition can not be decomposed properly

With  $P \rightarrow (Q \wedge R) \vee S$  the new rules are:

$$(P I_l) \frac{\Gamma \vdash Q \quad \Gamma \vdash R}{\Gamma \vdash P} \quad (P I_r) \frac{\Gamma \vdash S}{\Gamma \vdash P} \quad (P E) \frac{\Gamma \vdash P \quad \Gamma, Q \wedge R \vdash U \quad \Gamma, S \vdash U}{\Gamma \vdash U}$$

## Disjunctive connectors

When dealing with  $\vee$  and  $\exists$ , some part of the definition can not be decomposed properly

With  $P \rightarrow (Q \wedge R) \vee S$  the new rules are:

$$(P I_l) \frac{\Gamma \vdash Q \quad \Gamma \vdash R}{\Gamma \vdash P} \quad (P I_r) \frac{\Gamma \vdash S}{\Gamma \vdash P} \quad (P E) \frac{\Gamma \vdash P \quad \Gamma, Q \wedge R \vdash U \quad \Gamma, S \vdash U}{\Gamma \vdash U}$$

The discrepancy between  $(P I_l)$  and the second assumption of  $(P E)$  may ruin cut elimination, and suggests further decomposition:

$$(P E) \frac{\Gamma \vdash P \quad \Gamma, Q, R \vdash U \quad \Gamma, S \vdash U}{\Gamma \vdash U}$$

## Conservativity

$$\begin{array}{c} (K E) \frac{(Ax) -}{\vdots} \\ (P I) \frac{\frac{def \vdash H_1 \quad \dots \quad def \vdash H_n}{\vdots}}{def \vdash P} \end{array}$$

$$\begin{array}{c} (P E) \frac{P, \Gamma \vdash P \quad \dots \quad P, \Gamma \vdash \gamma}{\vdots} \\ (K I) \frac{\vdots}{P \vdash def} \end{array}$$

## About unsound rules

It is well-known that the rewrite rule  $R \rightarrow R \Rightarrow \perp$  gives an unsound deduction modulo

Its associated introduction and elimination rules are

$$(R I) \frac{\Gamma, R \vdash \perp}{\Gamma \vdash R} \qquad (R E) \frac{\Gamma \vdash R \quad \Gamma \vdash R}{\Gamma \vdash \perp}$$

## About unsound rules

It is well-known that the rewrite rule  $R \rightarrow R \Rightarrow \perp$  gives an unsound deduction modulo

Its associated introduction and elimination rules are

$$(R I) \frac{\Gamma, R \vdash \perp}{\Gamma \vdash R} \quad (R E) \frac{\Gamma \vdash R \quad \Gamma \vdash R}{\Gamma \vdash \perp}$$

and the (shortest) proof of  $\vdash \perp$  has the proof term

$$(\lambda R(\alpha). \alpha R(\alpha)) R(\lambda R(\alpha). \alpha R(\alpha))$$

## Curiosities

- Proof terms with patterns for the usual connectives

$$(\wedge I) \frac{\Gamma \vdash \pi : \phi \quad \Gamma \vdash \pi' : \psi}{\Gamma \vdash \wedge(\pi, \pi') : \phi \wedge \psi}$$

$$(\wedge E_l) \frac{\Gamma \vdash \pi : \phi \wedge \psi}{\Gamma \vdash (\lambda \wedge (x, y).x)\pi : \phi}$$



## Curiosities

- Proof terms with patterns for the usual connectives

$$(\wedge I) \frac{\Gamma \vdash \pi : \phi \quad \Gamma \vdash \pi' : \psi}{\Gamma \vdash \wedge(\pi, \pi') : \phi \wedge \psi}$$

$$(\wedge E_l) \frac{\Gamma \vdash \pi : \phi \wedge \psi}{\Gamma \vdash (\lambda \wedge (x, y).x)\pi : \phi}$$

- The NDM formalization of higher-order logic gives the rules for higher-order quantifiers

Predicates defined by induction give some natural rules

$$(N E) \frac{\Gamma \vdash n \in N \quad \Gamma \vdash 0 \in P \quad \Gamma, m \in P \vdash S(m) \in P}{\Gamma \vdash n \in P}$$