Typage et déduction dans le calcul de réécriture

Benjamin Wack

Encadrants : C. Kirchner, L. Liquori

Deduction and computation

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

Deduction and computation

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

- Simply typed λ-calculus **[Church 40, Curry 34]**
  - Ensures strong normalization
  - Isomorphism with natural deduction for intuitionistic logic **[Curry, Howard, de Bruijn]**

Deduction and computation

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

- Simply typed λ-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 λ-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 ρP
4. Using the ρ-calculus for deduction
   - P^2TS-proof terms for deduction modulo
   - Generalized Natural Deduction

The Untyped Syntax

$$\begin{array}{lcll} \mathcal{P} & \subseteq & \mathcal{T} & \text{Patterns} \\ \mathcal{T} & ::= & \mathcal{X} \mid \mathcal{K} \mid \lambda\mathcal{P}.\mathcal{T} \mid \mathcal{T}\,\mathcal{T} \mid \mathcal{T} \wr \mathcal{T} & \text{Terms} \end{array}$$

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 *α-conversion* and Barendregt's *hygiene-convention*

Some ρ-terms

$(\lambda x.x\ x)\ (\lambda x.x\ x)$ the λ-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

Some ρ-terms

$(\lambda x.x\ x)\ (\lambda x.x\ x)$	the λ-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

Some ρ-terms

$(\lambda x.x\ x)\ (\lambda x.x\ x)$	the λ-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

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

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

Some ρ-reductions

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

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

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

Some ρ-reductions

$(\lambda x.x\ x)\ (\lambda x.x\ x)$ $\qquad\qquad$ $\mapsto_{\rho}\ \{\omega\ \omega\} \mapsto\!\!\!\!\!\rightarrow_{\rho\delta}\ \ldots$

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

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

Some ρ-reductions

$(\lambda x.x\ x)\ (\lambda x.x\ x)$ $\quad\quad\quad\quad$ $\mapsto_\rho \{\omega\ \omega\} \mapsto\!\!\!\!\mapsto_{\rho\delta} \ \ldots$

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

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

Some ρ-reductions

$(\lambda x.x\ x)\ (\lambda x.x\ x)$ $\quad\quad$ $\mapsto_\rho \{\omega\ \omega\} \mapsto\!\!\!\!\mapsto_{\rho\delta} \ \ldots$

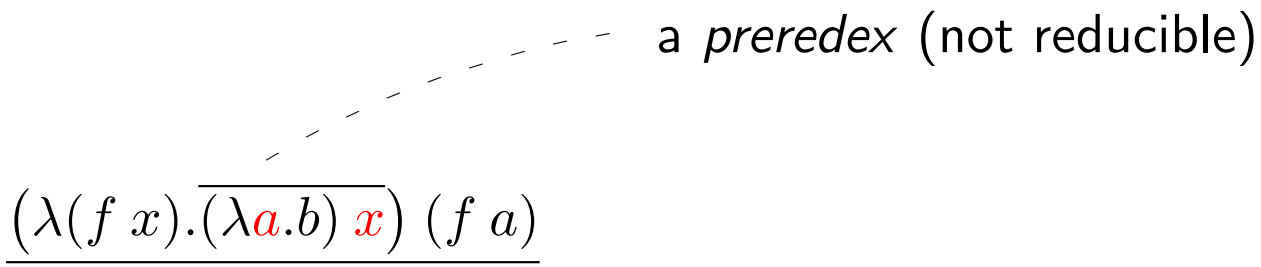
$(\lambda(f\ x\ y).g\ y\ x)\ (f\ a\ b)$ $\quad\quad$ $\mapsto_\rho g\ b\ a$

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

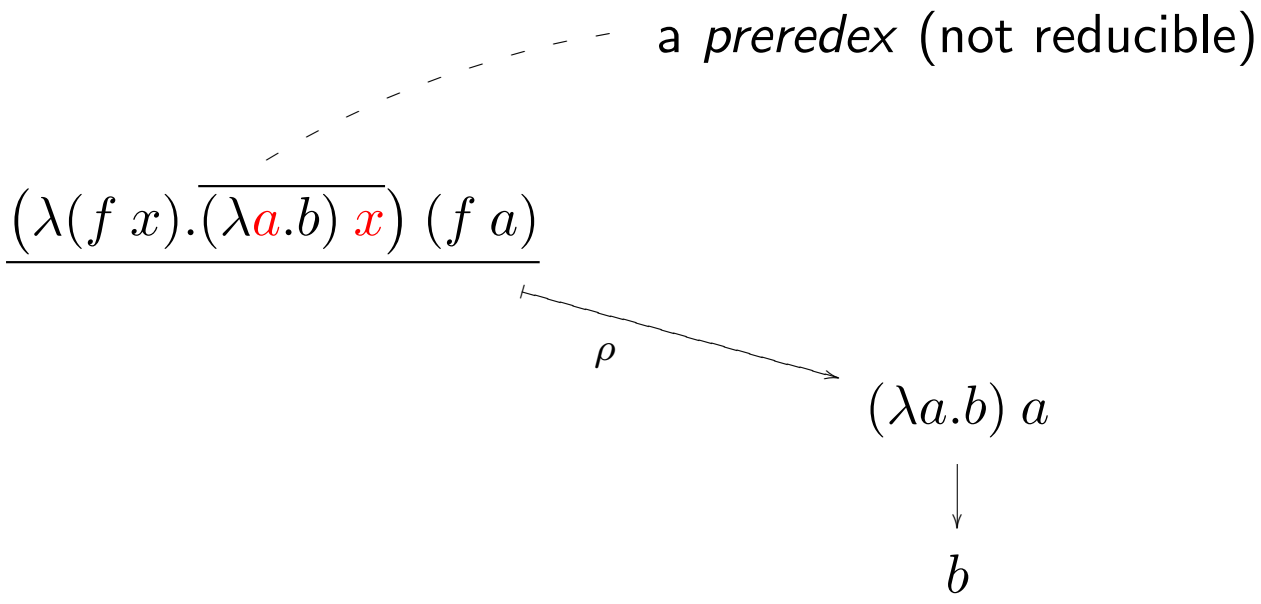
About preredexes

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

About preredexes



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 ρ-calculus

- The λ-calculus is fully embedded in the ρ-calculus **[Cirstea & Kirchner 98]**
  - β-reductions are faithfully mimicked
  - a λ-term ρ-reduces to λ-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 ρP
4. Using the ρ-calculus for deduction
   - P^2TS-proof terms for deduction modulo
   - Generalized Natural Deduction

A Simple Type System ρ_1

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

A Simple Type System ρ_1

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

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

A Simple Type System ρ_1

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

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

$$\frac{\Gamma, \Delta \vdash_\Sigma P : \sigma \quad \Gamma, \Delta \vdash_\Sigma A : \tau}{\Gamma \vdash_\Sigma \lambda(P{:}\Delta).A : \sigma \to \tau}\ (Abs) \qquad \mathcal{D}om(\Delta) = \mathcal{FV}(P)$$

A Simple Type System ρ_1

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

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

$$\frac{\Gamma, \Delta \vdash_\Sigma P : \sigma \quad \Gamma, \Delta \vdash_\Sigma A : \tau}{\Gamma \vdash_\Sigma \lambda(P{:}\Delta).A : \sigma \to \tau}\ (Abs) \qquad \mathcal{D}om(\Delta) = \mathcal{FV}(P)$$

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

Polymorphic extensions

à la Church	*à la* Curry
$\dfrac{\Gamma \vdash_\Sigma A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_\Sigma \lambda\alpha.A : \forall\alpha.\sigma}\ (Abs\forall)$	
$\dfrac{\Gamma \vdash_\Sigma A : \forall\alpha.\sigma}{\Gamma \vdash_\Sigma A\tau : \sigma[\alpha := \tau]}\ (App\forall)$	

Polymorphic extensions

à la Church	*à la* Curry
$\dfrac{\Gamma \vdash_\Sigma A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_\Sigma \lambda\alpha.A : \forall\alpha.\sigma}\ (Abs\forall)$	$\dfrac{\Gamma \vdash_\Sigma A : \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash_\Sigma A : \forall\alpha.\sigma}\ (Abs\forall)$
$\dfrac{\Gamma \vdash_\Sigma A : \forall\alpha.\sigma}{\Gamma \vdash_\Sigma A\tau : \sigma[\alpha := \tau]}\ (App\forall)$	$\dfrac{\Gamma \vdash_\Sigma A : \forall\alpha.\sigma}{\Gamma \vdash_\Sigma A : \sigma[\alpha := \tau]}\ (App\forall)$

Polymorphic extensions

à la Church

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

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

à la Curry

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

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

$$\forall(f{:}\sigma) \in \Sigma, \quad \sigma \equiv \forall\overline{\alpha}(\sigma_1 \to \ldots \iota(\overline{\beta}))$$
where $\overline{\beta} = \mathcal{BV}(\sigma)$

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 $A \mapsto\!\!\!\!\rightarrow_{\rho\delta} B$, 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, $\left.\begin{array}{ll} \textit{(typechecking)} & \Gamma \vdash_\Sigma \mathcal{T} : \sigma \; ? \\ \textit{(type reconstruction)} & \Gamma \vdash_\Sigma \mathcal{T} : ? \end{array}\right\}$ 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 $\forall\overline{\alpha}.\tau$ where τ 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

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

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

Normalization failure

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

$$\begin{aligned} \omega \quad \omega \quad &\equiv \quad (\lambda \quad x \ .x \quad x) \qquad \omega \\ &\to_\rho \quad \omega \quad \omega \\ &\to_\rho \quad \cdots \end{aligned}$$

Normalization failure

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

$$\begin{aligned} \omega\ (f\ \omega) &\equiv (\lambda(f\ x)\,.\,x\ (f\ x)))\ \ (f\ \omega) \\ &\to_\rho \omega\ (f\ \omega) \\ &\to_\rho \cdots \end{aligned}$$

Normalization failure (cont'd)

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

$$\dfrac{\dfrac{}{\Gamma \vdash_\Sigma x : \alpha \to \alpha} \qquad \dfrac{\Gamma \vdash_\Sigma f : (\alpha \to \alpha) \to \alpha \quad \Gamma \vdash_\Sigma x : \alpha \to \alpha}{\Gamma \vdash_\Sigma f\, x : \alpha}}{\Gamma \vdash_\Sigma x\, (f\, x) : \alpha}$$

$$\vdash_\Sigma \omega\, (f\, \omega) : \alpha$$

Encoding rewriting systems in the ρ-calculus

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

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

Encoding rewriting systems in the ρ-calculus

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

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

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

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

Encoding rewriting systems in the ρ-calculus

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

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

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

$$\mapsto\!\!\!\!\rightarrow_{\rho\delta}\ \ (\lambda 0.M)\ N\ \ \wr\ \ (\lambda 0.\widetilde{M+1})\ \widetilde{N-1}\ \ \cdots\ \ (\lambda 0.\widetilde{M+N})\ 0\ \ \wr\ \ (\lambda(S\, x).\ldots)\ 0$$

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

Detecting matching failures: the symbol stk

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

Detecting matching failures: the symbol stk

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

$$
\begin{array}{rcll}
(\lambda P{:}\Delta.A)\ B & \mapsto_{\mathsf{stk}} & \mathsf{stk} & \text{if } P \not\sqsubseteq B \\
\mathsf{stk} \wr A & \mapsto_{\mathsf{stk}} & A & \\
A \wr \mathsf{stk} & \mapsto_{\mathsf{stk}} & A & \\
\mathsf{stk}\ A & \mapsto_{\mathsf{stk}} & \mathsf{stk} &
\end{array}
$$

Detecting matching failures: the symbol stk

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

$$
\begin{array}{rcll}
(\lambda P{:}\Delta.A)\ B & \mapsto_{\mathsf{stk}} & \mathsf{stk} & \text{if } P \not\sqsubseteq B \\
\mathsf{stk} \wr A & \mapsto_{\mathsf{stk}} & A & \\
A \wr \mathsf{stk} & \mapsto_{\mathsf{stk}} & A & \\
\mathsf{stk}\ A & \mapsto_{\mathsf{stk}} & \mathsf{stk} &
\end{array}
$$

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

Systematic encoding

- There exists a ρ-term first (using stk) such that

$$(\mathit{first}\ A_1\ A_2 \ldots A_n)\ B \quad \mapsto^{\mathsf{stk}}_{\rho\delta} \quad A_{i+1}\ B$$

$$\text{if} \quad \begin{array}{rll} A_{i+1}\ B & \not\mapsto^{\mathsf{stk}}_{\rho\delta} & \mathsf{stk} \\ \forall j \leq i,\ A_j\ B & \mapsto^{\mathsf{stk}}_{\rho\delta} & \mathsf{stk} \end{array}$$

Systematic encoding

- There exists a ρ-term first (using stk) such that

$$(\mathit{first}\ A_1\ A_2 \ldots A_n)\ B \quad \mapsto\!\!\!\!\!\rightarrow^{\mathsf{stk}}_{\rho\delta} \quad A_{i+1}\ B$$

$$\text{if} \quad \begin{array}{rl} A_{i+1}\ B & \not\mapsto\!\!\!\!\!\rightarrow^{\mathsf{stk}}_{\rho\delta}\ \mathsf{stk} \\ \forall j \leq i,\ A_j\ B & \mapsto\!\!\!\!\!\rightarrow^{\mathsf{stk}}_{\rho\delta}\ \mathsf{stk} \end{array}$$

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

$$[\![\mathcal{R}]\!] \ = \ \lambda(\mathit{rec}\ z)\ .\ \mathit{first} \left(\begin{array}{l} \lambda l_1\ .\ z\ (\mathit{rec}\ z)\ r_1 \\ \ldots \\ \lambda(a_1\ \overline{x})\ .\ z\ (\mathit{rec}\ z)\ a_1(\overline{z\ (\mathit{rec}\ z)\ x}) \\ \ldots \end{array} \right)$$

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, ω 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:

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

Logical inconsistency

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

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

- How to fix it ?

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

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 ρP
4. Using the ρ-calculus for deduction
   - P^2TS-proof terms for deduction modulo
   - Generalized Natural Deduction

Dependent type discipline in P^2TS

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

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

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

Dependent type discipline in P^2TS

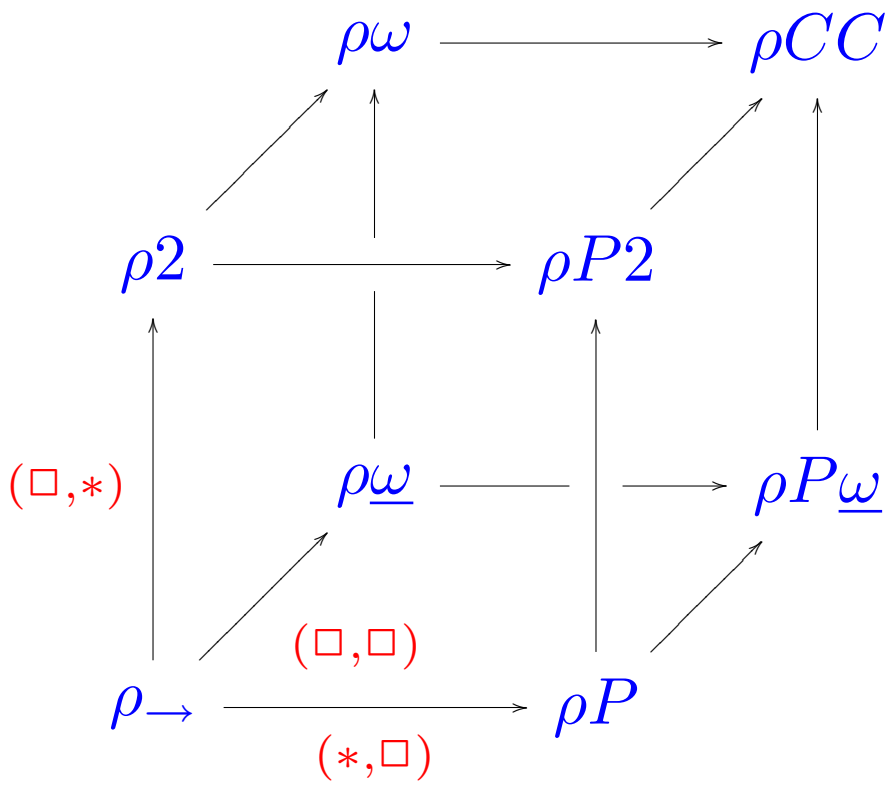
$$\frac{\Gamma, \Delta \vdash_\Sigma B : C \quad \Gamma \vdash_\Sigma \Pi P{:}\Delta.C : s}{\Gamma \vdash_\Sigma \lambda P{:}\Delta.B : \Pi P{:}\Delta.C} \ (Abs)$$

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

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

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

The ρ-cube


$\rho\omega$
ρCC
$\rho 2$
$\rho P2$
$(\Box,*)$
$\rho\underline{\omega}$
$\rho P\underline{\omega}$
$(\Box,\Box)$
$\rho\rightarrow$
ρP
$(*,\Box)$

Typing properties

[Barthe, Cirstea, Kirchner & Liquori 03]

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

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

Consistency: $A \in \mathsf{Nf}(\rho\delta) \ \Rightarrow \ \not\vdash_\Sigma A : \bot \ (\stackrel{\triangle}{=} \ \forall x{:}* .x)$

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

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 ω must have a common type σ

Strong normalization : sketch of the proof

Theorem [Wack 04]:

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

1. Find a translation $[\![\ ]\!] : P^2TS \rightarrow \lambda\omega$ correct w.r.t. reductions

   If $A \mapsto_{\rho\infty\delta} B$, then $[\![A]\!] \overset{\beta}{\mapsto\!\!\!\!\rightarrow} [\![B]\!]$ in at least one step

Strong normalization : sketch of the proof

Theorem [Wack 04]:

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

1. Find a translation $[\![\ ]\!] : P^2TS \rightarrow \lambda\omega$ correct w.r.t. reductions

   If $A \mapsto_{\rho\infty\delta} B$, then $[\![A]\!] \overset{\beta}{\mapsto\!\!\!\!\rightarrow} [\![B]\!]$ in at least one step

2. Typability of the translated terms

$$\Sigma, \Gamma \vdash_{\Sigma} A : C \quad \Rightarrow \quad \exists \tau, \quad [\![\Gamma]\!] \vdash_{\lambda\omega} [\![A]\!] : \tau$$

Strong normalization : sketch of the proof

Theorem [Wack 04]:

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

1. Find a translation $[\![\ ]\!] : P^2TS \rightarrow \lambda\omega$ correct w.r.t. reductions

   If $A \mapsto_{\rho\beta\delta} B$, then $[\![A]\!] \overset{\beta}{\mapsto\!\!\!\!\rightarrow} [\![B]\!]$ in at least one step

2. Typability of the translated terms

$$\Sigma, \Gamma \vdash_{\Sigma} A : C \quad \Rightarrow \quad \exists \tau, \quad [\![\Gamma]\!] \vdash_{\lambda\omega} [\![A]\!] : \tau$$

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

Correctness of reductions

- $[\![(\lambda(f\ x).x)\ (f\ a)]\!] = \big(\lambda u.(u(\lambda x.x))\big)\ \big((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)\big) \mapsto\!\!\!\!\rightarrow_\beta\ \lambda v.v = [\![a]\!]$

Correctness of reductions

- $[\![(\lambda(f\ x).x)\ (f\ a)]\!] = \big(\lambda u.(u(\lambda x.x))\big)\ \big((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)\big) \mapsto\!\!\!\!\rightarrow_\beta\ \lambda v.v = [\![a]\!]$

- The ρ-term $\big(\lambda y.(\lambda(f\ x).x)\ y\big)\ (f\ a)$ features a preredex

Correctness of reductions

- $[\![(\lambda(f\ x).x)\ (f\ a)]\!] = \big(\lambda u.(u(\lambda x.x))\big)\ \big((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)\big) \mapsto\!\!\!\!\rightarrow_\beta\ \lambda v.v = [\![a]\!]$
- The ρ-term $\big(\lambda y.(\lambda(f\ x).x)\ y\big)\ (f\ a)$ features a preredex
- Thus, the reductions of the λ-term $[\![\big(\lambda y.(\lambda(f\ x).x)\ y\big)\ (f\ a)]\!]$ must mimick first an external ρ-reduction

Correctness of reductions

- $[\![(\lambda(f\ x).x)\ (f\ a)]\!] = \big(\lambda u.(u(\lambda x.x))\big)\ \big((\lambda x_1.\lambda z.(zx_1))(\lambda v.v)\big) \mapsto\!\!\!\!\rightarrow_\beta\ \lambda v.v = [\![a]\!]$
- The ρ-term $\big(\lambda y.(\lambda(f\ x).x)\ y\big)\ (f\ a)$ features a preredex
- Thus, the reductions of the λ-term $[\![\big(\lambda y.(\lambda(f\ x).x)\ y\big)\ (f\ a)]\!]$ must mimick first an external ρ-reduction
- Remark: a term produced by the translation may have additional reductions

The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} [\![f\ B]\!] \quad : \quad (\sigma \to \beta) \to \beta$$

$$\vdash_{\lambda\omega} [\![\lambda(f\ x).A]\!] \quad : \quad ((\sigma \to \tau) \to \gamma) \to \gamma$$

where τ is the type of $[\![A]\!]$

The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} [\![f\ B]\!] \quad : \quad (\sigma \to \beta) \to \beta$$

$$\vdash_{\lambda\omega} [\![\lambda(f\ x).A]\!] \quad : \quad ((\sigma \to \tau) \to \gamma) \to \gamma$$

where τ is the type of $[\![A]\!]$

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

The type of a translated pattern

- A naive translation gives

$$\vdash_{\lambda\omega} [\![f\ B]\!] \quad : \quad (\sigma \to \beta) \to \beta$$

$$\vdash_{\lambda\omega} [\![\lambda(f\ x).A]\!] \quad : \quad ((\sigma \to \tau) \to \gamma) \to \gamma$$

where τ is the type of $[\![A]\!]$

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

- The actual translation features terms depending on types

$$[\![f\ B]\!] \quad : \quad \forall\beta.(\sigma \to \beta \to \beta)$$

$$[\![\lambda(f\ x).A]\!] \quad : \quad \forall\beta.(\sigma \to \beta \to \beta) \to \tau$$

The type of a translated variable

- Naive translation

$$
\begin{array}{lll}
x : \Pi y{:}\iota.\iota \vdash_\Sigma x & : & \Pi y{:}\iota \,.\, \iota \\
\vdash_\Sigma \lambda y{:}\iota \,.\, y & : & \Pi y{:}\iota \,.\, \iota \\
\vdash_\Sigma \lambda y{:}\iota \,.\, a & : & \Pi y{:}\iota \,.\, \iota
\end{array}
$$

The type of a translated variable

- Naive translation

$$
\begin{array}{lll}
x : \Pi y{:}\iota.\iota \vdash_\Sigma x & : & \Pi y{:}\iota \,.\, \iota \\
\vdash_\Sigma \lambda y{:}\iota \,.\, y & : & \Pi y{:}\iota \,.\, \iota \\
\vdash_\Sigma \lambda y{:}\iota \,.\, a & : & \Pi y{:}\iota \,.\, \iota
\end{array}
$$

$$
\begin{array}{lll}
\Gamma \vdash_{\lambda\omega} \lambda y{:}\beta_y.y & : & \beta_y \to \beta_y \\
\Gamma \vdash_{\lambda\omega} \lambda y{:}\beta_y.[\![a]\!] & : & \beta_y \to \forall\alpha.(\alpha \to \alpha)
\end{array}
$$

The type of a translated variable

- Naive translation

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

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

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

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

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

- Use of types depending on types

$$\beta_x : * \to *, \beta_y : * \vdash_{\lambda\omega} [\![x]\!] : \beta_y \to \beta_x\ \beta_y$$

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 ρP
4. Using the ρ-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: $e * x \cong x$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall y.(y{*}e' = y) \vdash_{\cong} \forall y.(y{*}e' = y)}\,(Ax)}{\forall y.(y{*}e' = y) \vdash_{\cong} e{*}e' = e}\,(\forall E)}{\forall y.(y{*}e' = y) \vdash_{\cong} e' = e}\,(\cong) \quad \text{with } e{*}e' \cong e'}{\vdash_{\cong} \forall y.(y{*}e' = y) \Rightarrow e' = e}\,(\Rightarrow I)$$

A linear representation of NDM proofs

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall y.(y{*}e' = y) \vdash_{\cong} \forall y.(y{*}e' = y)}\,(Ax)}{\forall y.(y{*}e' = y) \vdash_{\cong} e{*}e' = e}\,(\forall E)}{\forall y.(y{*}e' = y) \vdash_{\cong} e' = e}\,(\cong)}{\vdash_{\cong} \forall y.(y{*}e' = y) \Rightarrow e' = e}\,(\Rightarrow I)$$

with $e{*}e' \cong e'$

- λ-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 :

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

A more explicit representation

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

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

- The new proof term for our example is

$$\lambda\alpha.\Big(\mathsf{Rew}\ \big(\lambda y.(y{=}e)\big)\ (e{*}e')\ \big(\lambda(e{*}x).x\big)\ (\alpha\, e)\Big)$$

A more explicit representation

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

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

- The new proof term for our example is

$$\lambda\alpha.\Big(\mathsf{Rew}\ \big(\lambda y.(y{=}e)\big)\ (e{*}e')\ \big(\lambda(e{*}x).x\big)\ (\alpha\, e)\Big)$$

- **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

$$\dfrac{\dfrac{\dfrac{p \vdash_{\cong} p}{\vdash_{\cong} p \Rightarrow p}\,(\Rightarrow I)}{\vdash_{\cong} q \Rightarrow p}\,(\cong) \qquad \dfrac{\vdots}{\vdash_{\cong} q}}{\vdash_{\cong} p}\,(\Rightarrow E)$$

? 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 λ-proof term can always be extracted from a ρ-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 $\begin{cases} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \bot) \end{cases}$

A simple proof in Natural Deduction...

The theory $\mathcal{T}$ contains at least $\begin{cases} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \bot) \end{cases}$

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

A simple proof in Natural Deduction...

The theory $\mathcal{T}$ contains at least $\begin{cases} X \subseteq Y \Leftrightarrow \forall x(x \in X \Rightarrow x \in Y) \\ \forall x(x \in \emptyset \Rightarrow \bot) \end{cases}$

$$\dfrac{\dfrac{\dfrac{\dfrac{(Ax)\ \overline{\mathcal{T}, x \in \emptyset \vdash \forall x(x \in \emptyset \Rightarrow \bot)}}{(\forall E)\ \mathcal{T}, x \in \emptyset \vdash x \in \emptyset \Rightarrow \bot} \qquad (Ax)\ \overline{\mathcal{T}, x \in \emptyset \vdash x \in \emptyset}}{(\Rightarrow E)\ \mathcal{T}, x \in \emptyset \vdash \bot}}{(\bot E)\ \mathcal{T}, x \in \emptyset \vdash x \in A}}{(\Rightarrow I)\ \mathcal{T} \vdash x \in \emptyset \Rightarrow x \in A}}{(\forall I)\ \mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A)}$$

$$\dfrac{\dfrac{(Ax)\ \overline{\mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A) \Rightarrow \emptyset \subseteq A} \qquad \mathcal{T} \vdash \ldots}{(\Rightarrow E)\ \mathcal{T} \vdash \forall x(x \in \emptyset \Rightarrow x \in A)}}{\mathcal{T} \vdash \emptyset \subseteq A}$$

... shorter in deduction modulo

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

... shorter in deduction modulo

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \in \emptyset \vdash_{\cong} \bot}(Ax)}{x \in \emptyset \vdash_{\cong} x \in A}(\bot E)}{\vdash_{\cong} x \in \emptyset \Rightarrow x \in A}(\Rightarrow I)}{\vdash_{\cong} \emptyset \subseteq A}(\forall I)$$

$x \in \emptyset \cong \bot$

$\emptyset \subseteq A \cong \ldots$

... shorter in deduction modulo

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \in \emptyset \vdash_{\cong} \bot}(Ax)}{x \in \emptyset \vdash_{\cong} x \in A}(\bot E)}{\vdash_{\cong} x \in \emptyset \Rightarrow x \in A}(\Rightarrow I)}{\vdash_{\cong} \emptyset \subseteq A}(\forall I) \qquad \begin{array}{l} x \in \emptyset \cong \bot \\ \\ \\ \emptyset \subseteq A \cong \ldots \end{array}$$

The proof is shorter but not very informative

A generalization of Natural Deduction

We consider some new rules about predicate symbols:

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

A generalization of Natural Deduction

We consider some new rules about predicate symbols:

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

$$\vdash \emptyset \subseteq A$$

A generalization of Natural Deduction

We consider some new rules about predicate symbols:

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

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

A generalization of Natural Deduction

We consider some new rules about predicate symbols:

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

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

A generalization of Natural Deduction

We consider some new rules about predicate symbols:

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

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

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 $P \longrightarrow \phi$):

- decompose ϕ 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 $P \longrightarrow \phi$):

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

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

Systematic generation of the new inference rules

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

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

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

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

Systematic generation of the new inference rules

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

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

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

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

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

Systematic generation of the new inference rules

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

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

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

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

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

Systematic generation of the new inference rules

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

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

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

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

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

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:

$$(\subseteq E)\ \frac{\begin{array}{c}\vdots\ \mathcal{D}_2\\ \Gamma \vdash t \in X\end{array} \qquad (\subseteq I)\ \dfrac{\begin{array}{c}\vdots\ \mathcal{D}_1\\ \Gamma, x \in X \vdash x \in Y\end{array}}{\Gamma \vdash X \subseteq Y}\ (x \notin \mathcal{FV}(\Gamma))}{\Gamma \vdash t \in Y}$$

Cut elimination

A new notion of cut appears for each defined predicate:

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

reduces to

$$\frac{\dfrac{\vdots\ \mathcal{D}_2 \qquad \ldots \qquad \vdots\ \mathcal{D}_2}{\vdots\ \mathcal{D}_1}}{\Gamma \vdash t \in Y}$$

Cut elimination

A new notion of cut appears for each defined predicate:

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

reduces to

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

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 λ-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 λ-abstraction and store multiple assumptions and witnesses in patterns

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

Proof terms

Definition of proof terms for Generalized Natural Deduction

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

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

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

Proof terms

Definition of proof terms for Generalized Natural Deduction

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

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

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

▶ A collection of new type systems for the ρ-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 ρP
- 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 ρ-cube
  - Conjunction types for structures
  - Generalized Natural Deduction seen as a collection of type systems

Perspectives

- Types
  - Strong normalization in the remaining of the ρ-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 ρ-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 ρ-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 $\cong$:

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

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 $\rightarrow_{\mathcal{R}}$

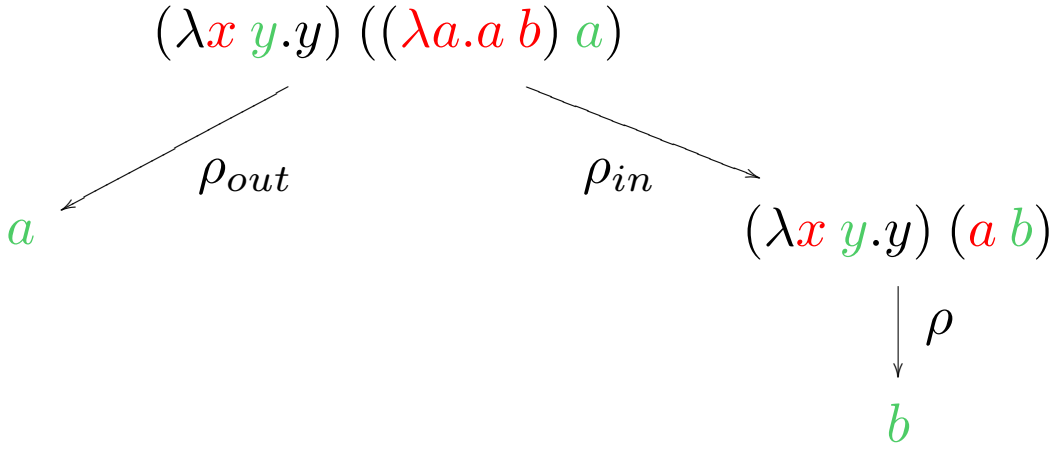
Every deduction rule is considered modulo $\cong$:

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

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

(Non-)Confluence of the ρ-calculus

- *Active* variables are troublesome



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

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} C\,A$

$$A$$

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} C\,A$

$$A \twoheadrightarrow C\,A$$

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} C\,A$

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

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} C\,A$

$$A \twoheadrightarrow C\,A \longrightarrow\!\!\!\!\!\rightarrow (\lambda(d\,z\,z).e)\,(d\,A\,(C\,A)) \downarrow (\lambda(d\,z\,z).e)\,(d\,(C\,A)\,(C\,A))$$

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\rightarrow_{\rho\delta} C\,A$

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

(Non-)Confluence of the ρ-calculus – part II

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

- C such that $C \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} \lambda y.(\lambda(d\,x\,x).e)\,(d\,y\,(C\,y))$
- A such that $A \longmapsto\!\!\!\!\!\rightarrow_{\rho\delta} C\,A$

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

Expressiveness

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

   (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 λ into the ρ.** $\varphi : \lambda \Rightarrow \rho$

   (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 $t_1 \mapsto\!\!\!\!\rightarrow_{\mathcal{R}} t_2$, then $\exists\ A$ such that $A \bullet t_1 \mapsto\!\!\!\!\rightarrow_{\rho\delta} t_2$

Normalization failure

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

$$\dfrac{\dfrac{\Gamma \vdash_\Sigma f : (\alpha \to \alpha) \to \alpha \quad \Gamma \vdash_\Sigma x : \alpha \to \alpha}{\Gamma \vdash_\Sigma f\, x : \alpha} \quad \dfrac{\Gamma \vdash_\Sigma x : \alpha \to \alpha \quad \dfrac{\vdots}{\Gamma \vdash_\Sigma f\, x : \alpha}}{\Gamma \vdash_\Sigma x\,(f\, x) : \alpha}}{\vdash_\Sigma \omega \equiv \lambda f\, x.x\,(f\, x) : \alpha \to \alpha}$$

$$\vdash_\Sigma \omega\,(f\, \omega) : \alpha$$

The relation $\subseteq$ and $first$

$$
\begin{array}{lcll}
f\ \overline{P} & \not\sqsubseteq & \lambda Q.B & \\
f\ \overline{P} & \not\sqsubseteq & g\ \overline{B} & \text{if } \ f \neq g \ \vee \ \exists i, P_i \not\sqsubseteq B_i \\
P & \not\sqsubseteq & (\lambda Q.A)\ B & \text{if } Q \not\sqsubseteq B \ \vee \ P \not\sqsubseteq A
\end{array}
$$

$$
\begin{array}{rcl}
first(A_1, A_2, \ldots, A_n) & \triangleq & X \to ((\mathsf{stk} \to A_n\ X \wr I)\ (\ldots\ (\mathsf{stk} \to A_2\ X \wr I)\ (A_1 X))) \\
first(A_1, A_2, \ldots, A_n)\ B & \mapsto_{\rho\infty} & first(A_2, \ldots, A_n)\ B
\end{array}
$$

Encoding of TRSs

$$
\begin{aligned}
[\![\mathcal{R}]\!] \;=\; & \lambda rec\, z \;.\; \mathit{first} \left(\begin{array}{l} \lambda l_1 \;.\; z\,(rec\, z)\, r_1, \\ \cdots, \\ \lambda a_1\, \overline{x} \;.\; z\,(rec\, z)\, a_1(\overline{z\,(rec\, z)\, x}), \\ \cdots \end{array} \right) \\
& \wr \;\; \lambda Rec\, z \;.\; \mathit{first} \left(\begin{array}{l} \lambda l_1 \;.\; z\,(rec\, z)\, r_1, \\ \cdots, \\ \lambda y.y \end{array} \right)
\end{aligned}
$$

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 λ-calculus

$$[\![x]\!] \triangleq x$$

$$[\![f]\!] \triangleq \lambda x_1 \ldots \lambda x_{\alpha_f}.\ (\lambda z.(z\ x_1 \ldots x_{\alpha_f}))$$

$$[\![f\ B_1 \ldots B_{\alpha_f}]\!] \triangleq \lambda z.(z\ B_1 \ldots B_{\alpha_f})$$

Encoding the P^2TS into λ-calculus

$$
\begin{aligned}
[\![x]\!] &\triangleq x \\
[\![f]\!] &\triangleq \lambda x_1 \ldots \lambda x_{\alpha_f}.\ (\lambda z.(z\ x_1 \ldots x_{\alpha_f})) \\
[\![f\ B_1 \ldots B_{\alpha_f}]\!] &\triangleq \lambda z.(z\ B_1 \ldots B_{\alpha_f}) \\
[\![\lambda(f\ P_1 \ldots P_p).A]\!] &\triangleq \lambda u.(u\ x_\perp \ldots x_\perp [\![\lambda P_1 \ldots \lambda P_p.\lambda x'_{p+1} \ldots \lambda x'_{\alpha_f}.A]\!]\ )
\end{aligned}
$$

Encoding the P^2TS into λ-calculus

$$
\begin{aligned}
[\![x]\!] &\triangleq x \\
[\![f]\!] &\triangleq \lambda x_1 \ldots \lambda x_{\alpha_f}.\ (\lambda z.(z\ x_1 \ldots x_{\alpha_f})) \\
[\![f\ B_1 \ldots B_{\alpha_f}]\!] &\triangleq \lambda z.(z\ B_1 \ldots B_{\alpha_f}) \\
[\![\lambda(f\ P_1 \ldots P_p).A]\!] &\triangleq \lambda u.(u\ x_\perp \ldots x_\perp [\![\lambda P_1 \ldots \lambda P_p.\lambda x'_{p+1} \ldots \lambda x'_{\alpha_f}.A]\!]\ ) \\
[\![\lambda x.A]\!] &\triangleq \lambda x.[\![A]\!] \\
[\![A\ B]\!] &\triangleq [\![A]\!][\![B]\!]
\end{aligned}
$$

Encoding the P^2TS into λ-calculus

$$
\begin{aligned}
[\![x]\!] &\triangleq x \\
[\![f]\!] &\triangleq \lambda x_1 \ldots \lambda x_{\alpha_f}.\ (\lambda z.(z\ x_1 \ldots x_{\alpha_f})) \\
[\![f\ B_1 \ldots B_{\alpha_f}]\!] &\triangleq \lambda z.(z\ B_1 \ldots B_{\alpha_f}) \\
[\![\lambda(f\ P_1 \ldots P_p).A]\!] &\triangleq \lambda u.(u\ x_\perp \ldots x_\perp [\![\lambda P_1 \ldots \lambda P_p.\lambda x'_{p+1} \ldots \lambda x'_{\alpha_f}.A]\!]\ ) \\
[\![\lambda x.A]\!] &\triangleq \lambda x.[\![A]\!] \\
[\![A\ B]\!] &\triangleq [\![A]\!][\![B]\!] \\
[\![A \wr B]\!] &\triangleq \lambda x_1 \ldots \lambda x_\alpha.\ \Big((\lambda z.([\![A]\!] x_1 \ldots x_\alpha))([\![B]\!] x_1 \ldots x_\alpha)\Big)
\end{aligned}
$$

An example of translated term

$$\overbrace{(\lambda y.\Big(\overbrace{(\lambda u.(u(\lambda x.x)))}^{[\![\lambda(f\,x).x]\!]}\,y\Big))\,\Big(\overbrace{(\lambda x_1.\lambda z.(z x_1))}^{[\![f]\!]}\overbrace{(\lambda v.v)}^{[\![a]\!]}\Big)}^{[\![\lambda y.(\lambda(f\,x).x)\,y]\!]}$$

An example of translated term

$$\overbrace{(\lambda y.\Big(\overbrace{(\lambda u.(u(\lambda x.x)))}^{[\![\lambda(f\,x).x]\!]}\,y\Big))\,\Big(\overbrace{(\lambda x_1.\lambda z.(z x_1))}^{[\![f]\!]}\overbrace{(\lambda v.v)}^{[\![a]\!]}\Big)}^{[\![\lambda y.(\lambda(f\,x).x)\,y]\!]}$$

$$\mapsto_\beta \quad \big(\lambda y.(y(\lambda x.x))\big)\Big((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)\Big)$$

An example of translated term

$$
\begin{aligned}
& \overbrace{(\lambda y.(\overbrace{\lambda u.(u(\lambda x.x))}^{[\![\lambda(f\,x).x]\!]}\,y))\,(\overbrace{\lambda x_1.\lambda z.(z x_1)}^{[\![f]\!]}\,\overbrace{\lambda v.v}^{[\![a]\!]})}^{[\![\lambda y.(\lambda(f\,x).x)\,y]\!]} \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)) \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,(\lambda z.(z(\lambda v.v)))
\end{aligned}
$$

An example of translated term

$$
\begin{aligned}
& \overbrace{(\lambda y.(\overbrace{\lambda u.(u(\lambda x.x))}^{[\![\lambda(f\,x).x]\!]}\,y))\,(\overbrace{\lambda x_1.\lambda z.(z x_1)}^{[\![f]\!]}\,\overbrace{\lambda v.v}^{[\![a]\!]})}^{[\![\lambda y.(\lambda(f\,x).x)\,y]\!]} \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)) \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,(\lambda z.(z(\lambda v.v))) \\
\mapsto_\beta \quad & (\lambda z.(z(\lambda v.v)))\,(\lambda x.x)
\end{aligned}
$$

An example of translated term

$$
\begin{aligned}
& \overbrace{(\lambda y.(\overbrace{\lambda u.(u(\lambda x.x))}^{[\![\lambda(f\,x).x]\!]}\,y))\,(\overbrace{\lambda x_1.\lambda z.(z x_1)}^{[\![f]\!]}\,\overbrace{\lambda v.v}^{[\![a]\!]})}^{[\![\lambda y.(\lambda(f\,x).x)\,y]\!]} \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)) \\
\mapsto_\beta \quad & (\lambda y.(y(\lambda x.x)))\,(\lambda z.(z(\lambda v.v))) \\
\mapsto_\beta \quad & (\lambda z.(z(\lambda v.v)))\,(\lambda x.x) \\
\mapsto_\beta \quad & (\lambda x.x)(\lambda v.v)
\end{aligned}
$$

An example of translated term

$$
\begin{array}{ll}
 & \overbrace{(\lambda y.(\lambda \overbrace{(f\ x)}^{[\![\lambda y.(\lambda(f\ x).x)\ y]\!]}.x)\ y)}^{[\![\lambda(f\ x).x]\!]}\ \Big(\overbrace{(\lambda x_1.\lambda z.(z x_1))}^{[\![f]\!]}\overbrace{(\lambda v.v)}^{[\![a]\!]}\Big) \\
\mapsto_\beta & (\lambda y.(y(\lambda x.x)))\Big((\lambda x_1.\lambda z.(z x_1))(\lambda v.v)\Big) \\
\mapsto_\beta & (\lambda y.(y(\lambda x.x)))(\lambda z.(z(\lambda v.v))) \\
\mapsto_\beta & (\lambda z.(z(\lambda v.v)))(\lambda x.x) \\
\mapsto_\beta & (\lambda x.x)(\lambda v.v) \\
\mapsto_\beta & (\lambda v.v) \\
= & [\![a]\!]
\end{array}
$$

The type of a translated constant

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

$$\vdash_{\lambda\omega} [\![f]\!] = \lambda x_1.\lambda z.(z\ x_1) \ : \quad \sigma \to (\sigma \to \beta) \to \beta$$

$$\vdash_{\lambda\omega} [\![f\ B]\!] \ : \qquad\qquad\qquad (\sigma \to \beta) \to \beta$$

Enhanced translation

$$\bigwedge \sigma_1, \ldots, \sigma_\alpha \triangleq \Pi(\beta : *).\big((\sigma_1 \to \ldots \sigma_\alpha \to \beta) \to \beta\big)$$

$$[\![f]\!] \triangleq \lambda x_1.\lambda(\beta : *)\ (\lambda z.(z\ x_1)) : \sigma \to \bigwedge \sigma$$

$$[\![\lambda f\ x.A]\!] \triangleq \lambda u.\big(u\ \tau\ \lambda x.[\![A]\!]\big) : (\bigwedge \sigma) \to \tau$$

where $[\![\Gamma]\!] \vdash_{\lambda\omega} [\![A]\!] : \tau$

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

Use of types depending on types

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

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

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

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

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:

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

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

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

Conservativity

$$(K\ E)\ (P\ I)\ \frac{(Ax)\ \dfrac{}{\vdots} \qquad \dfrac{}{\vdots}}{def \vdash H_1 \quad \ldots \quad def \vdash H_n} \qquad\qquad (P\ E)\ (K\ I)\ \frac{P, \Gamma \vdash P \quad \ldots \quad P, \Gamma \vdash \gamma}{\vdots}$$
$$def \vdash P \qquad\qquad P \vdash def$$

About unsound rules

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

Its associated introduction and elimination rules are

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

About unsound rules

It is well-known that the rewrite rule $R \rightarrow R \Rightarrow \bot$ gives an unsound deduction modulo
Its associated introduction and elimination rules are

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

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

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

Curiosities

- Proof terms with patterns for the usual connectives

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

Curiosities

- Proof terms with patterns for the usual connectives

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

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

  Predicates defined by induction give some natural rules

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