Principal Typings in a Restricted Intersection Type System for Beta Normal Forms with de Bruijn Indices

Daniel L. Ventura[1,2] & Mauricio Ayala-Rincón[1] & Fairouz D. Kamareddine[2]

[1]Grupo de Teoria da Computação - GTC/UnB
Universidade de Brasília - UnB, Brasil
[2]ULTRA Group
Heriot-Watt University, Edinburgh, Scotland

Research supported by the Brazilian Research Council - CNPq

9th International Workshop on Reduction Strategies in Rewriting and Programming
June 28, Brasília, 2009

Talk's Plan

Intersection type discipline

- Introduced by M. Coppo and M. Dezani-Ciancaglini. [CDC78, CDC80]
- It incorporates type polymorphism in a finitary way:

$$\lambda_x.x : (int \to int) \wedge (bool \to bool)$$

- IT called after realisability semantics interpretation of types
- Characterisation of the SN terms of the λ-calculus. [Pot80]
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type discipline

- Introduced by M. Coppo and M. Dezani-Ciancaglini. [CDC78, CDC80]
- It incorporates type polymorphism in a finitary way:

$$\lambda_x.x : (int \rightarrow int) \wedge (bool \rightarrow bool)$$

- IT called after realisability semantics interpretation of types
- Characterisation of the SN terms of the λ-calculus. [Pot80]
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type discipline

- Introduced by M. Coppo and M. Dezani-Ciancaglini. [CDC78, CDC80]
- It incorporates type polymorphism in a finitary way:

$$\lambda_x.x : (int \to int) \wedge (bool \to bool)$$

- IT called after realisability semantics interpretation of types
- Characterisation of the SN terms of the λ-calculus. [Pot80]
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type discipline

- Introduced by M. Coppo and M. Dezani-Ciancaglini. [CDC78, CDC80]
- It incorporates type polymorphism in a finitary way:

$$\lambda_x.x : (int \to int) \wedge (bool \to bool)$$

- IT called after realisability semantics interpretation of types
- Characterisation of the SN terms of the λ-calculus. [Pot80]
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type discipline

- Introduced by M. Coppo and M. Dezani-Ciancaglini. [CDC78, CDC80]
- It incorporates type polymorphism in a finitary way:

$$\lambda_x.x : (int \rightarrow int) \wedge (bool \rightarrow bool)$$

- IT called after realisability semantics interpretation of types
- Characterisation of the SN terms of the λ-calculus. [Pot80]
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Principal typings

Let $\Gamma \vdash M : \tau$ be a type judgement in some type system S

- $\langle \Gamma \vdash \tau \rangle$ is a typing of M in S, written as $M : \langle \Gamma \vdash_S \tau \rangle$.
- $\langle \Gamma \vdash \tau \rangle$ is a **principal typing** (PT) of M if $M : \langle \Gamma \vdash_S \tau \rangle$ and it "represents" any other possible typing of M.
- PT property allows *compositional* type inference

λ-calculus with de Bruijn indices

- Invented by N.G. de Bruijn [dB72].
- Own the same properties as the λ-calculus with names.
- Each α-classe of λ-terms corresponds to a unique term.
- Plays an important role in the implementation of programming languages and theorem provers. [Kam03]
- A variety of IT systems has been studied, usually with variable names and rarely with de Bruijn indices.

Syntax of λ_{dB}

Definition (Set Λ_{dB})

The set of λ_{dB}-terms

Terms $\quad M ::= \underline{n} \mid (M\ M) \mid \lambda.M \quad$ for $n \in \mathbb{N}_* = \mathbb{N} \setminus \{0\}$

Examples

$\lambda.(\lambda.(\underline{1}\ \underline{4}\ \underline{2})\ \underline{1})$

$\lambda.\underline{1} \simeq \lambda x.x \simeq \lambda y.y$

Remark: β and η are defined updating indices accordingly.

Definition (Set Λ_{dB})

The set of λ_{dB}-terms

Terms $\quad M ::= \underline{n} \mid (M\ M) \mid \lambda.M \quad$ for $n \in \mathbb{N}_* = \mathbb{N} \setminus \{0\}$

Examples

$\lambda.(\lambda.(\underline{1}\ \underline{4}\ \underline{2})\ \underline{1})$

$\lambda.\underline{1} \simeq \lambda x.x \simeq \lambda y.y$

Remark: β and η are defined updating indices accordingly.

Syntax of λ_{dB}

Definition (Free indices & closed terms)

1. $FI(M)$ is the set of **free indices** of M, defined by

$$
\begin{array}{rcl}
FI(\underline{n}) & = & \{\underline{n}\} \\
FI(\lambda.M) & = & \{\underline{n-1}, \forall\, \underline{n} \in FI(M), n > 1\} \\
FI(M_1\ M_2) & = & FI(M_1) \cup FI(M_2)
\end{array}
$$

2. M is **closed** if $FI(M) = \varnothing$.
3. $sup(M)$ is the greatest value of a free index in M.

Definition (i-lift)

M^{+i} is defined inductively as

1. $(M_1\, M_2)^{+i} = (M_1^{+i}\, M_2^{+i})$
2. $(\lambda.M_1)^{+i} = \lambda.M_1^{+(i+1)}$
3. $\underline{n}^{+i} = \begin{cases} \underline{n+1}, & \text{if } n > i \\ \underline{n}, & \text{if } n \leq i. \end{cases}$

The **lift** M^{+} of M is its 0-lift.

β-contraction in λ_{dB}

Definition (β-substitution)

The **β-substitution** $\{\underline{n}\,/N\}M$ is defined inductively by

1. $\{\underline{n}\,/N\}(M_1\ M_2) = (\{\underline{n}\,/N\}M_1\ \{\underline{n}\,/N\}M_2)$
2. $\{\underline{n}\,/N\}\lambda.M_1 = \lambda.\{\underline{n+1}\,/N^{+}\}M_1$
3. $\{\underline{n}\,/N\}\underline{m} = \begin{cases} \underline{m-1}\,, & \text{if } m > n \\ N, & \text{if } m = n \\ \underline{m}\,, & \text{if } m < n \end{cases}$

Definition (β-contraction in λ_{dB})

β-contraction in λ_{dB} is defined by

$$(\lambda.M\,N) \rhd_{\beta} \{\underline{1}\,/N\}M$$

Definition (Restricted intersection types and contexts)

1. The **restricted intersection types** are defined by:

$$\mathcal{T} ::= \mathcal{A} \,|\, \mathcal{U} \to \mathcal{T}$$
$$\mathcal{U} ::= \omega \,|\, \mathcal{U} \wedge \mathcal{U} \,|\, \mathcal{T}$$

   $\wedge$ is commutative, associative and has ω as neutral element.

2. The **contexts** are sequences of objects in $\mathcal{U}$, defined by:

$$\Gamma ::= \mathit{nil} \,|\, u.\Gamma, \quad \text{for } u \in \mathcal{U}$$

Our system is a de Bruijn version of a system by Sayag and Mauny in [SM96a, SM96b]

Restricted Intersection types in λ_{dB}

Definition (Restricted intersection types and contexts)

1. The **restricted intersection types** are defined by:

$$\mathcal{T} ::= \mathcal{A} \,|\, \mathcal{U} \to \mathcal{T}$$
$$\mathcal{U} ::= \omega \,|\, \mathcal{U} \wedge \mathcal{U} \,|\, \mathcal{T}$$

   $\wedge$ is commutative, associative and has ω as neutral element.

2. The **contexts** are sequences of objects in $\mathcal{U}$, defined by:

$$\Gamma ::= nil \,|\, u.\Gamma, \quad \text{for } u \in \mathcal{U}$$

Our system is a de Bruijn version of a system by Sayag and Mauny in [SM96a, SM96b]

Restricted intersection types in λ_{dB}

Definition

1. $\omega^{\underline{n}} := \omega.\omega.\cdots.\omega.nil$ such that $|\omega^{\underline{n}}| = n$.
2. The extension of $\wedge$ for contexts:
   - $nil \wedge \Gamma = \Gamma \wedge nil = \Gamma$
   - $(u_1.\Gamma) \wedge (u_2.\Delta) = (u_1 \wedge u_2).(\Gamma \wedge \Delta)$
3. Type substitutions $s : \mathcal{A} \rightarrow \mathcal{T}$ such that:
   - $s(u \rightarrow \tau) = s(u) \rightarrow s(\tau)$
   - $s(\omega) = \omega$ and $s(u \wedge v) = s(u) \wedge s(v)$
   - $s(nil) = nil$ and $s(u.\Gamma) = s(u).s(\Gamma)$

Recall: $M : \langle \Gamma \vdash \tau \rangle$ is used instead of $\Gamma \vdash M : \tau$

Restricted intersection types in λ_{dB}

Definition

1. $\omega^{\underline{n}} := \omega.\omega.\cdots.\omega.nil$ such that $|\omega^{\underline{n}}| = n$.
2. The extension of $\wedge$ for contexts:
   - $nil \wedge \Gamma = \Gamma \wedge nil = \Gamma$
   - $(u_1.\Gamma) \wedge (u_2.\Delta) = (u_1 \wedge u_2).(\Gamma \wedge \Delta)$
3. Type substitutions $s : \mathcal{A} \to \mathcal{T}$ such that:
   - $s(u \to \tau) = s(u) \to s(\tau)$
   - $s(\omega) = \omega$ and $s(u \wedge v) = s(u) \wedge s(v)$
   - $s(nil) = nil$ and $s(u.\Gamma) = s(u).s(\Gamma)$

Recall: $M : \langle \Gamma \vdash \tau \rangle$ is used instead of $\Gamma \vdash M : \tau$

Definition (Typing Rules)

1. System SM is defined by:

$$\frac{\tau \in \mathcal{T}}{\underline{1} : \langle \tau.nil \vdash \tau \rangle}\ \mathrm{var} \qquad \frac{M : \langle nil \vdash \tau \rangle}{\lambda.M : \langle nil \vdash \omega \rightarrow \tau \rangle}\ \rightarrow_i'$$

$$\frac{\underline{n} : \langle \Gamma \vdash \tau \rangle}{\underline{n+1} : \langle \omega.\Gamma \vdash \tau \rangle}\ \mathrm{varn} \qquad \frac{M : \langle u.\Gamma \vdash \tau \rangle}{\lambda.M : \langle \Gamma \vdash u \rightarrow \tau \rangle}\ \rightarrow_i$$

$$\frac{M_1 : \langle \Gamma \vdash \omega \rightarrow \tau \rangle \qquad M_2 : \langle \Delta \vdash \sigma \rangle}{M_1\ M_2 : \langle \Gamma \wedge \Delta \vdash \tau \rangle}\ \rightarrow_e'$$

$$\frac{M_1 : \langle \Gamma \vdash \wedge_{i=1}^{n} \sigma_i \rightarrow \tau \rangle \quad M_2 : \langle \Delta^1 \vdash \sigma_1 \rangle\ \ldots\ M_2 : \langle \Delta^n \vdash \sigma_n \rangle}{M_1\ M_2 : \langle \Gamma \wedge \Delta^1 \wedge \cdots \wedge \Delta^n \vdash \tau \rangle}\ \rightarrow_e$$

2. System SM_r is obtained from SM, taking $\tau = \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \alpha$ in rule var

Type systems properties

Lemma

If $M : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$, *then* $|\Gamma| = sup(M)$ *and* $\forall i$, $\Gamma_i \neq \omega$ *iff* $\underline{i} \in FI(M)$.

Lemma (Generation)

1. *If* $\underline{n} : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$, *then* $\Gamma_n = \tau$.
2. *If* $\underline{n} : \langle \Gamma \vdash_{SM_r} \tau \rangle$, *then* $\tau = \sigma_1 \rightarrow \cdots \rightarrow \sigma_k \rightarrow \alpha$.
3. *If* $\lambda.M : \langle nil \vdash_{SM/SM_r} \tau \rangle$, *then*
   - $\tau = \omega \rightarrow \sigma$ *and* $M : \langle nil \vdash_{SM/SM_r} \sigma \rangle$ *or*
   - $\tau = \wedge_{i=1}^{n} \sigma_i \rightarrow \sigma$ *and* $M : \langle \wedge_{i=1}^{n} \sigma_i . nil \vdash_{SM/SM_r} \sigma \rangle$.
4. *If* $\lambda.M : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$ *and* $|\Gamma| > 0$, *then* $\tau = u \rightarrow \sigma$ *s.t.* $M : \langle u.\Gamma \vdash_{SM/SM_r} \sigma \rangle$.
5. *If* $\underline{n}\ M_1 \cdots M_m : \langle \Gamma \vdash_{SM_r} \tau \rangle$,
   $\Gamma = (\omega^{\underline{n-1}} . \sigma_1 \rightarrow \cdots \rightarrow \sigma_m \rightarrow \tau . nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$,
   $\tau = \sigma_{m+1} \rightarrow \cdots \rightarrow \sigma_{m+k} \rightarrow \alpha$ *and* $M_i : \langle \Gamma^i \vdash_{SM_r} \sigma_i \rangle$.

Lemma

If $M : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$, *then* $|\Gamma| = sup(M)$ *and* $\forall i$, $\Gamma_i \neq \omega$ *iff* $\underline{i} \in FI(M)$.

Lemma (Generation)

1. *If* $\underline{n} : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$, *then* $\Gamma_n = \tau$.
2. *If* $\underline{n} : \langle \Gamma \vdash_{SM_r} \tau \rangle$, *then* $\tau = \sigma_1 \to \cdots \to \sigma_k \to \alpha$.
3. *If* $\lambda.M : \langle nil \vdash_{SM/SM_r} \tau \rangle$, *then*
   - $\tau = \omega \to \sigma$ *and* $M : \langle nil \vdash_{SM/SM_r} \sigma \rangle$ *or*
   - $\tau = \wedge_{i=1}^{n} \sigma_i \to \sigma$ *and* $M : \langle \wedge_{i=1}^{n} \sigma_i . nil \vdash_{SM/SM_r} \sigma \rangle$.
4. *If* $\lambda.M : \langle \Gamma \vdash_{SM/SM_r} \tau \rangle$ *and* $|\Gamma| > 0$, *then* $\tau = u \to \sigma$ *s.t.* $M : \langle u.\Gamma \vdash_{SM/SM_r} \sigma \rangle$.
5. *If* $\underline{n}\ M_1 \cdots M_m : \langle \Gamma \vdash_{SM_r} \tau \rangle$,
   $\Gamma = (\omega^{\underline{n-1}} . \sigma_1 \to \cdots \to \sigma_m \to \tau . nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$,
   $\tau = \sigma_{m+1} \to \cdots \to \sigma_{m+k} \to \alpha$ *and* $M_i : \langle \Gamma^i \vdash_{SM_r} \sigma_i \rangle$.

SM_r and β-normal forms

Theorem

Every β-nf in de Bruijn notation is typeable in system SM_r.

Type inference algorithm for β-nf

Let N be a β-nf

$\mathtt{Infer}(N) =$

Case $N = \underline{n}$

let α be a fresh type variable

return $(\omega \xrightarrow{n-1} .\alpha.nil, \alpha)$

Case $N = \lambda.N_1$

let $(\Gamma^1, \varphi_1) = \mathtt{Infer}(N_1)$

if $(\Gamma^1 = u.\Gamma')$ **then**

return $(\Gamma', u \to \varphi_1)$

else

return $(nil, \omega \to \varphi_1)$

Case $N = \underline{n}\, N_1 \cdots N_m$

let $(\Gamma^1, \varphi_1) = \mathtt{Infer}(N_1)$

$\vdots$

$(\Gamma^m, \varphi_m) = \mathtt{Infer}(N_m)$

α be a fresh type variable

return $((\omega \xrightarrow{n-1} .\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m, \alpha)$

Type inference algorithm

Theorem (Soundness)

If N is a β-nf and $\texttt{Infer}(N) = (\Gamma, \varphi)$, *then* $N : \langle \Gamma \vdash_{SM_r} \varphi \rangle$.

Theorem (Completeness)

If $N : \langle \Gamma \vdash_{SM_r} \varphi \rangle$, *$N$ a β-nf, then for* $(\Gamma', \varphi') = \texttt{Infer}(N)$ *exists a type substitution s such that* $s(\Gamma') = \Gamma$ *and* $s(\varphi') = \varphi$.

Characterisation of principal typings

Definition

1. Let $\mathcal{T}_C$, $\mathcal{T}_{NF}$ and $\mathcal{U}_C$ be defined by:

$$\begin{array}{rcl} \mathcal{T}_C & ::= & \mathcal{A} \,|\, \mathcal{T}_{NF} \to \mathcal{T}_C \\ \mathcal{T}_{NF} & ::= & \mathcal{A} \,|\, \mathcal{U}_C \to \mathcal{T}_{NF} \\ \mathcal{U}_C & ::= & \omega \,|\, \mathcal{U}_C \wedge \mathcal{U}_C \,|\, \mathcal{T}_C \end{array}$$

2. Let $\mathcal{C}$ be the set of contexts Γ with types in $\mathcal{U}_C$

Lemma

$Im(\mathtt{Infer}) \subseteq \mathcal{C} \times \mathcal{T}_{NF}$

Characterisation of principal typings

Definition

1. Let $\mathcal{T}_C$, $\mathcal{T}_{NF}$ and $\mathcal{U}_C$ be defined by:

$$
\begin{array}{rcl}
\mathcal{T}_C & ::= & \mathcal{A} \,|\, \mathcal{T}_{NF} \to \mathcal{T}_C \\
\mathcal{T}_{NF} & ::= & \mathcal{A} \,|\, \mathcal{U}_C \to \mathcal{T}_{NF} \\
\mathcal{U}_C & ::= & \omega \,|\, \mathcal{U}_C \wedge \mathcal{U}_C \,|\, \mathcal{T}_C
\end{array}
$$

2. Let $\mathcal{C}$ be the set of contexts Γ with types in $\mathcal{U}_C$

Lemma

$Im(\mathtt{Infer}) \subseteq \mathcal{C} \times \mathcal{T}_{NF}$

Characterisation of principal typings

Definition (Γ-types)

$$T ::= \Gamma \Rightarrow \varphi \mid \Delta \Rightarrow \qquad \Gamma, \Delta \in \mathcal{C} \text{ and } \varphi \in \mathcal{T}_{NF} \text{ and } |\Delta| > 0$$

Let T^N be obtained from `Infer`(N), for any β-nf N.

Definition (Left subtypes)

The set $L(T)$ is defined by:

- $L(\Gamma \Rightarrow \varphi) = L(\Gamma) \cup L(\varphi)$
- $L(\Gamma) = \cup_{i=0}^{m}\{\Gamma_i\}$, for $\Gamma_i \neq \omega$.
- $L(\alpha) = \emptyset$
- $L(\omega \to \sigma) = L(\sigma)$
- $L(\wedge_{i=1}^{n}\sigma_i \to \sigma) = \{\wedge_{i=1}^{n}\sigma_i\} \cup L(\sigma)$.

Characterisation of principal typings

Definition (Γ-types)

$$T ::= \Gamma \Rightarrow \varphi \mid \Delta \Rightarrow \qquad \Gamma, \Delta \in \mathcal{C} \text{ and } \varphi \in \mathcal{T}_{NF} \text{ and } |\Delta| > 0$$

Let T^N be obtained from $\texttt{Infer}(N)$, for any β-nf N.

Definition (Left subtypes)

The set $L(T)$ is defined by:

- $L(\Gamma \Rightarrow \varphi) = L(\Gamma) \cup L(\varphi)$
- $L(\Gamma) = \cup_{i=0}^{m} \{\Gamma_i\}$, for $\Gamma_i \neq \omega$.
- $L(\alpha) = \emptyset$
- $L(\omega \rightarrow \sigma) = L(\sigma)$
- $L(\wedge_{i=1}^{n} \sigma_i \rightarrow \sigma) = \{\wedge_{i=1}^{n} \sigma_i\} \cup L(\sigma)$.

Characterisation of principal typings

Definition
T is complete if:

- T is closed
- T is finally closed
- T is minimally closed.

Lemma
If N is a β-nf then T^N is complete.

Characterisation of principal typings

Definition

T is complete if:

- T is closed
- T is finally closed
- T is minimally closed.

Lemma

If N is a β-nf then T^N is complete.

Characterisation of principal typings

Definition (Principal)

A complete T is principal if:

- $T = \omega^{\underline{n-1}}.\alpha.nil \Rightarrow \alpha$
- $T = \Gamma \Rightarrow \alpha$ s.t. $\Gamma = (\omega^{\underline{n-1}}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$ and $\forall i$, $\Gamma^i \Rightarrow \varphi_i$ is principal.
- $T = nil \Rightarrow \omega \to \varphi_1$ and $nil \Rightarrow \varphi_1$ is principal.
- $T = \Gamma \Rightarrow u \to \varphi_1$ s.t. either $\Gamma \neq nil$ or $u \neq \omega$ and $u.\Gamma \Rightarrow \varphi_1$ is principal.

Lemma

Let $\mathcal{P} = \{(\Gamma, \varphi) \in \mathcal{C} \times \mathcal{T}_{NF} \mid \Gamma \Rightarrow \varphi \text{ is principal}\}$. *Then*

$$Im(\texttt{Infer}) \subseteq \mathcal{P}$$

Characterisation of principal typings

Definition (Principal)

A complete T is principal if:

- $T = \omega \xrightarrow{n-1} .\alpha.\mathit{nil} \Rightarrow \alpha$
- $T = \Gamma \Rightarrow \alpha$ s.t. $\Gamma = (\omega \xrightarrow{n-1} .\varphi_1 \to \cdots \to \varphi_m \to \alpha.\mathit{nil}) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$ and $\forall i$, $\Gamma^i \Rightarrow \varphi_i$ is principal.
- $T = \mathit{nil} \Rightarrow \omega \to \varphi_1$ and $\mathit{nil} \Rightarrow \varphi_1$ is principal.
- $T = \Gamma \Rightarrow u \to \varphi_1$ s.t. either $\Gamma \neq \mathit{nil}$ or $u \neq \omega$ and $u.\Gamma \Rightarrow \varphi_1$ is principal.

Lemma

Let $\mathcal{P} = \{(\Gamma, \varphi) \in \mathcal{C} \times \mathcal{T}_{NF} \mid \Gamma \Rightarrow \varphi \text{ is principal}\}$. *Then*

$$Im(\texttt{Infer}) \subseteq \mathcal{P}$$

Reconstruction algorithm

$\texttt{Recon}(\Gamma, \varphi) =$

Case (nil, α)

fail

Case (Γ, α)

let $\{(i^1, u_1), \ldots, (i^m, u_m)\} = FO(\alpha, \Gamma)$

if $m = 1$ and $u_1 = (\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \alpha) \wedge u'$ s.t. $\alpha \notin TypeVar(u')$

then if $\forall 1 \leq i \leq n$ there is Γ^i s.t. $\Gamma = \Gamma^i \wedge X^i$

and $\Gamma^i \Rightarrow \tau_i$ is principal

then let $(N_1, \Delta^1) = \texttt{Recon}(\Gamma^1, \tau_1)$

$\vdots$

$(N_n, \Delta^n) = \texttt{Recon}(\Gamma^n, \tau_n)$

$\Delta' = \omega^{\underline{i^1-1}}.\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \alpha.nil$

$\Gamma' = \Delta' \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^n$

$\Gamma = (\Gamma' \wedge \Delta^1 \wedge \cdots \wedge \Delta^n) \wedge \Delta$, s.t. $\Delta \neq \omega^{\underline{j}}, \forall 1 \leq j \leq |\Gamma|$

return $(\underline{i}^1 N_1 \cdots N_n, \Delta)$

else fail

else fail

Reconstruction algorithm(cont.)

Case $(\Gamma, u \to \varphi_1)$

if $\Gamma = \mathit{nil}$ and $u = \omega$

  then let $(N_1, \Delta) = \mathtt{Recon}(\Gamma, \varphi_1)$

  else let $\Gamma' = u.\Gamma$

    $(N_1, \Delta) = \mathtt{Recon}(\Gamma', \varphi_1)$

if $\Delta = \mathit{nil}$

  then return $(\lambda.N_1, \Delta)$

  else fail

Lemma

If $(\Gamma, \varphi) \in \mathcal{P}$ *then:*

1. $\mathtt{Recon}(\Gamma, \varphi) = (N, \mathit{nil})$ *and* N *is a* β*-nf.*
2. $\mathtt{Infer}(N) = (\Gamma, \varphi)$.

Corollary

$\mathcal{P} = \mathit{Im}(\mathtt{Infer})$

Reconstruction algorithm(cont.)

Case $(\Gamma, u \to \varphi_1)$
if $\Gamma = nil$ and $u = \omega$
  then let $(N_1, \Delta) = \mathtt{Recon}(\Gamma, \varphi_1)$
  else let $\Gamma' = u.\Gamma$
    $(N_1, \Delta) = \mathtt{Recon}(\Gamma', \varphi_1)$
if $\Delta = nil$
  then return $(\lambda.N_1, \Delta)$
  else fail

Lemma

If $(\Gamma, \varphi) \in \mathcal{P}$ *then:*

1. $\mathtt{Recon}(\Gamma, \varphi) = (N, nil)$ *and* N *is a* β*-nf.*
2. $\mathtt{Infer}(N) = (\Gamma, \varphi)$.

Corollary

$\mathcal{P} = Im(\mathtt{Infer})$

Reconstruction algorithm(cont.)

Case $(\Gamma, u \to \varphi_1)$
if $\Gamma = \mathit{nil}$ and $u = \omega$
  then let $(N_1, \Delta) = \mathtt{Recon}(\Gamma, \varphi_1)$
  else let $\Gamma' = u.\Gamma$
    $(N_1, \Delta) = \mathtt{Recon}(\Gamma', \varphi_1)$
if $\Delta = \mathit{nil}$
  then return $(\lambda.N_1, \Delta)$
  else fail

Lemma

If $(\Gamma, \varphi) \in \mathcal{P}$ then:

1. $\mathtt{Recon}(\Gamma, \varphi) = (N, \mathit{nil})$ *and N is a β-nf.*
2. $\mathtt{Infer}(N) = (\Gamma, \varphi)$.

Corollary

$\mathcal{P} = \mathit{Im}(\mathtt{Infer})$

Subject reduction failure

We have that

$$\dfrac{\dfrac{\dfrac{\underline{1}:\langle \alpha.\mathit{nil} \vdash \alpha\rangle}{\lambda.\underline{1}:\langle \mathit{nil} \vdash \alpha \to \alpha\rangle}}{\lambda.\lambda.\underline{1}:\langle \mathit{nil} \vdash \omega \to \alpha \to \alpha\rangle} \qquad \dfrac{\dfrac{\underline{1}:\langle \beta.\mathit{nil} \vdash \beta\rangle}{\underline{2}:\langle \omega.\beta.\mathit{nil} \vdash \beta\rangle}}{\underline{3}:\langle \omega.\omega.\beta.\mathit{nil} \vdash \beta\rangle}}{\lambda.\lambda.\underline{1}\ \underline{3}:\langle \omega.\omega.\beta.\mathit{nil} \vdash \alpha \to \alpha\rangle}$$

and that $\lambda.\lambda.\underline{1}\ \underline{3} \rhd_\beta \lambda.\underline{1}$

Conclusion, current and future works

- IT presented types all β-nf for λ_{dB}
- The inference algorithm returns principal typings for all β-nf
- Characterisation for those principal typing was given
- System SM_r is a first step towards IT systems with PT for λ_{dB}
- Extend the results for all SN terms.

Conclusion, current and future works

- IT presented types all β-nf for λ_{dB}
- The inference algorithm returns principal typings for all β-nf
- Characterisation for those principal typing was given
- System SM_r is a first step towards IT systems with PT for λ_{dB}
- Extend the results for all SN terms.

Completeness proof step

Let $\lambda.N : \langle nil \vdash \varphi \rangle$

- Case $\varphi = \omega \rightarrow \varphi_1$ and $N : \langle nil \vdash \varphi_1 \rangle$:
  By IH, $\texttt{Infer}(N) = (\Gamma', \varphi')$ s.t. $s(\varphi') = \varphi_1$ and $s(\Gamma') = nil$.
  Therefore, $\Gamma' = nil$, $\texttt{Infer}(\lambda.N) = (nil, \omega \rightarrow \varphi')$ and $s(\omega \rightarrow \varphi') = \varphi$.
- Case $\varphi = \wedge_{j=1}^{n} \sigma_j \rightarrow \varphi_1$ and $N : \langle \wedge_{j=1}^{n} \sigma_j . nil \vdash \varphi_1 \rangle$: analogous

References

M. Coppo and M. Dezani-Ciancaglini.
A new type assignment for lambda-terms.
Archiv für mathematische logik, 19:139–156, 1978.

M. Coppo and M. Dezani-Ciancaglini.
An Extension of the Basic Functionality Theory for the λ-Calculus.
Notre Dame Journal of Formal Logic, 21(4):685–693, 1980.

N.G. de Bruijn.
Lambda-Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem.
Indag. Mat., 34(5):381–392, 1972.

F. Kamareddine, editor.
Thirty Five Years of Automating Mathematics.
Kluwer, 2003.

G. Pottinger.
A type assignment for the strongly normalizable λ-terms.
In J.P. Seldin and J. R. Hindley (eds), *To H. B. Curry: Essays on combinatory logic, lambda calculus and formalism*, pp. 561–578. Academic Press, 1980.

E. Sayag and M. Mauny.
Characterization of principal type of normal forms in intersetion type system.
In *Proc. of FSTTCS'96*, LNCS, 1180:335–346. Springer, 1996.

E. Sayag and M. Mauny.
A new presentation of the intersection type discipline through principal typings of normal forms.
Tech. rep. RR-2998, INRIA, 1996.

J.B. Wells.
The essence of principal typings.
In 29^{th} *Int.Coll. on Automata, Languages and Programming*, v. 2380 of *LNCS*, pages 913–925. 2002.