Subject Reduction for the λ-Calculus with Intersection Types in de Bruijn Notation

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

Pesquisa financiada pelo
Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq

XV EBL/XIV SLALM, 11-17/05/2008

Talk's Plan

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the development of programing and specification languages.
- Develop more elaborated systems of types is necessary!

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the development of programing and specification languages.
- Develop more elaborated systems of types is necessary!

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the development of programing and specification languages.
- Develop more elaborated systems of types is necessary!

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- It avoids necessity of α-conversion.
- Used by some explicit substitutions calculi.
  (e.g. $\lambda\sigma$, λs_e).

JumpdB

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- It avoids necessity of α-conversion.
- Used by some explicit substitutions calculi.
  (e.g. $\lambda\sigma$, λs_e).

JumpdB

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- It avoids necessity of α-conversion.
- Used by some explicit substitutions calculi.
  (e.g. $\lambda\sigma$, λs_e).

JumpdB

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- It avoids necessity of α-conversion.
- Used by some explicit substitutions calculi.
  (e.g. $\lambda\sigma$, λs_e).

JumpdB

Intersection type disciplines

- Introduced by Coppo & Dezani-Ciancaglini [CDC80] and Sallé [Sal78] in order to provide a characterization of the SN terms of the λ-calculus.
- Used for characterizing evaluation properties of λ-terms.
- It incorporates type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type disciplines

- Introduced by Coppo & Dezani-Ciancaglini [CDC80] and Sallé [Sal78] in order to provide a characterization of the SN terms of the λ-calculus.
- Used for characterizing evaluation properties of λ-terms.
- It incorporates type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type disciplines

- Introduced by Coppo & Dezani-Ciancaglini [CDC80] and Sallé [Sal78] in order to provide a characterization of the SN terms of the λ-calculus.
- Used for characterizing evaluation properties of λ-terms.
- It incorporates type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Intersection type disciplines

- Introduced by Coppo & Dezani-Ciancaglini [CDC80] and Sallé [Sal78] in order to provide a characterization of the SN terms of the λ-calculus.
- Used for characterizing evaluation properties of λ-terms.
- It incorporates type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Syntax of λ_{dB}

Definition (Set Λ_{dB})

The set of λ_{dB}-terms

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

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 (Set Λ_{dB})

The set of λ_{dB}-terms

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

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.

Syntax of λ_{dB}

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.

Syntax of λ_{dB}

Lemma

$$FI(M^{+i}) = \{\, \underline{n} \,|\, \underline{n} \in FI(M), n \leq i\} \cup \{\, \underline{n+1} \,|\, \underline{n} \in FI(M), n > i\}$$

Lemma

1. $sup(M^{+i}) = sup(M)+1$, *if* $sup(M) > i$.
2. $sup(M^{+i}) = sup(M)$, *otherwise.*

β-contraction in λ_{dB}

Definition (β-substitution)

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

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

Definition (β-contraction in λ_{dB})

β-**contraction** in λ_{dB} is defined by

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

β-contraction in λ_{dB}fixme

Lemma

$FI(\{\underline{1}/N\}M) = FI(\lambda.M\ N)$, *if* $\underline{1} \in FI(M)$.
$FI(\{\underline{1}/N\}M) = FI(\lambda.M)$, *otherwise*.

Corollary

$sup(\{\underline{1}/N\}M) \leq sup(\lambda.M\ N)$.

β-reduction in λ_{dB}

Definition (β-reduction in λ_{dB})

β-**reduction** in λ_{dB} is defined by:

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

$$\frac{M_1 \longrightarrow_\beta N_1}{(M_1\ M_2) \longrightarrow_\beta (N_1\ M_2)} \qquad \frac{M_2 \longrightarrow_\beta N_2}{(M_1\ M_2) \longrightarrow_\beta (M_1\ N_2)}$$

β-reduction in λ_{dB}

Theorem (Free indices after β-reduction)

Let $M \longrightarrow_\beta N$:

- $FI(N) \subseteq FI(M)$.

Consequently,

- $sup(N) \leq sup(M)$.

Intersection types in λ_{dB}

Definition (Intersection types and contexts)

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

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

2. $\sqcap$ is commutative, associative and idempotent, where ω is neutral.

Intersection types in λ_{dB}

Definition

1. The **contexts** are sequences of types in $\mathbb{U}$, defined by:

$$\Gamma ::= nil \mid U.\Gamma, \quad \text{for } U \in \mathbb{U}$$

2. $env_\omega^M := \omega.\omega.\cdots.\omega.nil$ such that $|env_\omega^M| = sup(M)$.

3. The extension of $\sqcap$ for contexts is done by
   - $nil \sqcap \Gamma = \Gamma \sqcap nil = \Gamma$
   - $(U_1.\Gamma) \sqcap (U_2.\Delta) = (U_1 \sqcap U_2).(\Gamma \sqcap \Delta)$

Remark: $M : \langle \Gamma \vdash U \rangle$ is used instead of $\Gamma \vdash M : U$

Intersection types in λ_{dB}

Definition

1. The **contexts** are sequences of types in $\mathbb{U}$, defined by:

$$\Gamma ::= nil \mid U.\Gamma, \quad \text{for } U \in \mathbb{U}$$

2. $env_{\omega}^{M} := \omega.\omega.\cdots.\omega.nil$ such that $|env_{\omega}^{M}| = sup(M)$.
3. The extension of $\sqcap$ for contexts is done by
   - $nil \sqcap \Gamma = \Gamma \sqcap nil = \Gamma$
   - $(U_1.\Gamma) \sqcap (U_2.\Delta) = (U_1 \sqcap U_2).(\Gamma \sqcap \Delta)$

Remark: $M : \langle \Gamma \vdash U \rangle$ is used instead of $\Gamma \vdash M : U$

Definition ($\sqcap$ Typing Rules)

For $T \in \mathbb{T}$ and $U \in \mathbb{U}$:

$$\frac{}{\underline{1} : \langle T.nil \vdash T \rangle}\ \text{var} \qquad \frac{M : \langle nil \vdash T \rangle}{\lambda.M : \langle nil \vdash \omega \to T \rangle}\ \to_i'$$

$$\frac{\underline{n} : \langle \Gamma \vdash U \rangle}{\underline{n+1} : \langle \omega.\Gamma \vdash U \rangle}\ \text{varn} \qquad \frac{M_1 : \langle \Gamma \vdash U \to T \rangle \quad M_2 : \langle \Gamma' \vdash U \rangle}{M_1\ M_2 : \langle \Gamma \sqcap \Gamma' \vdash T \rangle}\ \to_e$$

$$\frac{}{M : \langle env_\omega^M \vdash \omega \rangle}\ \omega \qquad \frac{M : \langle \Gamma \vdash U_1 \rangle \quad M : \langle \Gamma \vdash U_2 \rangle}{M : \langle \Gamma \vdash U_1 \sqcap U_2 \rangle}\ \sqcap_i$$

$$\frac{M : \langle U.\Gamma \vdash T \rangle}{\lambda.M : \langle \Gamma \vdash U \to T \rangle}\ \to_i \qquad \frac{M : \langle \Gamma \vdash U \rangle \quad \langle \Gamma \vdash U \rangle \sqsubseteq \langle \Gamma' \vdash U' \rangle}{M : \langle \Gamma' \vdash U' \rangle}\ \sqsubseteq$$

Definition ($\sqsubseteq$)

The binary relation $\sqsubseteq$ is given by the following rules:

$$\frac{}{\Phi \sqsubseteq \Phi}\ \text{ref} \qquad \frac{\Phi_1 \sqsubseteq \Phi_2 \quad \Phi_2 \sqsubseteq \Phi_3}{\Phi_1 \sqsubseteq \Phi_3}\ \text{tr}$$

$$\frac{}{U_1 \sqcap U_2 \sqsubseteq U_1}\ \sqcap_e \qquad \frac{U_1 \sqsubseteq V_1 \quad U_2 \sqsubseteq V_2}{U_1 \sqcap U_2 \sqsubseteq V_1 \sqcap V_2}\ \sqcap$$

$$\frac{U_2 \sqsubseteq U_1 \quad T_1 \sqsubseteq T_2}{U_1 \rightarrow T_1 \sqsubseteq U_2 \rightarrow T_2}\ \rightarrow \qquad \frac{U_1 \sqsubseteq U_2}{\Gamma_{\leq i}.U_1.\Gamma_{>i} \sqsubseteq \Gamma_{\leq i}.U_2.\Gamma_{>i}}\ \sqsubseteq_c$$

$$\frac{U_1 \sqsubseteq U_2 \quad \Gamma' \sqsubseteq \Gamma}{\langle \Gamma \vdash U_1 \rangle \sqsubseteq \langle \Gamma' \vdash U_2 \rangle}\ \sqsubseteq_{\langle\rangle}$$

Basic properties

Lemma

1. *If* $U \in \mathbb{U}$, *then* $U = \omega$ *or* $U = \sqcap_{i=1}^{n} T_i$ *for* $n \geq 1$ *and* $T_i \in \mathbb{T}$.
2. $U \sqsubseteq \omega$.
3. *If* $\omega \sqsubseteq U$, *then* $U = \omega$.

Basic properties

Lemma (Properties of $\sqcap$, $\sqsubseteq$, typings and contexts)

1. *If* $\Gamma \sqsubseteq \Gamma'$ *and* $U \sqsubseteq U'$, *then* $U.\Gamma \sqsubseteq U'.\Gamma'$.
2. $\Gamma \sqsubseteq \Gamma'$ *iff* $|\Gamma| = |\Gamma'| = m$ *and, if* $m > 0$ *then* $\forall i,\ \Gamma_i \sqsubseteq \Gamma'_i$.
3. *If* $|\Gamma| = sup(M)$, *then* $\Gamma \sqsubseteq env_\omega^M$.
4. *If* $env_\omega^M \sqsubseteq \Gamma$, *then* $\Gamma = env_\omega^M$.
5. $\langle \Gamma \vdash U \rangle \sqsubseteq \langle \Gamma' \vdash U' \rangle$ *iff* $\Gamma' \sqsubseteq \Gamma$ *and* $U \sqsubseteq U'$.
6. *If* $\Gamma \sqsubseteq \Gamma'$ *and* $\Delta \sqsubseteq \Delta'$, *then* $\Gamma \sqcap \Delta \sqsubseteq \Gamma' \sqcap \Delta'$.

More properties

Lemma

1. *If* $M : \langle \Gamma \vdash U \rangle$, *then* $|\Gamma| = sup(M)$.
2. *For every* Γ *and* M *such that* $|\Gamma| = sup(M)$, *one has* $M : \langle \Gamma \vdash \omega \rangle$.

Lemma (derivable rules)

1. $$\frac{M : \langle \Gamma \vdash U_1 \rangle \quad M : \langle \Delta \vdash U_2 \rangle}{M : \langle \Gamma \sqcap \Delta \vdash U_1 \sqcap U_2 \rangle} \ \sqcap'_i$$
2. $$\frac{}{\underline{1} : \langle U.nil \vdash U \rangle} \ \mathrm{var}'$$

Subject reduction for λ_{dB} with $\sqcap$ types

Lemma (Generation)

1. *If* $\underline{n} : \langle \Gamma \vdash U \rangle$, *then* $\Gamma_{n} = V$ *where* $V \sqsubseteq U$.
2. *Let* $\lambda.M : \langle \Gamma \vdash U \rangle$*:*
   - $U = \omega$ *or* $U = \sqcap_{i=1}^{k} (V_i \to T_i)$
     where $k \geq 1$ *and* $\forall i,\ M : \langle V_i.\Gamma \vdash T_i \rangle$*, if* $sup(M) > 0$.
   - $U = \omega$ *or* $U = \sqcap_{i=1}^{k} (V_i \to T_i)$
     where $k \geq 1$ *and* $\forall i,\ M : \langle nil \vdash T_i \rangle$*, otherwise.*

Changes in typings for lifting and β-substitution

Lemma (Typings for lifted terms)

If $M : \langle \Gamma \vdash U \rangle$ *and* $0 \leq i < sup(M)$, *then* $M^{+i} : \langle \Gamma_{\leq i} . \omega . \Gamma_{>i} \vdash U \rangle$

Lemma (Typings for β-substitution)

Let $M : \langle \Gamma \vdash U \rangle$, *for* $sup(M) > 0$, *and* $N : \langle \Delta \vdash \Gamma_i \rangle$:

1. $\{\underline{i} / N\} M : \langle (\Gamma_{<i} . \Gamma_{>i}) \sqcap \Delta \vdash U \rangle$,
   if $\underline{i} \in FI(M)$ *and* $sup(N) \geq i - 1$.
2. $\{\underline{i} / N\} M : \langle \Gamma_{<i} . \Gamma_{>i} \vdash U \rangle$,
   if $\underline{i} \notin FI(M)$.

Subject Reduction

Definition (Restriction of contexts)

$$\Gamma\rfloor_M = \Gamma_{\leq sup(M)} \,.\, nil$$

Theorem (SR for β-contraction)

If $(\lambda.M\ N) : \langle \Gamma \vdash U \rangle$ *then* $\{\underline{1}/N\}M : \langle \Gamma\rfloor_{\{\underline{1}/N\}M} \vdash U \rangle$

Subject Reduction

Theorem (Subject Reduction in λ_{dB})

If $M : \langle \Gamma \vdash U \rangle$ and $M \longrightarrow_\beta N$, then $N : \langle \Gamma\downarrow_N \vdash U \rangle$.

Conclusion, current and future works

- λ-calculus in de Bruijn notation with a system of intersection types has been proved to preserve subject reduction.
- This is the first step towards the construction of adequate explicit substitutions calculi in de Bruijn notation using intersection type.
- *Principal typings property has to be guaranteed because this property supports the possibility of true separate compilation and compositional software analysis [Wel02].*

Conclusion, current and future works

- λ-calculus in de Bruijn notation with a system of intersection types has been proved to preserve subject reduction.
- This is the first step towards the construction of adequate explicit substitutions calculi in de Bruijn notation using intersection type.
- *Principal typings property has to be guaranteed because this property supports the possibility of true separate compilation and compositional software analysis [Wel02].*

Conclusion, current and future works

- λ-calculus in de Bruijn notation with a system of intersection types has been proved to preserve subject reduction.
- This is the first step towards the construction of adequate explicit substitutions calculi in de Bruijn notation using intersection type.
- *Principal typings property has to be guaranteed because this property supports the possibility of true separate compilation and compositional software analysis [Wel02].*

References

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 and K. Nour.
A completeness result for a realisability semantics for an intersection type system
Annals of Pure and Applied Logic , 146:180–198. 2007.

P. Sallé.
Une extension de la théorie des types en lambda-calcul.
In 5^{th} *Int. Conf. on Automata, Languages and Programing*, v. 62 of *LNCS*, pages 398–410. 1978.

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.