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

5^o Sem. Informal(+ Formal!) do GTC, 6-7/12/2007

Universidade de Brasília

Talk's Plan

Universidade de Brasília

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the develpment of programing and specification languages.
- Elaborated systems of types are necessary!

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the develpment of programing and specification languages.
- Elaborated systems of types are necessary!

Universidade de Brasília

Motivation: programs & types

- Nowadays it is well known the relation between programs and types.
- λ-calculus is the theoretical framework in the develpment of programing and specification languages.
- Elaborated systems of types are necessary!

Motivation: λ-calculus (with names)

TERMS $\quad a ::= x \mid (a\ a) \mid \lambda x.a$

- Basic Operators
  - $(a\ b)$ APLICATION
  - $\lambda x.a$ ABSTRACTION

Universidade de Brasília

Rewriting rules of the λ-calculus

- α-conversion

$$\lambda x.a \rightarrow \lambda y.[x/y]a$$

- β-contraction

$$(\lambda x.a\ b) \rightarrow [x/b]a$$

- η-contraction

$$\lambda x.(a\ x) \rightarrow a, \text{ if } x \notin FV(a)$$

Substitution is a meta-operation! JumpEx

Universidade de Brasília

Rewriting rules of the λ-calculus

- α-conversion

$$\lambda x.a \rightarrow \lambda y.[x/y]a$$

- β-contraction

$$(\lambda x.a\ b) \rightarrow [x/b]a$$

- η-contraction

$$\lambda x.(a\ x) \rightarrow a, \text{ if } x \notin FV(a)$$

Substitution is a meta-operation! JumpEx

Universidade de Brasília

Rewriting rules of the λ-calculus

- α-conversion

$$\lambda x.a \longrightarrow \lambda y.[x/y]a$$

- β-contraction

$$(\lambda x.a\ b) \longrightarrow [x/b]a$$

- η-contraction

$$\lambda x.(a\ x) \longrightarrow a, \text{ if } x \notin FV(a)$$

Substitution is a meta-operation! JumpEx

Motivation: examples with the λ-calculus

- (α) $\quad \lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw).$
- (α) $\quad \lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda z.(\lambda y.(z\underline{z}y)yz)$ Wrong!
- (β) $\quad (\lambda x.(\lambda y.(yx))\ y) \to_\beta \lambda y.(y\underline{y})$ Wrong!

  $(\lambda x.(\lambda y.(yx))\ y) \to_\beta \lambda z.(zy)$ Correct!

Motivation: examples with the λ-calculus

- (α) $\lambda x.(\lambda y.(xzy)yx) \rightarrow_{\alpha} \lambda w.(\lambda y.(wzy)yw)$.
- (α) $\lambda x.(\lambda y.(xzy)yx) \rightarrow_{\alpha} \lambda z.(\lambda y.(z\underline{z}y)yz)$ Wrong!
- (β) $(\lambda x.(\lambda y.(yx))\ y) \rightarrow_{\beta} \lambda y.(y\underline{y})$ Wrong!

  $(\lambda x.(\lambda y.(yx))\ y) \rightarrow_{\beta} \lambda z.(zy)$ Correct!

Motivation: examples with the λ-calculus

- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw)$.
- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda z.(\lambda y.(z\underline{z}y)yz)$ Wrong!
- (β) $(\lambda x.(\lambda y.(yx))\ y) \to_\beta \lambda y.(y\underline{y})$ Wrong!

  $(\lambda x.(\lambda y.(yx))\ y) \to_\beta \lambda z.(zy)$ Correct!

Universidade de Brasília

Motivation: examples with the λ-calculus

$(\lambda_x.x \;\; \lambda_x.x) \rightarrow_\beta \lambda_x.x$ self-aplication

$(\lambda_x.(x \;\; x) \;\; \lambda_x.(x \;\; x)) \rightarrow_\beta (\lambda_x.(x \;\; x) \;\; \lambda_x.(x \;\; x))$ self-reproduction

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- Avoids necessity of α-conversion.
- Our preferred initial approach towards making explicit substitutions.

JumpdB

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- Avoids necessity of α-conversion.
- Our preferred initial approach towards making explicit substitutions.

JumpdB

Universidade de Brasília

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- Avoids necessity of α-conversion.
- Our preferred initial approach towards making explicit substitutions.

JumpdB

Universidade de Brasília

λ-calculus in de Bruijn notation

- Invented by Nicolaas Govert de Bruijn [dB72].
- Own the same properties than the λ-calculus with names.
- Avoids necessity of α-conversion.
- Our preferred initial approach towards making explicit substitutions.

JumpdB

Historia

-  Nicolaas Govert de Bruijn (1918-). Matemático holandés lider del Projecto Automath.
- Projecto Automath iniciado en 1967. Primer proyecto que uso tecnología computacional para mecanizar el razonamento matemático:

  Especificación y verificación del libro-texto de  (1877-1938) Edmund Landau's *Grundlagen der Analyses*, Leipzig 1930.

Historia

- 

  http://automath.webhop.net/
- Automath es considerado predecesor de asistentes de demostración modernos: Coq, Nurpl, Isabelle, ...
- [Kam03], [NGdV94], etc.

Historia

- En el proyecto Automath de Bruijn desarrollo la primera formalización de una versión del cálculo λ con un tratamiento explícito de la operación de substitución [dB78]

- 

  N.G. de Bruijn was a well established mathematician before deciding in 1967 at the age of 49 to work on a new direction related to Automating Mathematics. In the 1960s he became fascinated by the new computer technology and decided to start the new Automath project where he could check, with the help of the computer, the correctness of books of mathematics. Through his work on Automath, de Bruijn started a revolution in using the computer for verification, and since, we have seen more and more proof-checking and theorem-proving systems.

  Fairouz D. Kamareddine

Historia

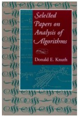
- En el proyecto Automath de Bruijn desarrollo la primera formalización de una versión del cálculo λ con un tratamiento explícito de la operación de substitución [dB78]
- 

N.G. de Bruijn was a well established mathematician before deciding in 1967 at the age of 49 to work on a new direction related to Automating Mathematics. In the 1960s he became fascinated by the new computer technology and decided to start the new Automath project where he could check, with the help of the computer, the correctness of books of mathematics. Through his work on Automath, de Bruijn started a revolution in using the computer for verification, and since, we have seen more and more proof-checking and theorem-proving systems.

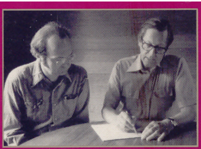
Fairouz D. Kamareddine

Historia

- La influencia de N.G. De Bruijn en computación no se restringe a Automath:



"Selected Papers on Analysis of Algorithms" (CSLI, 2000)
Donal Knuth dedica su libro a su *mentor* de Bruijn.

- 

Knuth with his mentor, N. G. de Bruijn, in 1977. Photo courtesy of Jill Knuth

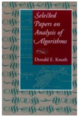
... I'm dedicating this book to N.G. "Dick" de Bruijn because his influence can be felt on every page. Ever since the 1960s he has been my chief mentor, the main person who would answer my questions when I was stuck on a problem that I had not been taught how to solve. I originally wrote Chapter 26 for his $(3 \cdot 4 \cdot 5)$th birthday; now he is 3^4 years young as I gratefully present him with this book.

Donald E. Knuth

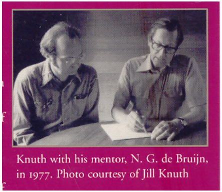
Universidade de Brasília

Historia

- La influencia de N.G. De Bruijn en computación no se restringe a Automath:



“Selected Papers on Analysis of Algorithms” (CSLI, 2000)
Donal Knuth dedica su libro a su *mentor* de Bruijn.

- 

Knuth with his mentor, N. G. de Bruijn, in 1977. Photo courtesy of Jill Knuth

... I'm dedicating this book to N.G. “Dick” de Bruijn because his influence can be felt on every page.
Ever since the 1960s he has been my chief mentor, the main person who would answer my questions when I was stuck on a problem that I had not been taught how to solve.
I originally wrote Chapter 26 for his $(3 \cdot 4 \cdot 5)$th birthday; now he is 3^4 years young as I gratefully present him with this book.

Donald E. Knuth

Universidade de Brasília

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.
- Incorporate 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.
- Incorporate type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings*.

Universidade de Brasília

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.
- Incorporate 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.
- Incorporate type polymorphism in a finitary way (listed instead quantified)
- Some problems arise such as the necessity for a practical treatment of *principal typings.*

Universidade de Brasília

Syntax of λ_{dB}

Definition (Set Λ_{dB})

The syntax of λ_{dB}-calculus. **The set of λ_{dB}-terms**, denoted as Λ_{dB}, is defined inductively as

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

Syntax of λ_{dB}

Examples

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

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

β and η are defined updating indices adequately.

Syntax of λ_{dB}

Definition (Free indices & closed terms)

1. For $M \in \Lambda_{dB}$, let the set of **free indices** of M, denoted as $FI(M)$, be defined by

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

2. A term M is called **closed** if $FI(M) \equiv \emptyset$.
3. The greatest value of a free index in M, denoted as $sup(M)$, is defined as 0, if $FI(M) \equiv \emptyset$, and n such that $\underline{n} \in FI(M)$ and $n \geq i$, $\forall\, \underline{i} \in FI(M)$, otherwise.

Syntax of λ_{dB}

Lemma

1. $sup(M_1\ M_2) = max(sup(M_1), sup(M_2))$
2. *If* $sup(M) = 0$, *then* $sup(\lambda.M) = 0$. *Otherwise,* $sup(\lambda.M) = sup(M) - 1$.

Syntax of λ_{dB}

Definition (i-lift)

Let $M \in \Lambda_{dB}$ and $i \in \mathbb{N}$. The **i-lift** of M, denoted as 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** of a term M is its 0-lift, denoted as M^{+}.
Intuitively, the lift of M corresponds to an increment by 1 of all free indices occurring in M.

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

If $i \geq sup(M)$, then $M^{+i} \equiv M$.

Lemma

1. *If $sup(M) > i$, then $sup(M^{+i}) = sup(M)+1$.*
2. *Otherwise, $sup(M^{+i}) = sup(M)$.*

β-contraction in λ_{dB}

Definition (β-substitution)

Let $m, n \in \mathbb{N}^*$. The β-**substitution** for free occurrences of $\underline{n}$ in $M \in \Lambda_{dB}$ by term N, denoted as $\{\underline{n}/N\}M$, 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 $(\lambda.M\ N) \rightarrow_\beta \{\underline{1}/N\}M$.

β-contraction in λ_{dB}

Lemma (Free indices after β-substitution)

1. *If* $\underline{i} \notin FI(M)$, *then* $FI(\{\underline{i}/N\}M) = \{\underline{n} \mid \underline{n} \in FI(M), n < i\} \cup \{\underline{n-1} \mid \underline{n} \in FI(M), n > i\}$.
2. *Otherwise,* $FI(\{\underline{i}/N\}M) = FI(N) \cup \{\underline{n} \mid \underline{n} \in FI(M), n < i\} \cup \{\underline{n-1} \mid \underline{n} \in FI(M), n > i\}$.

Corollary

If $\underline{1} \in FI(M)$, *then* $FI(\{\underline{1}/N\}M) = FI(\lambda.M\ N)$. *Otherwise,* $FI(\{\underline{1}/N\}M) = FI(\lambda.M)$.

β-contraction in λ_{dB}

Lemma

If $i > sup(M)$, then $\{\underline{i}/N\}M \equiv M$.

Lemma

Let M be a term such that $sup(M) = m$:

1. *If $i < m$ and $\underline{i} \notin FI(M)$, then $sup(\{\underline{i}/N\}M) = m-1$.*
2. *If $i > m$, then $sup(\{\underline{i}/N\}M) = m$.*
3. *Suppose $\underline{i} \in FI(M)$. If $FI(M) = \{\underline{i}\}$, then $sup(\{\underline{i}/N\}M) = sup(N)$. Otherwise, $sup(\{\underline{i}/N\}M) = max(sup(N), m-1)$.*

β-reduction in λ_{dB}

Lemma

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

Definition (β-reduction in λ_{dB})

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

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

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

β-reduction in λ_{dB}

Theorem (Preservation of 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 the following grammars:

$$\mathbb{T} ::= \mathcal{A} \mid \mathbb{U} \rightarrow \mathbb{T}$$

$$\mathbb{U} ::= \omega \mid \mathbb{U} \sqcap \mathbb{U} \mid \mathbb{T}$$

The types are quotiented by taking $\sqcap$ to be commutative, associative, idempotent and to have ω as neutral.

2. The **contexts** are ordered lists of types $U \in \mathbb{U}$, defined by:

$$\Gamma ::= nil \mid U.\Gamma$$

Intersection types in λ_{dB}

- $env_{\omega}^{M} := \omega.\omega.\cdots.\omega.nil$ such that $|env_{\omega}^{M}| = sup(M)$.
- The extension of $\sqcap$ for contexts is done by $nil \sqcap \Gamma = \Gamma \sqcap nil = \Gamma$ and $(U_1.\Gamma) \sqcap (U_2.\Delta) = (U_1 \sqcap U_2).(\Gamma \sqcap \Delta)$.
- Hence, $\sqcap$ is commutative, associative and idempotent on contexts.

Intersection types in λ_{dB}

- $env^M_\omega := \omega.\omega.\cdots.\omega.nil$ such that $|env^M_\omega| = sup(M)$.
- The extension of $\sqcap$ for contexts is done by
  $nil \sqcap \Gamma = \Gamma \sqcap nil = \Gamma$ and
  $(U_1.\Gamma) \sqcap (U_2.\Delta) = (U_1 \sqcap U_2).(\Gamma \sqcap \Delta)$.
- Hence, $\sqcap$ is commutative, associative and idempotent on contexts.

Intersection types in λ_{dB}

- $env_{\omega}^{M} := \omega.\omega.\cdots.\omega.nil$ such that $|env_{\omega}^{M}| = sup(M)$.
- The extension of $\sqcap$ for contexts is done by $nil \sqcap \Gamma = \Gamma \sqcap nil = \Gamma$ and $(U_1.\Gamma) \sqcap (U_2.\Delta) = (U_1 \sqcap U_2).(\Gamma \sqcap \Delta)$.
- Hence, $\sqcap$ is commutative, associative and idempotent on contexts.

Properties of the extension of $\sqcap$ over contexts

Lemma ($\sqcap$ over contexts: properties)

Let Γ and Δ be contexts, where neither Γ nor Δ are nil:

1. *If $|\Gamma| \geq sup(M)$, then $\Gamma \sqcap env_{\omega}^{M} = \Gamma$*
2. $\Gamma \sqcap \Delta = (\Gamma_1 \sqcap \Delta_1).(\Gamma_{>1} \sqcap \Delta_{>1})$
3. *If $i \leq |\Gamma|, |\Delta|$, then $(\Gamma \sqcap \Delta)_i = \Gamma_i \sqcap \Delta_i$.*
4. $(\Gamma \sqcap \Delta)_{<i} = \Gamma_{<i} \sqcap \Delta_{<i}$ *and* $(\Gamma \sqcap \Delta)_{>i} = \Gamma_{>i} \sqcap \Delta_{>i}$. *The same for* $(\Gamma \sqcap \Delta)_{\leq i}$ *and* $(\Gamma \sqcap \Delta)_{\geq i}$.
5. $|\Gamma \sqcap \Delta| = max(|\Gamma|, |\Delta|)$.

Universidade de Brasília

Definition ($\sqcap$ Typing Rules)

The typing rules are the following:

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

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

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

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

Universidade de Brasília

Definition ($\sqsubseteq$)

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

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

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

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

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

Basic properties

Lemma

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

Universidade de Brasília

Lemma (Properties of $\sqcap$ and $\sqsubseteq$)

Let $V \neq \omega$.

1. *If $U \sqsubseteq V$, then $U = \sqcap_{j=1}^{k} T_j$, $V = \sqcap_{i=1}^{p} T'_i$ where $p, k \geq 1$, $\forall 1 \leq j \leq k$, $1 \leq i \leq p$, $T_j, T'_i \in \mathbb{T}$, and $\forall 1 \leq i \leq p$, $\exists 1 \leq j \leq k$ such that $T_j \sqsubseteq T'_i$.*
2. *If $U \sqsubseteq V' \sqcap a$, then $U = U' \sqcap a$ and $U' \sqsubseteq V'$.*
3. *Let $p, k \geq 1$. If $\sqcap_{j=1}^{k}(U_j \to T_j) \sqsubseteq \sqcap_{i=1}^{p}(U'_i \to T'_i)$, then $\forall 1 \leq i \leq p$, $\exists 1 \leq j \leq k$ such that $U'_i \sqsubseteq U_j$ and $T_j \sqsubseteq T'_i$.*
4. *If $U \to T \sqsubseteq V$, then $V = \sqcap_{i=1}^{p}(U_i \to T_i)$ where $p \geq 1$ and $\forall 1 \leq i \leq p$, $U_i \sqsubseteq U$ and $T \sqsubseteq T_i$.*
5. *If $\sqcap_{j=1}^{k}(U_j \to T_j) \sqsubseteq V$ where $k \geq 1$, then $V = \sqcap_{i=1}^{p}(U'_i \to T'_i)$ where $p \geq 1$ and $\forall 1 \leq i \leq p$, $\exists 1 \leq j \leq k$ such that $U'_i \sqsubseteq U_j$ and $T_j \sqsubseteq T'_i$.*

Brasília

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* $\forall 1 \leq i \leq m$, $\Gamma_i \sqsubseteq \Gamma'_i$.
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* $|\Gamma| = sup(M)$, *we have* $M:\langle\Gamma \vdash \omega\rangle$.

Lemma (Typings intersection)

1. *The rule* $\dfrac{M:\langle\Gamma \vdash U_1\rangle \quad M:\langle\Delta \vdash U_2\rangle}{M:\langle\Gamma \sqcap \Delta \vdash U_1 \sqcap U_2\rangle} \sqcap'_i$ *is derivable.*
2. *The rule* $\dfrac{}{\underline{1}:\langle U.nil \vdash U\rangle}\ \mathrm{var}'$ *is derivable.*

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. *If* $\lambda.M : \langle \Gamma \vdash U \rangle$ *and* $sup(M) > 0$, *then* $U = \omega$ *or* $U = \sqcap_{i=1}^{k}(V_i \to T_i)$ *where* $k \geq 1$ *and* $\forall 1 \leq i \leq k$, $M : \langle V_i.\Gamma \vdash T_i \rangle$.
3. *If* $\lambda.M : \langle \Gamma \vdash U \rangle$ *and* $sup(M) = 0$, *then* $\Gamma = nil$, $U = \omega$ *or* $U = \sqcap_{i=1}^{k}(V_i \to T_i)$ *where* $k \geq 1$ *and* $\forall 1 \leq i \leq k$, $M : \langle nil \vdash T_i \rangle$.

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

Subject Reduction

Definition (Restriction of contexts)

Let M be a term and $sup(M) = m$. For a context Γ, let $\Gamma\downharpoonright_M$ be the restriction of Γ to $FI(M)$, given by $\Gamma_{\leq m}.nil$.

Lemma

1. *If* $sup(N) \leq sup(M)$, *then* $env_\omega^M\downharpoonright_N = env_\omega^N$.
2. *If* $|\Gamma| \leq sup(M)$, *then* $(\Gamma \sqcap \Delta)\downharpoonright_M = \Gamma \sqcap \Delta\downharpoonright_M$.
3. *If* $sup(N) > 0$, *then* $(U.\Gamma)\downharpoonright_N = U.\Gamma\downharpoonright_{(\lambda.N)}$.

Universidade de Brasília

Subject Reduction

Theorem (SR for β-contraction)

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

Theorem (Subject Reduction in λ_{dB})

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

Conclusion, current and future work

- 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 with intersection type discipline.
- *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 work

- 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 with intersection type discipline.
- *Principal typings property has to be guaranteed because this property supports the possibility of true separate compilation and compositional software analysis [Wel02].*

Universidade de Brasília

Conclusion, current and future work

- 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 with intersection type discipline.
- *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.

N.G. de Bruijn.
A namefree lambda calculus with facilities for internal definition of expressions and segments.
T.H.-Report 78-WSK-03, Technische Hogeschool Eindhoven, Nederland, 1978.

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

R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer.
Selected papers on Automath. North-Holland, 1994.

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.

Universidade de Brasília