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

Daniel Lima Ventura[1]*and Mauricio Ayala-Rincón[1]†and Fairouz Kamareddine[2]

[1]Grupo de Teoria da Computação, Dep. de Matemática
Universidade de Brasília, Brasília D.F., Brasil

[2] School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, Scotland

{ventura,ayala}@mat.unb.br, fairouz@macs.hw.ac.uk

July 28, 2009

Abstract

The λ-calculus with de Bruijn indices assembles each α-class of λ-terms in a unique term, using indices instead of variable names. Intersection types provide finitary type polymorphism and can characterise normalisable λ-terms, that is a term is normalisable if and only if it is typeable. To be closer to computations and to simplify the formalisation of the atomic operations involved in β-contractions several calculi of explicit substitution were developed and some of them are written with de Bruijn indices. Versions of explicit substitutions calculi without types and with simple type systems are well investigated in contrast to versions with more elaborated type systems such as intersection types. In previous work, we introduced a de Bruijn version of the λ-calculus with an intersection type system and proved it preserves the subject reduction, a basic type system property. In this paper a version with de Bruijn indices of an intersection type system originally introduced to characterise principal typings for β-normal forms (β-nf for short) is presented. We present the characterisation in this new system and the corresponding versions for the type inference and the reconstruction of normal forms from principal typings algorithms. We briefly discuss about the failure of the subject reduction property and some possible solutions for it.

*Supported by a PhD scholarship at the Universidade de Brasília.

†Supported by the Fundação de Apoio à Pesquisa do Distrito Federal [FAPDF 8-004/2007]

1 Introduction

The λ-calculus à la de Bruijn [dB72] was introduced by the Dutch mathematician N.G. de Bruijn in the context of the project Automath [NGdV94] and has been adopted for several calculi of explicit substitutions ever since (e.g. [dB78], [ACCL91], [KR95]). Term variables in the λ-calculus à la de Bruijn are represented by indices instead of names, assembling each α-class of terms in the λ-calculus [Bar84] in a unique term with de Bruijn indices, thus turning it more *"machine-friendly"* than its counterpart. Those calculi with de Bruijn indices have been investigated for both type free and simply typed versions but to the best of our knowledge there is no work on more elaborated type systems such as intersection types.

Intersection types were introduced to provide a characterisation of strongly normalising λ-terms [CDC78, CDC80, Pot80]. In programming, the intersection type discipline is of interest because λ-terms corresponding to correct programs not typeable in the standard Curry type assignment system [CF58], or in extensions allowing some sort of polymorphism as the one present in ML [Mil78], are typeable with intersection types. In [VAK08] an intersection type system for the λ-calculus with de Bruijn indices was introduced, based on the type system given in [KN07], and proved to satisfy the subject reduction property (SR for short); that is the property of preserving types under β-reduction: whenever $\Gamma \vdash M : \sigma$ and M β-reduces into N, $\Gamma \vdash N : \sigma$.

A relevant problem in type theory is whether the system has principal typings (PT for short), which means that for any typeable term M there is a type judgement $\Gamma \vdash M : \tau$ representing all possible typings (Γ', τ') of M in this system. The system in [VAK08] was not proved to have PT while the system introduced here is proved to have PT for β-nf. The concept of a *most general* typing is usually linked to syntactic operations and they vary from system to system. For example, the operations to obtain one typing from another in simply typed systems are *weakening* and *type substitutions*, mapping type variables to types, while in an intersection type system *expansion* is performed to obtain intersection types replicating a simple type through some specific rules. In [We02] J. Wells introduced a system-independent definition of PT and proved that it was the correct generalisation of well known system-dependent definitions such as Hindley's PT for simple type systems [Hi97]. Principal typings has been studied for some intersection type systems ([CDV80], [RV84], [Roc88], [Bak95], [KW04]) and in [CDV80, RV84] were proved that PT for some term's β-nf is principal for the term itself. Partial algorithms, yielding PT whenever succeeds, were proposed in [Roc88, KW04]. In [CW04a] S. Carlier and Wells presented the exact correspondence between the inference mechanism for their intersection type system and the β-reduction. They introduce the *expansion variables*, integrating expansion operations into the type system (see [CW04b]).

We present in this paper a de Bruijn version of the intersection type system originally introduced in [SM96a], with the purpose of characterising the syntactic structure of PT for β-nfs. E. Sayag and M. Mauny intended to develop a system such that, similarly to simply typed systems, the definition of PT would depends on type substitutions only and, as a consequence, the typing system in [SM96a] does not have SR. Although SR is the most basic property and should be satisfied by any typing system, the system infers types to all β-nfs and, because it is a restriction of more complex and well studied systems, is a reasonable way to characterise PT for intersection type systems. In fact, the system in [SM96a] is a proper restriction of some systems presented in [Bak95].

Following, we give some definitions and properties for the untyped λ-calculus with de Bruijn indices, as in [VAK08]. We introduce the type system in Section 2, where some properties are stated and counterexamples for another properties, such as SR, are presented. The type inference algorithm introduced here, its soundness and completeness are at the end of Section 2. The characterisation of PT for β-nfs and the reconstruction algorithm are presented in Section 3. Both algorithms introduced here are similar to the ones presented in [SM96a].

1.1 λ-calculus with de Bruijn indices

Definition 1 (Set Λ_{dB})**.** *The syntax of the λ-calculus with de Bruijn indices, **the λ_{dB}-calculus**, is defined inductively by:*

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

Definition 2. *$FI(M)$, the* **set of free indices** *of $M \in \Lambda_{dB}$, is defined by:*

$$FI(\underline{n}) = \{\underline{n}\} \qquad FI(M_1\ M_2) = FI(M_1) \cup FI(M_2)$$

$$FI(\lambda.M) = \{\underline{n-1}, \forall \underline{n} \in FI(M), n > 1\}$$

The free indices corresponds to the notion of free variables in the λ-calculus with names, hence M is called closed when $FI(M) \equiv \emptyset$. The greatest value of $FI(M)$ is denoted by $sup(M)$. In [VAK08] we give the formal definitions of those concepts. Following, a lemma stating properties about sup related with the structure of terms.

Lemma 1 ([VAK08])**.** *1. $sup(M_1\ M_2) = max(sup(M_1), sup(M_2))$.*

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

Terms like $((\ldots((M_1\ M_2)\ M_3)\ldots)\ M_n)$ are written as $(M_1\, M_2\, \cdots\, M_n)$, as usual. The β-contraction definition in this notation needs a mechanism which detects and updates free indices of terms. Intuitively, the **lift** of M, denoted by M^+, corresponds to an increment by 1 of all free indices occurring in M. Thus, we are able to present the definition of the substitution used by β-contractions, similarly to the one presented in [ARK01].

Definition 3. *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. $\{\underline{n}/N\}\underline{m} = \begin{cases} \underline{m-1}, & \textit{if } m > n \\ N, & \textit{if } m = n \\ \underline{m}, & \textit{if } m < n \end{cases}$

Observe that in item 2 of Definition 3, the lift operator is used to avoid captures of free indices in N. We present the β-contraction as defined in [ARK01].

Definition 4. *β-**contraction** in λdB is defined by $(\lambda.M\,N) \to_\beta \{\underline{1}/N\}M$.*

Notice that item 3 in Definition 3 is the mechanism which does the substitution and updates the free indices in M as consequence of the lead abstractor elimination. The **β-reduction** is defined to be the λ-compatible closure of the β-contraction defined above. A term is in **β-normal form**, β-nf for short, if there is no β-reduction to be done.

Lemma 2. *A term $N \in \Lambda_{dB}$ is a β-nf iff N is one of the following :*

- *$N \equiv \underline{n}$, for any $n \in \mathbb{N}^*$.*
- *$N \equiv \lambda.N'$ and N' is a β-nf.*
- *$N \equiv \underline{n}\, N_1 \cdots N_m$, for some $n \in \mathbb{N}^*$ and $\forall 1 \le j \le m$, N_j is a β-nf.*

Proof. The *necessary* proof is straightforward from β-nf definition. Now, suppose that N is a β-nf. The *sufficient* proof is by induction on the structure of $N \in \Lambda_{dB}$:

- If $N \equiv \underline{n}$ then N is a β-nf.
- Let $N \equiv \lambda.N'$. If N' is not a β-nf then, by the definition of β-reduction, N would not be a β-nf. Hence, N' is a β-nf.
- Let $N \equiv N_1\ N_2$. Since N is a β-nf, one has that both N_1 and N_2 are β-nfs. Hence, by IH, $N_1 \equiv \lambda.N'$, for N' a β-nf, or $\underline{n}\, N'_1 \cdots N'_m$ for $m \ge 0$ and $\forall 1 \le j \le m$, N'_j a β-nf. If $N \equiv \lambda.N'$ then N would β-contract. Hence, $N \equiv \underline{n}\, N'_1 \cdots N'_m\ N_2$.

□

2 The type system and properties

Definition 5. 1. *Let $\mathcal{A}$ be a denumerably infinite* **set of type variables** *and let α, β range over $\mathcal{A}$.*

2. *The set $\mathcal{T}$ of* **restricted intersection types** *is defined by:*

$$\tau, \sigma \in \mathcal{T} ::= \mathcal{A} \,|\, \mathcal{U} \to \mathcal{T} \qquad u \in \mathcal{U} ::= \omega \,|\, \mathcal{U} \wedge \mathcal{U} \,|\, \mathcal{T}$$

Types are quotiented by taking $\wedge$ to be commutative, associative and to have ω as the neutral element.

3. **Contexts** *are ordered lists of $u \in \mathcal{U}$, defined by: $\Gamma ::= nil \,|\, u.\Gamma$*

Γ_i denotes the i-th element of Γ and $|\Gamma|$ denotes the length of Γ.

$\omega^{\underline{n}}$ denotes the sequence $\omega.\omega.\cdots.\omega$ of length n and let $\omega^{\underline{0}}.\Gamma = \Gamma$.

The extension of $\wedge$ to contexts is done by taking nil as the neutral element and $(u_1.\Gamma) \wedge (u_2.\Delta) = (u_1 \wedge u_2).(\Gamma \wedge \Delta)$. Hence, $\wedge$ is commutative and associative on contexts.

4. **Type substitution** *maps type variables to types. Given a type substitution $s : \mathcal{A} \to \mathcal{T}$, the extension for types in $\mathcal{T}$ is given by $s(u \to \tau) = s(u) \to s(\tau)$ and for elements in $\mathcal{U}$ by $s(\omega) = \omega$ and $s(u \wedge v) = s(u) \wedge s(v)$. The extension for contexts is given by $s(nil) = nil$ and $s(u.\Gamma) = s(u).s(\Gamma)$. The domain of a substitution s is defined by $Dom(s) = \{\alpha \mid s(\alpha) \neq \alpha\}$ and, for two substitutions s_1 and s_2 with disjoint domains, let $s_1 + s_2$ be defined by*

$$(s_1 + s_2)(\alpha) \begin{cases} s_i(\alpha) & \text{if } \alpha \in Dom(s_i), \text{for } i \in \{1,2\} \\ \alpha & \text{if } \alpha \notin Dom(s_1) \cup Dom(s_2) \end{cases}$$

5. *$TypeVar(u)$ is the* **set of type variables occurring** *in $u \in \mathcal{U}$. The extension to contexts is straightforward.*

The set $\mathcal{T}$ defined here is equivalent to the one defined in [SM96a].

Lemma 3. *1. If $u \in \mathcal{U}$, then $u = \omega$ or $u = \wedge_{i=1}^{n} \tau_i$ where $n > 0$ and $\forall\, 1 \leq i \leq n$, $\tau_i \in \mathcal{T}$.*

2. If $\tau \in \mathcal{T}$, then $\tau = \alpha$, $\tau = \omega \to \sigma$ or $\tau = \wedge_{i=1}^{n} \tau_i \to \sigma$, where $n > 0$ and $\sigma, \tau_1, \ldots, \tau_n \in \mathcal{T}$.

Proof. 1. By induction on $u \in \mathcal{U}$.

2. By induction on $\tau \in \mathcal{T}$ and Lemma 3.1.

□

Definition 6. *1. The typing rules for system SM are given as follows:*

$$\frac{}{\underline{1} : \langle \tau.nil \vdash \tau \rangle}\ \text{var} \qquad \frac{M : \langle u.\Gamma \vdash \tau \rangle}{\lambda.M : \langle \Gamma \vdash u \to \tau \rangle}\ \to_i$$

$$\frac{\underline{n} : \langle \Gamma \vdash \tau \rangle}{\underline{n{+}1} : \langle \omega.\Gamma \vdash \tau \rangle}\ \text{varn} \qquad \frac{M : \langle nil \vdash \tau \rangle}{\lambda.M : \langle nil \vdash \omega \to \tau \rangle}\ \to_i'$$

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

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

2. System SM_r is obtained from system SM, replacing rule var *by rule*

$$\frac{}{\underline{1} : \langle \sigma_1 \to \cdots \to \sigma_n \to \alpha.nil \vdash \sigma_1 \to \cdots \to \sigma_n \to \alpha \rangle}\ (n \geq 0) \quad \text{var}_r$$

Type judgements will be of the form $M : \langle \Gamma \vdash_S \tau \rangle$, meaning that term M has type τ in system S provided Γ for $FI(M)$. Briefly, M has type τ with Γ in S or (Γ, τ) is a typing of M in S. The S is omitted whenever its is clear to which system we are referring to.

Lemma 4. *SM is a proper extension of SM_r.*

Proof. If $M:\langle\Gamma\vdash_{SM_r}\tau\rangle$ then one has $M:\langle\Gamma\vdash_{SM}\tau\rangle$, trivially. As a counter example on the opposite direction, take $M\equiv(\underline{1}\ \lambda.\underline{1})$. Then one has that $M:\langle\tau\wedge(\tau\to\tau)\to\beta.nil\vdash_{SM}\beta\rangle$, for $\tau=\alpha\to\alpha$, and $(\tau\wedge(\tau\to\tau)\to\beta.nil,\beta)$ is not a typing of M in SM_r. $\square$

Hence, properties stated for the system SM are also true for the system SM_r. The following lemma states that SM is relevant in the sense of [DG94].

Lemma 5. *If $M:\langle\Gamma\vdash_{SM}\tau\rangle$, then $|\Gamma|=sup(M)$ and $\forall 1\leq i\leq|\Gamma|$, $\Gamma_i\neq\omega$ iff $\underline{i}\in FI(M)$.*

Proof. By induction on the derivation $M:\langle\Gamma\vdash u\rangle$.

- If $\dfrac{}{\underline{1}:\langle\tau.nil\vdash\tau\rangle}$, then $|\Gamma|=1=sup(\underline{1})$. Note that $FI(\underline{1})=\{\underline{1}\}$ and $\Gamma_1=\tau$.

- If $\dfrac{\underline{n}:\langle\Gamma\vdash\tau\rangle}{\underline{n+1}:\langle\omega.\Gamma\vdash\tau\rangle}$, then by IH one has $|\Gamma|=sup(\underline{n})=n$, $\Gamma_n\neq\omega$ and $\forall 1\leq i<n$, $\Gamma_i=\omega$. Thus, $|\omega.\Gamma|=1+|\Gamma|=n+1=sup(\underline{n+1})$, $(\omega.\Gamma)_{n+1}=\Gamma_n\neq\omega$, $(\omega.\Gamma)_1=\omega$ and $\forall 1\leq i<n$, $(\omega.\Gamma)_{i+1}=\Gamma_i=\omega$.

- Let $\dfrac{M:\langle u.\Gamma\vdash\sigma\rangle}{\lambda.M:\langle\Gamma\vdash u\to\sigma\rangle}$. By IH, $|u.\Gamma|=sup(M)$ and $\forall 0\leq i\leq sup(M)-1$, $(u.\Gamma)_{i+1}\neq\omega$ iff $\underline{i+1}\in FI(M)$. Hence, $sup(M)=1+|\Gamma|>0$ and, by Lemma 1.2, $sup(\lambda.M)=sup(M)-1=|\Gamma|$. By Definition 2, $\forall 1\leq i\leq sup(\lambda.M)$, $\underline{i}\in FI(\lambda.M)$ iff $\underline{i+1}\in FI(M)$, thus, $(u.\Gamma)_{i+1}=\Gamma_i\neq\omega$ iff $\underline{i}\in FI(\lambda.M)$.

- Let $\dfrac{M:\langle nil\vdash\sigma\rangle}{\lambda.M:\langle nil\vdash\omega\to\sigma\rangle}$. By IH one has $|nil|=sup(M)=0$. Thus, by Lemma 1.2, $sup(\lambda.M)=sup(M)=|nil|$. Note that $FI(M)=FI(\lambda.M)=\emptyset$.

- Let $\dfrac{M_1:\langle\Gamma\vdash\omega\to\tau\rangle\qquad M_2:\langle\Delta\vdash\sigma\rangle}{M_1\ M_2:\langle\Gamma\wedge\Delta\vdash\tau\rangle}$. By IH, $|\Gamma|=sup(M_1)$, $\forall 1\leq i\leq|\Gamma|$ one has $\Gamma_i\neq\omega$ iff $\underline{i}\in FI(M_1)$, $|\Delta|=sup(M_2)$ and $\forall 1\leq j\leq|\Delta|$ one has $\Delta_j\neq\omega$ iff $\underline{j}\in FI(M_2)$. By Lemma 1.1 one has $sup(M_1\ M_2)=max(sup(M_1),sup(M_2))=max(|\Gamma|,|\Delta|)=|\Gamma\wedge\Delta|$. Let $1\leq l\leq|\Gamma\wedge\Delta|$ and suppose w.l.o.g. that $l\leq|\Gamma|,|\Delta|$. Thus, $(\Gamma\wedge\Delta)_l=\Gamma_l\wedge\Delta_l\neq\omega$ *iff* $\Gamma_l\neq\omega$ or $\Delta_l\neq\omega$ *iff* $\underline{l}\in FI(M_1)$ or $\underline{l}\in FI(M_2)$ *iff* $\underline{l}\in FI(M_1)\cup FI(M_2)=FI(M_1\ M_2)$.

- Let $\dfrac{M_1:\langle\Gamma\vdash\wedge_{k=1}^{n}\sigma_k\to\tau\rangle\qquad M_2:\langle\Delta^1\vdash\sigma_1\rangle\ \ldots\ M_2:\langle\Delta^n\vdash\sigma_n\rangle}{M_1\ M_2:\langle\Gamma\wedge\Delta^1\wedge\cdots\wedge\Delta^n\vdash\tau\rangle}$. By IH, $|\Gamma|=sup(M_1)$, $\forall 1\leq i\leq|\Gamma|$ one has $\Gamma_i\neq\omega$ iff $\underline{i}\in FI(M_1)$ and $\forall 1\leq k\leq n$, $|\Delta^k|=sup(M_2)$ and $\forall 1\leq j\leq|\Delta^k|$ one has $\Delta^k_j\neq\omega$ iff $\underline{j}\in FI(M_2)$. Let $\Delta'=\Delta^1\wedge\cdots\wedge\Delta^n$. Thus, $|\Delta'|=sup(M_2)$ and $\forall 1\leq j\leq|\Delta'|$, $\Delta'_j\neq\omega$ *iff* $\underline{j}\in FI(M_2)$. The proof is analogous to the one above.

$\square$

Note that, by Lemma 5 above, system SM is not only relevant but there is a strict relation between the free indices of terms and the length of contexts in their typings. Following, a generation lemma is presented for typings in SM and some specific for SM_r

Lemma 6 (Generation). *1. If $\underline{n}:\langle\Gamma \vdash_{SM} \tau\rangle$, then $\Gamma_n = \tau$.*

2. *If $\underline{n}:\langle\Gamma \vdash_{SM_r} \tau\rangle$, then $\tau = \sigma_1 \to \cdots \to \sigma_k \to \alpha$ for $k \geq 0$.*

3. *If $\lambda.M : \langle nil \vdash_{SM} \tau\rangle$, then either $\tau = \omega \to \sigma$ and $M : \langle nil \vdash \sigma\rangle$ or $\tau = \wedge_{i=1}^{n}\sigma_i \to \sigma$, $n > 0$, and $M : \langle \wedge_{i=1}^{n}\sigma_i.nil \vdash_{SM} \sigma\rangle$ for some $\sigma, \sigma_1, \ldots, \sigma_n \in \mathcal{T}$.*

4. *If $\lambda.M : \langle\Gamma \vdash_{SM} \tau\rangle$ and $|\Gamma| > 0$, then $\tau = u \to \sigma$ for some $u \in \mathcal{U}$ and $\sigma \in \mathcal{T}$, where $M : \langle u.\Gamma \vdash_{SM} \sigma\rangle$.*

5. *If $\underline{n}\ M_1 \cdots M_m : \langle\Gamma \vdash_{SM_r} \tau\rangle$, then $\Gamma = (\omega^{\underline{n-1}}.\sigma_1 \to \cdots \to \sigma_m \to \tau.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$, $\tau = \sigma_{m+1} \to \cdots \to \sigma_{m+k} \to \alpha$ and $\forall 1 \leq i \leq m$, $M_i : \langle\Gamma^i \vdash_{SM_r} \sigma_i\rangle$.*

Proof. 1. By induction on the derivation $\underline{n}:\langle\Gamma \vdash_{SM} \tau\rangle$. Note that $(\omega.\Gamma)_{n+1} = \Gamma_n$.

2. By induction on the derivation $\underline{n}:\langle\Gamma \vdash_{SM_r} \tau\rangle$.

3. By case analysis on the derivation $\lambda.M : \langle nil \vdash_{SM} \tau\rangle$.

4. By case analysis on the derivation $\lambda.M : \langle\Gamma \vdash_{SM} \tau\rangle$, for $|\Gamma| > 0$.

5. By induction on m.

   If $m = 0$, then, by Lemma 6.2, $\tau = \sigma_1 \to \cdots \to \sigma_k \to \alpha$. Thus, by Lemmas 5 and 6.1, $\Gamma = \omega^{\underline{n-1}}.\tau.nil$.

   If $m = m' + 1$, then by case analysis the last step of the derivation is

$$\frac{\underline{n}\ M_1 \cdots M_{m'} : \langle\Gamma \vdash \wedge_{j=1}^{l}\tau_j \to \tau\rangle \quad M_{m'+1} : \langle\Delta^1 \vdash \tau_1\rangle \ \ldots \ M_{m'+1} : \langle\Delta^l \vdash \tau_l\rangle}{\underline{n}\ M_1 \cdots M_{m'}\ M_{m'+1} : \langle\Gamma \wedge \Delta^1 \wedge \cdots \wedge \Delta^l \vdash \tau\rangle}$$

   By IH, $\Gamma = (\omega^{\underline{n-1}}.\sigma_1 \to \cdots \to \sigma_{m'} \to (\wedge_{j=1}^{l}\tau_j \to \tau).nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^{m'}$, $\wedge_{j=1}^{l}\tau_j \to \tau = \sigma_{m'+1} \to \cdots \to \sigma_{m'+k} \to \alpha$ and $\forall 1 \leq i \leq m'$, $M_i : \langle\Gamma^i \vdash_{SM_r} \sigma_i\rangle$. Hence, $l = 1$, $\tau_1 = \sigma_{m'+1}$ and $\tau = \sigma_{m'+2} \to \cdots \to \sigma_{m+k} \to \alpha$. Thus, taking $\Gamma^{m'+1} = \Delta^1$ and $\sigma_{m'+1} = \tau_1$, the result holds.

□

In the following, we will give counterexamples to show that neither subject expansion nor reduction holds.

Example 1. *In order to have the subject expansion property, we need to prove the statement: If $\{\underline{1}/N\}M : \langle\Gamma \vdash \tau\rangle$ then $(\lambda.M\ N) : \langle\Gamma \vdash \tau\rangle$. Let $M \equiv \lambda.\underline{1}$ and $N \equiv \underline{3}$, hence $\{\underline{1}/\underline{3}\}\lambda.\underline{1} = \lambda.\underline{1}$. We have that, by generation lemmas, $\lambda.\underline{1} : \langle nil \vdash \alpha \to \alpha\rangle$. Thus, $\lambda.\lambda.\underline{1} : \langle nil \vdash \omega \to \alpha \to \alpha\rangle$ and $\underline{3} : \langle\omega.\omega.\beta.nil \vdash \beta\rangle$, then $\lambda.\lambda.\underline{1}\ \underline{3} : \langle\omega.\omega.\beta.nil \vdash \alpha \to \alpha\rangle$.*

For subject reduction, we need the statement: If $(\lambda.M\ N) : \langle\Gamma \vdash \tau\rangle$ then $\{\underline{1}/N\}M : \langle\Gamma \vdash \tau\rangle$. Note that if we take M and N as in the example above, we have the same problem as before but in the other way round. In other words, we have a restriction on the original context after the β-reduction, since we loose the typing information regarding $N \equiv \underline{3}$.

One possible solution for those problems is to replace rule $\to'_e$ by

$$\frac{M : \langle\Gamma \vdash \omega \to \tau\rangle}{M\ N : \langle\Gamma \vdash \tau\rangle}$$

This approach was originally presented in [SM96b], but a new notion replacing free index should be introduced since we would not have the typing information for all free indices occurring in a term. In [SM96b], and in [SM97], no notion is presented instead of the usual free variables, which is wrongly used to state things that are not actually true.

The other way to achieve the desired properties is to think about the meaning of the properties itself. Since, by Lemma 5, the system is related to relevant logic (see [DG94]), the notion of expansion and restriction of contexts is an interesting way to talk about subject expansion and reduction. These concepts were presented in [KN07] for environments.

Even though, any β-nf is typeable with system SM_r. We introduce the type inference algorithm `Infer` for β-nfs, similarly to [SM96a].

Definition 7 (Type inference algorithm)**.** *Let N be a β-nf:*

`Infer`$(N) =$

Case $N = \underline{n}$
  let α *be a fresh type variable*
  return $(\omega \underline{^{n-1}} . \alpha.nil, \alpha)$
Case $N = \lambda.N'$
  let $(\Gamma', \sigma) =$ `Infer`(N')
  if $(\Gamma' = u.\Gamma)$ **then**
  return $(\Gamma, u \to \sigma)$
  else
  return $(nil, \omega \to \sigma)$
Case $N = \underline{n}\, N_1 \cdots N_m$
  let $(\Gamma^1, \sigma_1) =$ `Infer`(N_1)
  $\vdots$
  $(\Gamma^m, \sigma_m) =$ `Infer`(N_m)
  α *be a fresh type variable*
  return $((\omega \underline{^{n-1}} . \sigma_1 \to \cdots \to \sigma_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m, \alpha)$

Remark The notion of *fresh type variables* is used to prove completeness. The freshness of a variable is to guarantee that each time some type variable is picked up from $\mathcal{A}$ it is a new one. Therefore, two non overlapped calls to `Infer` return pairs with disjoints sets of type variables.

Theorem 1 (Soundness). *If N is a β-nf and* $\texttt{Infer}(N) = (\Gamma, \sigma)$*, then* $N : \langle \Gamma \vdash_{SM_r} \sigma \rangle$.

Proof. By structural induction on N.

- If $N \equiv \underline{n}$ then $\texttt{Infer}(\,\underline{n}\,) = (\omega \frac{n-1}{} .\alpha.nil, \alpha)$. By rule var_r, $\underline{1} : \langle \alpha.nil \vdash \alpha \rangle$ and, by rule varn applied $n-1$ times, $\underline{n} : \langle \omega \frac{n-1}{} .\alpha.nil \vdash \alpha \rangle$.
- Let $N \equiv \lambda.N'$. If $(\Gamma', \sigma) = \texttt{Infer}(N')$ then, by IH, $N' : \langle \Gamma' \vdash \sigma \rangle$. Thus, if $\Gamma' = u.\Gamma$ then $\texttt{Infer}(\lambda.N') = (\Gamma, u \to \sigma)$ and, by rule $\to_i$, $\lambda.N' : \langle \Gamma \vdash u \to \sigma \rangle$, otherwise $\texttt{Infer}(\lambda.N') = (nil, \omega \to \sigma)$ and, by rule $\to_i'$, $\lambda.N' : \langle nil \vdash \omega \to \sigma \rangle$.
- Let $N \equiv \underline{n}\, N_1 \cdots N_m$. If $\forall 1 \leq i \leq m$, $(\Gamma^i, \sigma_i) = \texttt{Infer}(N_i)$ then, by IH, $\forall 1 \leq i \leq m$, $N_i : \langle \Gamma^i \vdash \sigma_i \rangle$. Let $\Delta = \omega \frac{n-1}{} .\sigma_1 \to \cdots \to \sigma_m \to \alpha.nil$. Hence $\texttt{Infer}(N) = (\Delta \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m, \alpha)$ for some fresh type variable α. By rule var_r and by rule varn $n-1$-times, $\underline{n} : \langle \Delta \vdash \sigma_1 \to \cdots \to \sigma_m \to \alpha \rangle$ and, by rule $\to_e$ m-times, $N : \langle \Delta \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \vdash \alpha \rangle$.

$\square$

Note that, since the choice of the new type variables is not fixed, $\texttt{Infer}$ is well defined up to the name of type variables.

Corollary 1. *If N is a β-nf then N is typeable in system SM_r.*

Theorem 2 (Completeness). *If $N : \langle \Gamma \vdash_{SM_r} \sigma \rangle$, N a β-nf, then for $(\Gamma', \sigma') =$* $\texttt{Infer}(N)$ *exists a type substitution s such that $s(\Gamma') = \Gamma$ and $s(\sigma') = \sigma$.*

Proof. By structural induction on N

- Let $N \equiv \underline{n}$. If $\underline{n} : \langle \Gamma \vdash \sigma \rangle$ then, by Lemmas 5 and 6.1, $\Gamma \equiv \omega \frac{n-1}{} .\sigma.nil$. One has that $\texttt{Infer}(\,\underline{n}\,) = (\omega \frac{n-1}{} .\alpha.nil, \alpha)$, then take $s = [\alpha/\sigma]$.
- Let $N \equiv \lambda.N'$ and suppose that $\lambda.N' : \langle \Gamma \vdash \sigma \rangle$.

  If $\Gamma \equiv nil$, then by Lemma 6.3 either $\sigma = \omega \to \tau$ and $N' : \langle nil \vdash \tau \rangle$ or $\sigma = \wedge_{j=1}^{n} \sigma_j \to \tau$ and $N' : \langle \wedge_{j=1}^{n} \sigma_j .nil \vdash \tau \rangle$. The former, by IH, $\texttt{Infer}(N') = (\Gamma', \tau')$ and there exists s such that $s(\tau') = \tau$ and $s(\Gamma') = nil$, then one has $\Gamma' = nil$. Hence, $\texttt{Infer}(\lambda.N') = (nil, \omega \to \tau')$ and $s(\omega \to \tau') = s(\omega) \to s(\tau') = \sigma$. The latter, by IH, $\texttt{Infer}(N') = (\Gamma', \tau')$ and there exists s such that $s(\tau') = \tau$ and $s(\Gamma') = \wedge_{j=1}^{n} \sigma_j .nil$. Then $\Gamma' = u.nil$ for $s(u) = \wedge_{j=1}^{n} \sigma_j$, hence $\texttt{Infer}(\lambda.N') = (nil, u \to \tau')$ and $s(u \to \tau') = s(u) \to s(\tau') = \sigma$.

  Otherwise, by Lemma 6.4 one has that $\sigma = u \to \tau$ and $N' : \langle u.\Gamma \vdash \tau \rangle$. The proof is analogous to the one above.
- Let $N \equiv \underline{n}\, N_1 \cdots N_m$. If $\underline{n}\, N_1 \cdots N_m : \langle \Gamma \vdash \sigma \rangle$ then, by Lemma 6.5, $\Gamma = (\omega \frac{n-1}{} .\sigma_1 \to \cdots \to \sigma_m \to \sigma.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$ and $\forall 1 \leq i \leq m$, $N_i : \langle \Gamma^i \vdash \sigma_i \rangle$. By IH, $\forall 1 \leq i \leq m$, $\texttt{Infer}(N_i) = (\Gamma^{i'}, \sigma_i')$ and there is a s_i such that $s_i(\sigma_i') = \sigma_i$ and $s_i(\Gamma^{i'}) = \Gamma^i$. One has that $\texttt{Infer}(N) = ((\omega \frac{n-1}{} .\sigma_1' \to \cdots \to \sigma_m' \to \alpha.nil) \wedge \Gamma^{1'} \wedge \cdots \wedge \Gamma^{m'}, \alpha)$, for some fresh type

variable α. The domain of each s_i is compounded by the type variables returned by each call of `Infer` for the corresponding N_i, consequently they are disjoint. Thus, for $s=[\alpha/\sigma]+s_1+\cdots+s_m$ the result holds.

□

Hence, the pair returned by `Infer` for some β-nf N is a most general typing of N is SM_r. Note that these typings are unique up to renaming of type variables.

Corollary 2. *If N is a β-nf, then $(\Gamma,\sigma) = \texttt{Infer}(N)$ is a principal typing of N in SM_r.*

3 Characterisation of principal typings

Following, we give some characterisation of principal typings for β-nfs, analogue to [SM96a]. To begin with, we introduce proper subsets of $\mathcal{T}$ and $\mathcal{U}$ containing the pairs returned by `Infer`.

Definition 8. *1. Let $\mathcal{T}_C$, $\mathcal{T}_{NF}$ and $\mathcal{U}_C$ be defined by:*

$$\rho \in \mathcal{T}_C ::= \mathcal{A} \,|\, \mathcal{T}_{NF} \rightarrow \mathcal{T}_C \qquad\qquad \varphi \in \mathcal{T}_{NF} ::= \mathcal{A} \,|\, \mathcal{U}_C \rightarrow \mathcal{T}_{NF}$$

$$v \in \mathcal{U}_C ::= \omega \,|\, \mathcal{U}_C \wedge \mathcal{U}_C \,|\, \mathcal{T}_C$$

2. Let $\mathcal{C}$ be the set of contexts $\Gamma ::= nil \,|\, v.\Gamma$ such that $v \in \mathcal{U}_C$. Observe that $\mathcal{C}$ is closed under $\wedge$.

Lemma 7. *If $\texttt{Infer}(N) = (\Gamma,\sigma)$, N a β-nf, then $(\Gamma,\sigma) \in \mathcal{C}\times\mathcal{T}_{NF}$.*

Proof. By structural induction on N.

- Let $N \equiv \underline{n}$. One has that $\texttt{Infer}(\underline{n})=(\underline{\omega^{n-1}}.\alpha.nil,\alpha)$. Note that $\alpha\in\mathcal{T}_{NF}$ and $\omega,\alpha\in\mathcal{U}_C$, then $\underline{\omega^{n-1}}.\alpha.nil\in\mathcal{C}$.
- Let $N \equiv \lambda.N'$. If $(\Gamma',\sigma)=\texttt{Infer}(N')$ then, by IH, $\sigma\in\mathcal{T}_{NF}$ and $\Gamma'\in\mathcal{C}$. If $\Gamma'=v.\Gamma$ then $\texttt{Infer}(\lambda.N')=(\Gamma, v\rightarrow\sigma)$, hence $\Gamma\in\mathcal{C}$ and, since $v\in\mathcal{U}_C$, $v\rightarrow\sigma\in\mathcal{T}_{NF}$. Otherwise, $\texttt{Infer}(\lambda.N')=(nil,\omega\rightarrow\sigma)$ and the result holds.
- Let $N \equiv \underline{n}\,N_1\cdots N_m$. If $\forall 1\leq i\leq m$, $(\Gamma^i,\sigma_i)=\texttt{Infer}(N_i)$ then, by IH, $\Gamma^i\in\mathcal{C}$ and $\sigma_i\in\mathcal{T}_{NF}$. Let $\Delta=\underline{\omega^{n-1}}.\sigma_1\rightarrow\cdots\rightarrow\sigma_m\rightarrow\alpha.nil$, for some fresh type variable α, hence $\texttt{Infer}(N)=(\Delta\wedge\Gamma^1\wedge\cdots\wedge\Gamma^m,\alpha)$. Note that $\sigma_1\rightarrow\cdots\rightarrow\sigma_m\rightarrow\alpha\in\mathcal{T}_C\subset\mathcal{U}_C$. Thus, since $\Delta,\Gamma^1,\ldots,\Gamma^m\in\mathcal{C}$, one has that $\Delta\wedge\Gamma^1\wedge\cdots\wedge\Gamma^m\in\mathcal{C}$ and $\alpha\in\mathcal{T}_{NF}$.

□

Definition 9. *Let $Im(\texttt{Infer})$ be defined as the set of pairs (Γ,σ) such that $(\Gamma,\sigma)=\texttt{Infer}(N)$ for some β-nf N.*

Corollary 3. *$Im(\texttt{Infer})\subseteq\mathcal{C}\times\mathcal{T}_{NF}$.*

We use the usual notion of **positive** and **negative** occurrences of type variables and of **final occurrences** for elements $u \in \mathcal{U}$ (see [Kri93]). For contexts, the positive and negative occurrences are the respective occurrences in the types forming the contexts' sequences.

Definition 10. *Let $\Gamma \in \mathcal{C}$ and $\varphi \in \mathcal{T}_{NF}$. The $\mathcal{C}$-**types** T are defined by*

$$T ::= \Gamma \Rightarrow \varphi \mid \Delta \Rightarrow \qquad s.t.\ |\Delta| > 0$$

Note that, for any β-nf N, $\texttt{Infer}(N)$ has a unique corresponding $\mathcal{C}$-type T^N. The corresponding A-types in [SM96a] are defined by taking the set of multisets associated to an environment and transforming them in a single multiset used on the left hand of $\Rightarrow$. Thus, for an environment A and type τ, $\overline{A} \Rightarrow \tau$ is the A-type with $\overline{A}$ being the multiset obtained from A. On Definition 10 above the sequential structure of contexts are preserved.

Definition 11. *Let $T = \Gamma \Rightarrow \varphi$ be a $\mathcal{C}$-type, T' is* **held** *in T if $T' = \Gamma' \Rightarrow$ or $\Gamma' \Rightarrow \varphi$, such that $\Gamma = \Gamma' \wedge \Delta$ for $\Gamma' \neq \omega^{\underline{n}}$ and some context Δ. If $T' \neq T$ then T' is* **strictly held** *in T.*

Observe that on Definition 11 above we have that Γ' can be *nil* for $T' = \Gamma' \Rightarrow \varphi$ and $\Delta = \omega^{\underline{n}}$ for any $n \leq |\Gamma|$ when $\Gamma' = \Gamma$.

Definition 12. *The set $L(T)$ of the* **left subtypes** *for some $\mathcal{C}$-type T is defined by structural induction:*

- $L(\Gamma \Rightarrow) = L(\Gamma)$.
- $L(\Gamma \Rightarrow \varphi) = L(\Gamma) \cup L(\varphi)$.
- $L(v.\Gamma) = \{v\} \cup L(\Gamma)$ *if* $v \neq \omega$ *and* $L(\Gamma)$ *otherwise.*
- $L(nil) = \emptyset$.
- $L(v \rightarrow \varphi) = \{v\} \cup L(\varphi)$ *if* $v \neq \omega$ *and* $L(\varphi)$ *otherwise.*
- $L(\alpha) = \emptyset$.

The notion of sign of occurrences for type variable are straightforward extended to $\mathcal{C}$-types, where the polarity changes on the left side of $\Rightarrow$. We have that $TypeVar(\Gamma \Rightarrow \varphi) = TypeVar(\Gamma) \cup TypeVar(\varphi)$.

Definition 13. *A $\mathcal{C}$-type T is* **closed** *if each type variable $\alpha \in TypeVar(T)$ has exactly one positive and one negative occurrences in T.*

Lemma 8. *1. $v.\Gamma \Rightarrow \varphi$ is closed iff $\Gamma \Rightarrow v \rightarrow \varphi$ is closed.*

2. $nil \Rightarrow \varphi$ is closed iff $nil \Rightarrow \omega \rightarrow \varphi$ is closed.

3. If $\forall 1 \leq i \leq m$, $T_i = \Gamma^i \Rightarrow \varphi_i$ is closed and $TypeVar(T_i)$ are pairwise disjoint, then $(\omega^{\underline{n-1}}.\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \Rightarrow \alpha$ is closed for any fresh type variable α.

Proof. 1. Let $T = v.\Gamma \Rightarrow \varphi$ and $T' = \Gamma \Rightarrow v \to \varphi$. Note that $TypeVar(T) = TypeVar(T')$ and that the sign for type variable occurrences in v for both T and T' are exactly the same.

2. analogous to the proof above.

3. Let $T = (\omega \underline{n-1}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \Rightarrow \alpha$ be the $\mathcal{C}$-type as described. Since $TypeVar(T_i)$ are pairwise disjoint, $TypeVar(T) = \cup_{i=1}^{m} TypeVar(T_i) \cup \{\alpha\}$ and T has exactly two occurrences of each type variable. Note that $\forall 1 \leq i \leq m$ the type variable occurrences in Γ^i and φ_i have exactly the same sign on both T_i and T and that α has one positive and one negative occurrence in T. Hence, T is closed.

□

Definition 14. *A $\mathcal{C}$-type $T = \Gamma \Rightarrow \varphi$ is* **finally closed***, f.c. for short, if the final occurrence of φ is also the final occurrence of a type in $L(T)$.*

Lemma 9. *1. $v.\Gamma \Rightarrow \varphi$ is finally closed iff $\Gamma \Rightarrow v \to \varphi$ is finally closed.*

2. $nil \Rightarrow \varphi$ is finally closed iff $nil \Rightarrow \omega \to \varphi$ is finally closed.

Proof. 1. Let $T = v.\Gamma \Rightarrow \varphi$ and $T' = \Gamma \Rightarrow v \to \varphi$. Note that the final occurrence of $v \to \varphi$ is the same as of φ. If $v \neq \omega$, by Definition 12, $L(T) = L(v.\Gamma) \cup L(\varphi) = \{v\} \cup L(\Gamma) \cup L(\varphi) = L(\Gamma) \cup L(v \to \varphi) = L(T')$. Otherwise, $L(T) = L(\omega.\Gamma) \cup L(\varphi) = L(\Gamma) \cup L(\varphi) = L(\Gamma) \cup L(\omega \to \varphi) = L(T')$. Hence, T is f.c. iff T' is f.c.

2. analogous to the proof above.

□

Definition 15. *A $\mathcal{C}$-type T is said* **minimally closed***, m.c. for short, if there is no closed T' strictly held in T.*

Lemma 10. *1. If $v.\Gamma \Rightarrow \varphi$ is m.c. for $v \neq \omega$, then $\Gamma \Rightarrow v \to \varphi$ is m.c.*

2. $\omega.\Gamma \Rightarrow \varphi$ is m.c. iff $\Gamma \Rightarrow \omega \to \varphi$ is m.c.

3. $nil \Rightarrow \varphi$ is m.c. iff $nil \Rightarrow \omega \to \varphi$ is m.c.

4. If $\forall 1 \leq i \leq m$, $T_i = \Gamma^i \Rightarrow \varphi_i$ is m.c. and $TypeVar(T_i)$ are pairwise disjoint, then $T = (\omega \underline{n-1}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \Rightarrow \alpha$ is m.c. for any fresh type variable α.

Proof. 1. Let $T = v.\Gamma \Rightarrow \varphi$ be m.c. for $v \neq \omega$ and let $T' = \Gamma \Rightarrow v \to \varphi$. Let T'' be strictly held in T'. If $T'' = \Gamma' \Rightarrow v \to \varphi$ then $T''' = v.\Gamma' \Rightarrow \varphi$ is strictly held in T. By Lemma 8.1, T'' is closed iff T''' is closed. Thus, since T is m.c., T'' cannot be closed. If $T'' = \Gamma' \Rightarrow$ then one has similarly that T'' cannot be closed. Hence, T' is m.c.

2. Let T be strictly held in $\omega.\Gamma \Rightarrow \varphi$. One has that $T = \omega.\Gamma' \Rightarrow \varphi$ is strictly held in $\omega.\Gamma \Rightarrow \varphi$ iff $T' = \Gamma' \Rightarrow \omega \rightarrow \varphi$ is strictly held in $\Gamma \Rightarrow \omega \rightarrow \varphi$. There is a corresponding T' for $T = nil \Rightarrow \varphi$ and for $T = \omega.\Gamma' \Rightarrow$. Therefore, by Lemma 8.1, there is a closed T strictly held in $\omega.\Gamma \Rightarrow \varphi$ iff there is a closed T' strictly held in $\Gamma \Rightarrow \omega \rightarrow \varphi$.

3. analogous to the proof above.

4. Let T' be held in T defined above and suppose that T' is closed. If $T' = \Gamma' \Rightarrow$ then, since $|\Gamma'| > 0$, $\Gamma' = \Delta^i \wedge \Gamma''$ for some i s.t. $\Gamma^i = \Delta^i \wedge \Delta'$, $|\Delta^i| > 0$. Note that $TypeVar(\Gamma^i)$ are pairwise disjoint, thus if $\Delta^i \neq \Gamma^i$ ($\Delta' \neq nil$) then $\Delta^i \Rightarrow$ would be closed and strictly held in T^i. Hence, $\Delta^i = \Gamma^i$ ($\Delta' = nil$) and similarly $\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha$ must be in Γ', giving a non closed $\mathcal{C}$-type T'. If $T' = \Gamma' \Rightarrow \alpha$ then with a similar argument one has that $\Gamma' = (\omega \underline{n-1}.\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$. Therefore, T' is closed iff T is closed and $T' = T$. Hence, T is m.c.

□

Definition 16. *A $\mathcal{C}$-type T is called* **complete** *if T is closed, finally closed and minimally closed.*

Lemma 11. *1. If $v.\Gamma \Rightarrow \varphi$ is complete for $v \neq \omega$ then $\Gamma \Rightarrow v \rightarrow \varphi$ is complete.*

2. $\omega.\Gamma \Rightarrow \varphi$ is complete iff $\Gamma \Rightarrow \omega \rightarrow \varphi$ is complete.

3. $nil \Rightarrow \varphi$ is complete iff $nil \Rightarrow \omega \rightarrow \varphi$ is complete.

4. If $\forall 1 \leq i \leq m$, $T_i = \Gamma^i \Rightarrow \varphi_i$ is complete and $TypeVar(T_i)$ are pairwise disjoint, then $T = (\omega \underline{n-1}.\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \Rightarrow \alpha$ is complete for any fresh type variable α.

Proof. 1. By Lemmas 8.1, 9.1 and 10.1.

2. By Lemmas 8.1, 9.1 and 10.2.

3. By Lemmas 8.2, 9.2 and 10.3.

4. By Lemmas 8.3 and 10.4 one has that the T described above is respectively closed and m.c. Note that $(\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha) \wedge (\Gamma^1 \wedge \cdots \wedge \Gamma^m)_n \in L(T)$, thus T is f.c.

□

Lemma 12. *If N is a β-nf then T^N is complete.*

Proof. By structural induction on N.

- Let $N \equiv \underline{n}$. One has that $\texttt{Infer}(N) = (\omega \underline{n-1}.\alpha.nil, \alpha)$, hence $T^N = \omega \underline{n-1}.\alpha.nil \Rightarrow \alpha$. Note that $L(T^N) = \{\alpha\}$. Thus, T^N is closed and finally closed. The only two $\mathcal{C}$-types strictly held in T^N are $\omega \underline{n-1}.\alpha.nil \Rightarrow$ and $nil \Rightarrow \alpha$ which are not closed, hence T^N is minimally closed.

- Let $N \equiv \lambda.N'$. If $(\Gamma', \varphi) = \texttt{Infer}(N')$ then, by IH, $T^{N'} = \Gamma' \Rightarrow \varphi$ is complete.

  If $\Gamma' = v.\Gamma$ then $\texttt{Infer}(\lambda.N') = (\Gamma, v \to \varphi)$ and $T^N = \Gamma \Rightarrow v \to \varphi$. If $v \neq \omega$, then by Lemma 11.1 T^N is complete. Otherwise, by Lemma 11.2, T^N is complete.

  If $\Gamma' = nil$ then $\texttt{Infer}(\lambda.N') = (nil, \omega \to \varphi)$ and, by Lemma 11.3, T^N is complete.

- Let $N \equiv \underline{n}\, N_1 \cdots N_m$. If $\forall 1 \leq i \leq m$, $(\Gamma^i, \varphi_i) = \texttt{Infer}(N_i)$ then, by IH, T^{N_i} is complete. Observe that $TypeVar(T^{N_i})$ are pairwise disjoint because they correspond to disjoint calls of $\texttt{Infer}$. One has that $\texttt{Infer}(N) = ((\omega \underline{n-1}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m, \alpha)$, for some fresh type variable α. Thus, by Lemma 11.4, T^N is complete.

□

Note that on items 1 and 4 in Lemma 11 we only have *sufficiency* proofs. Following we give counterexamples for each *necessary* condition.

Example 2. *Let $T = \Gamma \Rightarrow \varphi$ be complete. Then, for any fresh $\alpha \in \mathcal{A}$, take $T' = \Gamma \Rightarrow (\alpha \to \alpha) \to \varphi$. Therefore, T' is complete but $\alpha \to \alpha.\Gamma \Rightarrow \varphi$ is not m.c.*

Example 3. *Let $T = \beta_1 \to (\beta_2 \to \beta_3) \to \beta_4.(\beta_1 \to \beta_4) \to (\beta_3 \to \beta_2) \to \alpha.nil \Rightarrow \alpha$. Note that T is complete but there is no such a partition of complete $\mathcal{C}$-types.*

Hence, to have complete $\mathcal{C}$-types which satisfy those *necessary* conditions, we present the notion of principal $\mathcal{C}$-types, as done in [SM96a].

Definition 17. *Let T be a complete $\mathcal{C}$-type. T is called principal if:*

- *$T = \omega \underline{n-1}.\alpha.nil \Rightarrow \alpha$.*
- *$T = nil \Rightarrow \omega \to \varphi$ and $nil \Rightarrow \varphi$ is principal.*
- *$T = \Gamma \Rightarrow v \to \varphi$ such that either $\Gamma \neq nil$ or $v \neq \omega$ and $v.\Gamma \Rightarrow \varphi$ is principal.*
- *$T = \Gamma \Rightarrow \alpha$ and there are $\Gamma^1, \ldots, \Gamma^m \in \mathcal{C}$ and $n \in \mathbb{N}^*$ such that $\Gamma = (\omega \underline{n-1}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$ and $\forall 1 \leq i \leq m$, $\Gamma^i \Rightarrow \varphi_i$ is principal.*

Observe that in Definition 17 above we explicitly require the existence of the corresponding partition in the case $T = \Gamma \Rightarrow \alpha$ for $\Gamma \neq \omega \underline{n-1}.\alpha.nil$ and that $v.\Gamma \Rightarrow \varphi$ is also principal thus complete for $T = \Gamma \Rightarrow v \to \varphi$ such that $\Gamma \neq nil$ or $v \neq \omega$. Although we have that, by Lemma 11.2, $T = nil \Rightarrow \omega \to \varphi$ is complete iff $T' = nil \Rightarrow \varphi$ is complete, this case has to be defined similarly. If in Definition 17 we only have instead: "$T = nil \Rightarrow \omega \to \varphi$" then we would guarantee only the completeness of T', letting a counterexample as in Example 2 to be presented.

Lemma 13. *If N is a β-nf then T^N is principal.*

Proof. By structural induction on N. By Lemma 12, T^N is complete:

- If $N \equiv \underline{n}$ then $T^N = \omega \underline{^{n-1}}.\alpha.nil \Rightarrow \alpha$.
- Let $N \equiv \lambda.N'$ and $T^{N'} = \Gamma' \Rightarrow \varphi$. By IH $T^{N'}$ is principal.

  If $\Gamma' = v.\Gamma$ then $T^{\lambda.N'} = \Gamma \Rightarrow v \rightarrow \varphi$. Observe that if $\Gamma = nil$ then, by Lemma 5, $v \neq \omega$. Hence, $T^{\lambda.N'}$ is principal.

  Otherwise $T^{\lambda.N'} = nil \Rightarrow \omega \rightarrow \varphi$, hence $T^{\lambda.N'}$ is principal.
- Let $N \equiv \underline{n}\, N_1 \cdots N_m$ and $\forall 1 \leq i \leq m$, $T^{N_i} = \Gamma^i \Rightarrow \varphi_i$. One has that $T^N = (\omega \underline{^{n-1}}.\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m \Rightarrow \alpha$ for some fresh $\alpha \in \mathcal{A}$ and, by IH, T^{N_i} is principal $\forall 1 \leq i \leq m$. Thus, T^N is principal.

□

Therefore, the syntactic definition of principal $\mathcal{C}$-types contains the principal typings for β-nfs returned by `Infer`.

Definition 18. *Let* $\mathcal{P} = \{(\Gamma, \varphi) \in \mathcal{C} \times \mathcal{T}_{NF} \mid \Gamma \Rightarrow \varphi$ *is principal*$\}$.

In other words, by Lemma 13 and analogously to [SM96a]:

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

Definition 19. *Let* $FO(\alpha, \Gamma)$ *be a set defined as*

$$FO(\alpha, \Gamma) = \{(i, \Gamma_i) \mid \alpha \textit{ is the final occurrence of } \Gamma_i, \forall 1 \leq i \leq |\Gamma|\}$$

The set $FO(\alpha, \Gamma)$ for $T = \Gamma \Rightarrow \alpha$ principal, specifically closed and finally closed, has properties used in the reconstruction algorithm's definition.

Lemma 14. *Let* $T = \Gamma \Rightarrow \alpha$ *be a* $\mathcal{C}$*-type. If* T *is finally closed then* $FO(\alpha, \Gamma) \neq \emptyset$*. If* T *is also closed then* $FO(\alpha, \Gamma)$ *has exactly one element* (i, v)*, such that* $v = (\varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha) \wedge v'$*, for* $m \geq 0$*,* $\alpha \notin TypeVar(v')$ *and* $v' \in \mathcal{U}_C$*.*

Proof. Let $T = \Gamma \Rightarrow \alpha$. By Definition 12, $L(T) = \{\Gamma_i \neq \omega, \forall 1 \leq i \leq |\Gamma|\}$, hence if T is finally closed then at least one element of Γ has α as its final occurrence. Let $(i, v) \in FO(\alpha, \Gamma)$. If T is also closed then Γ has exactly one positive occurrence of α, hence α occurs uniquely in $v = \Gamma_i$. Note that $v \in \mathcal{U}_C$. If $v \in \mathcal{T}_C$ then by induction on its structure $v = \varphi_1 \rightarrow \cdots \rightarrow \varphi_m \rightarrow \alpha$ for $m \geq 0$ ($v = \alpha$ if $m = 0$). Otherwise, $v = v_1 \wedge v_2$ and α occurs positively either in v_1 or in v_2. Thus, by induction on the structure of elements in $\mathcal{U}_C$, commutativity and associativity of $\wedge$, the result holds. □

We introduce the algorithm `Recon`, to reconstruct a β-nf N from $(\Gamma, \varphi) \in \mathcal{P}$ such that `Infer`$(N) = (\Gamma, \varphi)$, similar to the algorithm introduced in [SM96a].

Definition 20 (Reconstruction algorithm)**.** .

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

Case (nil,α)
  fail
Case (Γ,α)
  let $\{(i^1,u_1),\dots,(i^m,u_m)\} = FO(\alpha,\Gamma)$
  if $m=1$ *and* $u_1 = (\tau_1 \to \cdots \to \tau_n \to \alpha) \wedge u'$ *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 $(N_1,\Delta^1) = \texttt{Recon}(\Gamma^1,\tau_1)$
        $\vdots$
        $(N_n,\Delta^n) = \texttt{Recon}(\Gamma^n,\tau_n)$
        $\Delta' = \omega^{\underline{i^1-1}}.\tau_1 \to \cdots \to \tau_n \to \alpha.nil$
        $\Gamma' = \Delta' \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^n$
        $\Gamma = \Gamma' \wedge \Delta$, *s.t.* $\Delta \neq \omega^{\underline{j}}$, $\forall 1\leq j\leq|\Gamma|$
      return $(\underline{i}^1 N_1 \cdots N_n, \Delta \wedge \Delta^1 \wedge \cdots \wedge \Delta^n)$
      else fail
    else fail
Case $(\Gamma, u \to \tau)$
  if $\Gamma = nil$ *and* $u = \omega$
    then let $(N,\Delta) = \texttt{Recon}(nil,\tau)$
    else let $(N,\Delta) = \texttt{Recon}(u.\Gamma,\tau)$
  if $\Delta = nil$
    then return $(\lambda.N,\Delta)$
    else fail

Lemma 15. *Let* $(\Gamma,\varphi) \in \mathcal{P}$*. Then* $\texttt{Recon}(\Gamma,\varphi) = (N, nil)$*,* N *a* β*-nf such that* $\texttt{Infer}(N) = (\Gamma,\varphi)$*.*

Proof. By recurrence on the number of calls to $\texttt{Recon}$.

- Case (Γ,α). Let $T = \Gamma \Rightarrow \alpha$.

  By hypothesis $(\Gamma,\alpha) \in \mathcal{P}$, thus T is principal and in particular closed and finally closed. By Lemma 14, $FO(\alpha,\Gamma) = \{(i, (\varphi_1 \to \cdots \to \varphi_m \to \alpha) \wedge v')\}$ where $\alpha \notin TypeVar(v')$. Since Γ_i is the only occurrence of α in Γ, $\Gamma = (\omega^{\underline{i-1}}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Delta''$ such that $\alpha \notin TypeVar(\Delta'')$.

  If $m=0$, then in $\texttt{Recon}$ one has $\Gamma' = \Delta' = \omega^{\underline{i-1}}.\alpha.nil$, hence $T = \Gamma' \wedge \Delta'' \Rightarrow \alpha$. T is minimally closed, thus $\Delta'' = nil$ and $\Gamma = \Gamma'$. Then, $\texttt{Recon}(\Gamma,\alpha) = (\underline{i}, nil)$ and $\texttt{Infer}(\underline{i}) = (\omega^{\underline{i-1}}.\alpha.nil, \alpha)$.

  Otherwise, since T is principal, there are $\Gamma^1,\dots,\Gamma^m$ and $n \in \mathbb{N}^*$ such that $\Gamma = (\omega^{\underline{n-1}}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m$ and $\forall 1\leq j\leq m$, $\Gamma^j \Rightarrow \varphi_j$ is principal. Hence, $n = i$ and by IH $\forall 1\leq j\leq m$, $\texttt{Recon}(\Gamma^j,\varphi_j) = (N_j, nil)$, N_j a β-nf such that $\texttt{Infer}(N_j) = (\Gamma^j,\varphi_j)$. Hence, in $\texttt{Recon}$ one has that $\Gamma = \Gamma'$, consequently $\Delta = nil$. Then, $\texttt{Recon}(\Gamma,\alpha) = (\underline{i}\, N_1 \cdots N_m, nil)$ and $\texttt{Infer}(\underline{i}\, N_1 \cdots N_m) = ((\omega^{\underline{i-1}}.\varphi_1 \to \cdots \to \varphi_m \to \alpha.nil) \wedge \Gamma^1 \wedge \cdots \wedge \Gamma^m, \alpha)$.

- Case $(\Gamma, v \to \varphi)$. Let $T = \Gamma \Rightarrow v \to \varphi$.

  By hypothesis $(\Gamma, v \to \varphi) \in \mathcal{P}$, thus T is principal.

  If $\Gamma = nil$ and $v = \omega$ then $T' = nil \Rightarrow \varphi$ is principal and, by IH, $\texttt{Recon}(nil, \varphi) = (N, nil)$, N a β-nf such that $\texttt{Infer}(N) = (nil, \varphi)$. Thus, $\texttt{Recon}(nil, \omega \to \varphi) = (\lambda.N, nil)$ and $\texttt{Infer}(\lambda.N) = (nil, \omega \to \varphi)$.

  Otherwise, $T' = v.\Gamma \Rightarrow \varphi$ is principal. By IH, $\texttt{Recon}(v.\Gamma, \varphi) = (N, nil)$, N a β-nf such that $\texttt{Infer}(N) = (v.\Gamma, \varphi)$. Hence, $\texttt{Recon}(\Gamma, v \to \varphi) = (\lambda.N, nil)$ and $\texttt{Infer}(\lambda.N) = (\Gamma, v \to \varphi)$.

$\square$

Observe that, by Lemma 15, we have that:

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

Thus, $\mathcal{P}$ is the set of all, and only, principal typings for β-nfs in SM_r. Therefore,

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

4 Conclusion

We intend to add intersection types for two calculi with explicit substitutions, $\lambda\sigma$ and λs_e, both with de Bruijn indices. The investigation of systems with de Bruijn indices, as the one presented here, helps to have a deep understanding of their behaviour in this notation. There are works on intersection types and explicit substitution, e.g. [LLDDvB04], but no work for systems where the composition of substitutions is allowed.

The restriction in the system in [SM96a] prevents it, and consequently the system introduced here, to have SR in the usual sense, in contrast with the system in [VAK08]. However, every β-nf is typeable in the system in [SM96a], a property that does not hold for the simply typed system. A characterisation of PT for β-nfs is then given. We have introduced a de Bruijn version of the typing system with similar characteristics as a first step towards some extended systems in which PT depends on more complex syntactic operations such as expansion. As future work, we will introduce a de Bruijn version for systems such as the ones studied in [CDV80] and [RV84] and try to add similar systems to both $\lambda\sigma$ and λs_e.

References

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *J. func. program.*, 1(4):375–416, 1991.

[ARK01] M. Ayala-Rincón and F. Kamareddine. Unification via the λs_e-Style of Explicit Substitution. *Logical journal of the IGPL*, 9(4):489–523, 2001.

[Bak95] S. van Bakel. Intersection Type Assignment Systems. *Theoret. comput. sci.*, 151:385-435, 1995.

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. symbolic logic*, 48:931–940, 1983.

[Bar84] H. Barendregt, *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, 1984.

[dB72] 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.

[dB78] 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.

[CW04a] S. Carlier and J. B. Wells. Type Inference with Expansion Variables and Intersection Types in System E and an Exact Correspondence with β-reduction. In *Proc. of PPDP '04*, pp. 132–143. ACM, 2004.

[CW04b] S. Carlier and J. B. Wells. Expansion: the Crucial Mechanism for Type Inference with Intersection Types: a Survey and Explanation. In *ITRS '04 workshop*, 2004.

[CDC78] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische logik*, 19:139–156, 1978.

[CDC80] M. Coppo and M. Dezani-Ciancaglini. An Extension of the Basic Functionality Theory for the λ-Calculus. *Notre dame j. formal logic*, 21(4):685–693, 1980.

[CDV80] M. Coppo, M. Dezani-Ciancaglini and B. Venneri. Principal Type Schemes and λ-calculus Semantics. In J.P. Seldin and J.R. Hindley (eds), *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism*, pp. 536–560. Academic Press, 1980.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*, vol. 1. North Holland, 1958.

[DG94] F. Damiani and P. Giannini. A Decidable Intersection Type System based on Relevance. In *Proc. of TACS94*, *LNCS* 789:707725. Springer-Verlag, 1994.

[Hi97] J. R. Hindley. *Basic Simple Type Theory.* Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[KN07] F. Kamareddine and K. Nour. A completeness result for a realisability semantics for an intersection type system. *Annals pure and appl. logic*, 146:180–198, 2007.

[KR95] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with Explicit Substitutions. In *Proc. of PLILP'95*, *LNCS* 982:45–62. Springer, 1995.

[KW04] A.J. Kfoury and J.B. Wells. Principality and type inference for intersection types using expansion variables. *Theoret. comput. sci.*, 311(1–3):1–70, 2004.

[Kri93] J-L. Krivine. *Lambda-calculus, types and models.* Ellis Horwood, 1993.

[LLDDvB04] S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Inform. and comput.*, 189(1):1742, 2004.

[Mil78] R. Milner. A theory of type polymorphism in programming. *J. comput. and system sci.*, 17(3):348–375, 1978.

[NGdV94] R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. *Selected papers on Automath.* North-Holland, 1994.

[Pot80] 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.

[RV84] S. Ronchi Della Rocca and B. Venneri. Principal Type Scheme for an Extended Type Theory. *Theoret. comput. sci.*, 28:151–169, 1984.

[Roc88] S. Ronchi Della Rocca. Principal Type Scheme and Unification for Intersection Type Discipline. *Theoret. comput. sci.*, 59:181–209, 1988.

[SM96a] E. Sayag and M. Mauny. Characterization of principal type of normal forms in intersection type system. In *Proc. of FSTTCS'96*, LNCS, 1180:335–346. Springer, 1996.

[SM96b] 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.

[SM97] E. Sayag and M. Mauny. Structural properties of intersection types. In *Proc. of LIRA'97*, pp. 167-175. Novi Sad, Yugoslavia, 1997.

[VAK08] D. Ventura and M. Ayala-Rincón and F. Kamareddine. Intersection Type System with de Bruijn Indices. `http://www.mat.unb.br/~ayala/ITdeBruijn.pdf` - revised version to appear in *The many sides of logic*. Studies in logic, College publications. London, 2009.

[We02] J.B. Wells. The essence of principal typings. In *Proc. of ICALP 2002*, LNCS, 2380:913–925. Springer, 2002.