Characterising strongly normalising intuitionistic sequent terms

J. Espirito-Santo[1], S. Ghilezan[2], J. Ivetić[2]

[1] Mathematics Department, University of Minho, Portugal
jes@math.uminho.pt
[2] Faculty of Engineering, University of Novi Sad, Serbia
gsilvia@uns.ns.ac.yu jelena@imft.ftn.ns.ac.yu

Abstract. This paper gives a characterisation, via intersection types, of the strongly normalising terms of an intuitionistic sequent calculus (where LJ easily embeds). The soundness of the typing system is reduced to that of a well known typing system with intersection types for the ordinary λ-calculus. The completeness of the typing system is obtained from subject expansion at root position. This paper's sequent term calculus integrates smoothly the λ-terms with generalised application or explicit substitution. Strong normalisability of these terms *as sequent terms* characterises their typeability in certain "natural" typing systems with intersection types. The latter are in the natural deduction format, like systems previously studied by Matthes and Lengrand *et al.*, except that they do not contain any extra, exceptional rules for typing generalised applications or substitution.

Introduction

The recent interest in the Curry-Howard correspondence for sequent calculus [9, 2, 5, 8, 6] made it clear that the computational content of sequent derivations and cut-elimination can be expressed through an extension of the λ-calculus, where the construction that interprets cut subsumes both explicit substitution and an enlarged concept of application, exhibiting the features of "multiarity" and "generality" [8]. The sequent calculus acts relatively to such calculus of sequent terms as a typing system, and the ensuing notion of typeability is sufficient, but not necessary, for strong normalisability.

This situation is well-known in the context of the ordinary λ-calculus, where simple-typeability is sufficient, but not necessary, for strong β-normalisability. A form of getting a characterisation of strongly normalising λ-terms is to extend the typing system with intersection types. For this reason intersection type assignment systems were introduced into λ-calculus in the late 1970s by Coppo and Dezani [3], Pottinger [15] and Sallé [18]. Intersection types completely characterise strong normalisation in lambda calculus (see [1]).

In this paper we seek a characterisation of strongly normalising sequent terms via intersection types. We first introduce, following [6], an extension of the λ-calculus named λ^{Gtz} (after Gentzen) corresponding to a sequent calculus for intuitionistic implicational logic, equipped with reduction rules for cut-elimination.

The typing system is from the beginning equipped with intersection types, following [4]. The correctness of the typing system is obtained by a reduction to the correctness of the system $\mathcal{D}$ [12]. The completeness of the typing system is obtained as a corollary to subject expansion at root position.

A recent topic of research is the use of intersection types for the characterisation of strong-normalisability in extensions of the λ-calculus with generalised applications or explicit substitutions [14, 13, 11]. A common symptom of these works is the need to throw in the typing system some extra, exceptional rules for typing generalised applications or substitutions. This breaks somehow the harmony observed in the ordinary λ-calculus between typeability induced by intersection types and strong β-normalisability. One may wonder whether, in the extended scenario with generalised applications or explicit substitutions the blame for the slight mismatch is on some insufficiency of the intersection types technique, or on some insufficiency of the reduction relations causing too many terms to be terminating.

It turns out that, because of its expressive power, λ^{Gtz} is a good tool to analyze this question. A simple analysis of our main characterisation result shows that strong normalisability *as sequent terms* (*i.e.* inside λ^{Gtz}) of λ-terms with generalised applications or explicit substitutions characterises their typeability in certain "natural" typing systems with intersection types. The latter are in the natural deduction format, like systems previously studied in [14, 13], except that they do not contain any extra, exceptional rules for typing generalised applications or substitution. So one is led to compare the behavior under reduction of λ-terms with generalised applications or explicit substitutions inside λ^{Gtz} and inside their native system ΛJ [10] or $\lambda\mathtt{x}$ [17]. We conclude that the problem in ΛJ is that we cannot form explicit substitutions, and in $\lambda\mathtt{x}$ is that we cannot compose substitutions.

The paper is organized as follows. Section 1 presents the syntax of the untyped λ^{Gtz} calculus. Section 2 introduces an intersection type system $\lambda^{\mathsf{Gtz}}\cap$. Strong normalisation is proved in Section 3, and characterisation of strong normalisation is given in Section 4. In Section 5, the relation between λ^{Gtz} calculus and calculi with generalised applications and explicit substitutions is discussed. Finally, Section 6 concludes this paper.

1 Syntax of λ^{Gtz}

The abstract syntax of λ^{Gtz} is given by:

$$\begin{array}{ll} \text{(Terms)} & t, u, v ::= x \mid \lambda x.t \mid tk \\ \text{(Contexts)} & k ::= \widehat{x}.t \mid u :: k, \end{array}$$

where x ranges over a denumerable set of term variables.

Terms are either variables, abstractions or *cuts* tk. A context is either a *selection* or a *context cons(tructor)*. Terms and contexts are together referred to as the *expressions* and will be ranged over by E. In $\lambda x.t$ and $\widehat{x}.t$, t is the scope of

the binders λx and $\widehat{x}$, respectively. Free variables in λ^{Gtz} calculus are those that are not bound neither by abstraction nor by selection operator and Barendregt's convention should be applied in both cases. In order to avoid parentheses, we let the scope of binders extend to the right as much as possible.

According to the form of k, a cut may be an explicit substitution $t(\widehat{x}.v)$ or a multiary generalised application $t(u_1 :: \cdots :: u_m :: \widehat{x}.v)$ $(m \geq 1)$. In the last case, if $m = 1$, we get a generalised application $t(u :: \widehat{x}.v)$; if $v = x$, we get a multiary application $t[u_1, \cdots, u_m]$ (think of $\widehat{x}.x$ as the empty list of arguments); a combination of constraints $m = 1$ and $v = x$ brings cuts to the form of an ordinary application.

Reduction rules of λ^{Gtz} are as follows:

$$\begin{array}{ll} (\beta) & (\lambda x.t)(u :: k) \to u(\widehat{x}.tk) \\ (\pi) & (tk)k' \to t(k@k') \\ (\sigma) & t(\widehat{x}.v) \to v[x := t] \\ (\mu) & \widehat{x}.xk \to k, \text{ if } x \notin k \end{array}$$

where $t[x := u]$ (or $k[x := u]$) denotes meta-substitution, and $k@k'$ is defined by $(u :: k)@k' = u :: (k@k')$ and $(\widehat{x}.v)@k' = \widehat{x}.vk'$.

The rules β, π, and σ reduce cuts to the trivial form $y(u_1 :: \cdots u_m :: \widehat{x}.v)$, for some $m \geq 1$, which represents a sequence of left introductions. Rule β generates a substitution, and rule σ executes a substitution in the meta-level. Rule π generalises the permutative conversion of the λ-calculus with generalised applications. Rule μ has a structural character, and either performs a trivial substitution in the reduction $t(\widehat{x}.xk) \to tk$, or minimizes the use of the generality feature in the reduction $t(u_1 \cdots u_m :: \widehat{x}.xk) \to t(u_1 \cdots u_m :: k)$.

$\beta\pi\sigma$-normal forms of λ^{Gtz} are:

$$\begin{array}{ll} \text{(Terms)} & t_{nf}, u_{nf}, v_{nf} = x \mid \lambda x.t_{nf} \mid x(u_{nf} :: k_{nf}) \\ \text{(Contexts)} & k_{nf} = \widehat{x}.t_{nf} \mid t_{nf} :: k_{nf} \end{array}$$

λ^{Gtz} is a flexible system for representing logical derivations in the sequent calculus format and studying cut-elimination. The inference rules of LJ axiom, right introduction, left introduction, and cut, are represented by the constructions x, $\lambda x.t$, $y(u :: \widehat{x}.v)$, and $t(\widehat{x}.v)$, respectively. The $\beta\pi\sigma$-normal forms correspond to the multiary, cut-free, sequent terms of [19]. See [6] for more on λ^{Gtz}.

2 Intersection types for λ^{Gtz}

Definition 1. *The set of types Types, ranged over by $A, B, C, ..., A_1, ...$, is inductively defined as follows:*

$$A, B \quad ::= \quad p \mid A \to B \mid A \cap B$$

where p ranges over a denumerable set of type atoms.

Definition 2. *(i) Pre-order $\leq$ over the set of types is the smallest relation that satisfies the following properties:*

1. $A \leq A$
2. $A \cap B \leq A$ *and* $A \cap B \leq B$
3. $(A \to B) \cap (A \to C) \leq A \to (B \cap C)$
4. $A \leq B$ *and* $B \leq C$ *implies* $A \leq C$
5. $A \leq B$ *and* $A \leq C$ *implies* $A \leq B \cap C$
6. $A' \leq A$ *and* $B \leq B'$ *implies* $A \to B \leq A' \to B'$

(ii) Two types are equivalent, $A \sim B$, if and only if $A \leq B$ and $B \leq A$.

In this paper, we will consider types modulo the equivalence relation.

Remark 3. The equivalence $(A \to B) \cap (A \to C) \sim A \to (B \cap C)$, or more generally $\cap(\cap A_k \to B_i) \sim \cap A_k \to \cap B_i$, follows from the given set of rules, and will be used in the sequel.

Definition 4. *(i) A basic type assignment is an expression of the form $x : A$, where x is a term variable and A is a type.*
(ii) A basis Γ is a set of basic type assignments, where all term variables are different.
(iii) There are two kinds of type assignment:
- *$\Gamma \vdash t : A$ for typing terms;*
- *$\Gamma; B \vdash k : A$ for typing contexts.*

The following typing system for λ^{Gtz} is named $\lambda^{\mathsf{Gtz}}\cap$. In Ax, $\to_L$, and Cut $\cap A_i = A_1 \cap \cdots \cap A_n$, for some $n \geq 1$.

$$\frac{j \in \{1, \cdots, n\}}{\Gamma, x : \cap A_i \vdash x : A_j}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_R) \qquad \frac{\Gamma \vdash u : A_i,\ \ \forall i \in \{1, \cdots, n\} \qquad \Gamma; B \vdash k : C}{\Gamma; \cap A_i \to B \vdash u :: k : C}\ (\to_L)$$

$$\frac{\Gamma \vdash t : A_i,\ \forall i \in \{1, \cdots, n\} \qquad \Gamma; \cap A_i \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash v : B}{\Gamma; A \vdash \widehat{x}.v : B}\ (Sel)$$

By taking $n = 1$ in Ax, $\to_L$, and Cut we get the typing rules of [6] for assigning simple types.

Notice that in this typing system there are no separate rules for the right introduction of intersections. The management of intersection is built in the other rules.

Proposition 5 (Admissible rule - $(\cap_L)$).

(i) *If* $\Gamma, x : A_i \vdash t : B$, *for some* i, *then* $\Gamma, x : \cap A_i \vdash t : B$.

(ii) *If* $\Gamma, x : A_i; C \vdash k : B$, *for some* i, *then* $\Gamma, x : \cap A_i; C \vdash k : B$.

Proof. By mutual induction on the derivation. □

Proposition 6 (Basis expansion).

(i) $\Gamma \vdash t : A \;\Leftrightarrow\; \Gamma, x : B \vdash t : A$ *and* $x \notin Fv(t)$.
(ii) $\Gamma; C \vdash k : A \;\Leftrightarrow\; \Gamma, x : B; C \vdash k : A$ *and* $x \notin Fv(k)$.

Definition 7.

$$\begin{aligned}\Gamma_1 \cap \Gamma_2 = &\{x : A | x : A \in \Gamma_1 \;\&\; x \notin \Gamma_2\} \\ &\cup \{x : A | x : A \in \Gamma_2 \;\&\; x \notin \Gamma_1\} \\ &\cup \{x : A \cap B | x : A \in \Gamma_1 \;\&\; x : B \in \Gamma_2\}.\end{aligned}$$

Proposition 8 (Bases intersection).

(i) $\Gamma_1 \vdash t : A \;\Rightarrow\; \Gamma_1 \cap \Gamma_2 \vdash t : A$.
(ii) $\Gamma_1; B \vdash k : A \;\Rightarrow\; \Gamma_1 \cap \Gamma_2; B \vdash k : A$.

Proposition 9 (Generation lemma - GL).

(i) $\Gamma \vdash x : A$ *iff* $x : \cap A_i \in \Gamma$ *and* $A \equiv A_i$, *for some* i.
(ii) $\Gamma \vdash \lambda x.t : A$ *iff* $A \equiv B \to C$ *and* $\Gamma, x : B \vdash t : C$.

(iii) $\Gamma; A \vdash \widehat{x}.t : B$ *iff* $\Gamma, x : A \vdash t : B$.

(iv) $\Gamma \vdash tk : A$ *iff there is a type* $B \equiv \cap B_i$ *such that* $\Gamma \vdash t : B_i$ *for all* i, *and* $\Gamma; \cap B_i \vdash k : A$.

(v) $\Gamma; D \vdash t :: k : C$ *iff* $D \equiv \cap A_i \to B$, *and* $\Gamma; B \vdash k : C$ *and* $\Gamma \vdash t : A_i$ *for all* i.

Proof. The proof is straightforward since all rules are syntax-directed. □

Lemma 10 (Substitution and append lemma).

(i) *If* $\Gamma, x : \cap A_i \vdash t : B$ *and* $\Gamma \vdash u : A_i$, *for each* i, *then* $\Gamma \vdash t[x := u] : B$.

(ii) *If* $\Gamma, x : \cap A_i; C \vdash k : B$ *and* $\Gamma \vdash u : A_i$, *for each* i, *then* $\Gamma; C \vdash k[x := u] : B$.

(iii) *If* $\Gamma; B \vdash k : C_i$, $\forall i$, *and* $\Gamma; \cap C_i \vdash k : A$, *then* $\Gamma; B \vdash k@k' : A$.

Proof. (i) and (ii) is proved by simultaneous induction on t and k. (iii) is proved by induction on k. □

Theorem 11 (Subject Reduction). *If $\Gamma \vdash t : A$ and $t \to t'$, then $\Gamma \vdash t' : A$.*

Proof. The proof employs the previous lemma. It is omitted because of the lack of space. □

Example 12. In λ-calculus, the term $\lambda x.xx$ has the type $(A \cap (A \to B)) \to B$. The corresponding term in λ^{Gtz}-calculus is $\lambda x.x(x :: \widehat{y}.y)$. Although being a normal form this term is not typeable in the simply typed λ^{Gtz}-calculus. It is typeable in $\lambda^{\mathsf{Gtz}}\cap$ in the following way:

$$\dfrac{\dfrac{\dfrac{}{x : A \cap (A \to B) \vdash x : A \to B}\,Ax \quad \dfrac{\dfrac{}{x : A \cap (A \to B) \vdash x : A}\,Ax \quad \dfrac{\dfrac{}{x : A \cap (A \to B), y : B \vdash y : B}\,Ax}{x : A \cap (A \to B); B \vdash \widehat{y}.y : B}\,Sel}{x : A \cap (A \to B); A \to B \vdash (x :: \widehat{y}.y) : B}\,{\to_L}}{x : A \cap (A \to B) \vdash x(x :: \widehat{y}.y) : B}\,Cut}{\vdash \lambda x.x(x :: \widehat{y}.y) : (A \cap (A \to B)) \to B}\,{\to_R}\,.$$

3 Typeability ⇒ SN

In order to prove strong normalisation for the $\lambda^{\mathsf{Gtz}}\cap$ system, we connect it with the well-known system $\mathcal{D}$ for λ-calculus via an appropriate mapping, and then use strong normalisation theorem for λ-terms typeable in $\mathcal{D}$ system.

λ-terms are given by

$$M, N, P ::= x \mid \lambda x.M \mid MN$$

and equipped with

$$\begin{array}{ll} (\beta) & (\lambda x.M)N \to M[x := N] \\ (\pi_1) & (\lambda x.M)NP \to (\lambda x.MP)N \\ (\pi_2) & M((\lambda x.P)N) \to (\lambda x.MP)N \end{array}$$

without clash of free and bound variables (Barendregt's convention). We let $\pi = \pi_1 \cup \pi_2$.

Proposition 13. *If a λ-term M is β-SN, then M is $\beta\pi$-SN.*

Proof. This is Theorem 2 in [7]. □

The following typing system for λ is named $\mathcal{D}$ in [12].

$$\frac{}{\Gamma, x : A \vdash x : A}\ Ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to I \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \to E$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \cap B} \cap I \qquad \frac{\Gamma \vdash M : A_1 \cap A_2}{\Gamma \vdash M : A_i} \cap E$$

Lemma 14. *The following rules are admissible in $\mathcal{D}$:*

$$\frac{\Gamma \vdash M : A \quad \Gamma \subseteq \Gamma'}{\Gamma' \vdash M : A}\ Weak \qquad \frac{\Gamma \vdash N : A \quad \Gamma, x : A \vdash M : B}{\Gamma \vdash M[x := N] : B}\ Subst$$

Proposition 15 (SN). *If a λ-term M is typeable in $\mathcal{D}$, then M is β-SN.*

Proof. A result from [16], [12]. □

We define a mapping F from λ^{Gtz} to λ. The idea is as follows. If $F(t) = M$, $F(u_i) = N_i$ and $F(v) = P$, then $t(u_1 :: u_2 :: \widehat{x}.v)$, say, is mapped to $(\lambda x.P)(MN_1N_2)$. Formally, a mapping $F : \lambda^{\mathsf{Gtz}} - Terms \longrightarrow \lambda - Terms$ is defined simultaneously with an auxiliary mapping $F' : \lambda - Terms \times \lambda^{\mathsf{Gtz}} - Contexts \longrightarrow \lambda - Terms$ as follows:

$$\begin{aligned} F(x) &= x \\ F(\lambda x.t) &= \lambda x.F(t) \\ F(tk) &= F'(F(t), k) \end{aligned}$$

$$\begin{aligned} F'(N, \widehat{x}.t) &= (\lambda x.F(t))N \\ F'(N, u :: k) &= F'(NF(u), k) \end{aligned}$$

Proposition 16. *If $\lambda^{\mathsf{Gtz}}\cap$ proves $\Gamma \vdash t : A$, then $\mathcal{D}$ proves $\Gamma \vdash F(t) : A$.*

Proof. The proposition is proved together with the claim: if $\lambda^{\mathsf{Gtz}}\cap$ proves $\Gamma; A \vdash k : B$ and $\mathcal{D}$ proves $\Gamma \vdash N : A$, then $\mathcal{D}$ proves $\Gamma \vdash F'(N, k) : B$. The proof is by simultaneous induction on derivations Π_1 and Π_2 of $\Gamma \vdash t : A$ and $\Gamma; A \vdash k : B$, respectively. Cases according to the last typing rule used.

The case (Ax) is obtained by the corresponding Ax in $\mathcal{D}$ together with the $\cap E$. The case $\rightarrow R$, is easy, because $\mathcal{D}$ has the corresponding typing rule.

Case (Cut). Π_1 has the shape

$$\frac{\begin{matrix}\Pi_{11i} \\ \Gamma \vdash t : A_i, \forall i\end{matrix} \quad \begin{matrix}\Pi_{12} \\ \Gamma; \cap A_i \vdash k : B\end{matrix}}{\Gamma \vdash tk : B}\ (Cut)$$

By IH(Π_{11i}), $\mathcal{D}$ proves $\Gamma \vdash F(t) : A_i$. By repeated application of $\cap I$, $\mathcal{D}$ proves $\Gamma \vdash F(t) : A_i$. By IH(Π_{12}), $\mathcal{D}$ proves $\Gamma \vdash F'(F(t), k) : B$. This is what we want, since $F'(F(t), k) = F(tk)$.

Case (Sel). Π_2 has the shape

$$\frac{\begin{matrix}\Pi_{21} \\ \Gamma, x : A \vdash t : B\end{matrix}}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

Suppose $\mathcal{D}$ proves $\Gamma \vdash N : A$. Then in $\mathcal{D}$ one has

$$\dfrac{\dfrac{\overset{IH}{\Gamma, x : A \vdash F(t) : B}}{\Gamma \vdash \lambda x.F(t) : A \to B} \to I \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.F(t))N : B} \ (\to E)$$

This is what we want, since $F'(N, \widehat{x}.t) = (\lambda x.F(t))N$.

Case $(\to L)$. Π_2 has the shape

$$\dfrac{\overset{\Pi_{21i}}{\Gamma \vdash u : A_i, \forall i} \qquad \overset{\Pi_{22}}{\Gamma; B \vdash k : C}}{\Gamma; \cap A_i \to B \vdash u :: k : C} \ (\to L)$$

Suppose $\mathcal{D}$ proves $\Gamma \vdash N : \cap A_i \to B$. By IH (Π_{21i}) $\mathcal{D}$ proves $\Gamma \vdash F(u) : A_i, \forall i$; therefore, by repeated application of $\cap I$, $\mathcal{D}$ proves $\Gamma \vdash F(u) : \cap A_i$. Then in $\mathcal{D}$ one has

$$\dfrac{\Gamma \vdash N : \cap A_i \to B \quad \Gamma \vdash F(u) : \cap A_i}{\Gamma \vdash NF(u) : B} \ (\to E)$$

Hence, by IH(Π_{22}), $\mathcal{D}$ proves $\Gamma \vdash F'(NF(u), k) : C$. This is what we want, since $F'(NF(u), k) = F'(N, u :: k)$. □

Proposition 17. *For all $t \in \lambda^{\mathsf{Gtz}}$, if $F(t)$ is $\beta\pi$-SN, then t is $\beta\pi\sigma\mu$-SN.*

Proof. Consequence of the following properties of F: (i) if $t \to_{\beta\pi} u$ in λ^{Gtz}, then $F(t) \to_{\pi}^{+} F(u)$ in λ; (ii) if $t \to_{\sigma\mu} u$ in λ^{Gtz}, then $F(t) \to_{\beta} F(u)$ in λ. □

Theorem 18 (Typeability ⇒ SN). *If a λ^{Gtz}-term t is typeable in $\lambda^{\mathsf{Gtz}}\cap$, then t is $\beta\pi\sigma\mu$-SN.*

Proof. Suppose t is typeable in $\lambda^{\mathsf{Gtz}}\cap$. Then, by Proposition 16, $F(t)$ is typeable in $\mathcal{D}$. So, by Proposition 15, $F(t)$ is β-SN. Hence, by Proposition 13, $F(t)$ is $\beta\pi$-SN. Finally, by Proposition 17, t is $\beta\pi\sigma\mu$-SN. □

4 SN ⇒ Typeability

4.1 Typeability of normal forms

Proposition 19. *$\beta\pi\sigma$-normal forms of λ^{Gtz} calculus are typeable in $\lambda^{\mathsf{Gtz}}\cap$ system. Hence so are $\beta\pi\sigma\mu$-normal forms.*

Proof. By simultaneous induction on the structure of $\beta\pi\sigma$-normal terms and contexts.

- Basic case: Every variable is typeable.
- $\lambda x.t_{nf}$ is typeable.
  By IH, t_{nf} is typeable, so $\Gamma \vdash t_{nf} : B$. We examine two cases:

Case 1. If $x : A \in \Gamma$, then $\Gamma = \Gamma', x : A$ and we can assign the following type to $\lambda x.t_{nf}$:

$$\frac{\Gamma', x : A \vdash t_{nf} : B}{\Gamma' \vdash \lambda x.t_{nf} : A \to B.} (\to_R)$$

Case 2. If $x : A \notin \Gamma$, then by Proposition 6 we get $\Gamma, x : A \vdash t_{nf} : B$ thus concluding

$$\frac{\Gamma, x : A \vdash t_{nf} : B}{\Gamma \vdash \lambda x.t_{nf} : A \to B.} (\to_R)$$

- $\widehat{x}.t_{nf}$ is typeable.
  Proof is very similar to the previous one.
- $t_{nf} :: k_{nf}$ is typeable.
  By IH t_{nf} and k_{nf} are typeable, i.e. $\Gamma_1 \vdash t_{nf} : A$ and $\Gamma_2; B \vdash k_{nf} : C$. Then, by Proposition 8 we get $\Gamma_1 \cap \Gamma_2 \vdash t_{nf} : A$ and $\Gamma_1 \cap \Gamma_2; B \vdash k_{nf} : C$, so we assign the following type to $t_{nf} :: k_{nf}$:

$$\frac{\Gamma_1 \cap \Gamma_2 \vdash t_{nf} : A \quad \Gamma_1 \cap \Gamma_2; B \vdash k_{nf} : C}{\Gamma_1 \cap \Gamma_2; A \to B \vdash t_{nf} :: k_{nf} : C.} (\to_L)$$

- $x(t_{nf} :: k_{nf})$ is typeable.
  By IH and the previous case, context $t_{nf} :: k_{nf}$ is typeable, i.e. $\Gamma; A \to B \vdash t_{nf} :: k_{nf} : C$. We examine 3 cases:

Case 1. If $x : A \to B \in \Gamma$, then:

$$\frac{\dfrac{}{\Gamma \vdash x : A \to B} (Ax) \qquad \Gamma; A \to B \vdash t_{nf} :: k_{nf} : C}{\Gamma \vdash x(t_{nf} :: k_{nf}) : C.} (Cut)$$

Case 2. If $x : D \in \Gamma$, then $\Gamma = \Gamma', x : D$ and we can expand basis of $x : A \to B \vdash x : A \to B$ to $\Gamma', x : D \cap (A \to B) \vdash x : A \to B$ using Propositions 5 and 6. Also, by Proposition 5, we can write $\Gamma', x : D \cap (A \to B); A \to B \vdash t_{nf} :: k_{nf} : C$. Now, the corresponding type assignment is:

$$\frac{\Gamma', x : D \cap (A \to B) \vdash x : A \to B \qquad \Gamma', x : D \cap (A \to B); A \to B \vdash t_{nf} :: k_{nf} : C}{\Gamma', x : D \cap (A \to B) \vdash x(t_{nf} :: k_{nf}) : C.} (Cut)$$

Case 3. If x isn't declared at all, by Proposition 6 we get $\Gamma, x : A \to B; A \to B \vdash t_{nf} :: k_{nf} : C$ from $\Gamma; A \to B \vdash t_{nf} :: k_{nf} : C$, and then conclude:

$$\frac{\dfrac{}{\Gamma, x : A \to B \vdash x : A \to B} (Ax) \qquad \Gamma, x : A \to B; A \to B \vdash t_{nf} :: k_{nf} : C}{\Gamma, x : A \to B \vdash x(t_{nf} :: k_{nf}) : C.} (Cut)$$

□

4.2 Subject expansion at root position

Lemma 20. *If* $\Gamma \vdash u(\widehat{x}.tk) : A$ *and* $x \notin Fv(u) \cup Fv(k)$, *then* $\Gamma \vdash (\lambda x.t)(u :: k) : A$.

Proof. $\Gamma \vdash u\widehat{x}.(tk) : A$ implies, by $GL(iv)$, that there is a type $B \equiv \cap B_i$, such that $\Gamma \vdash u : B_i$, for all i and $\Gamma; \cap B_i \vdash \widehat{x}.(tk) : A$. Further, this implies, by $GL(iii)$, that $\Gamma, x : \cap B_i \vdash tk : A$ so then there is a $C \equiv \cap C_j$ such that $\Gamma, x : \cap B_i \vdash t : C_j$ for all j and $\Gamma, x : \cap B_i; \cap C_j \vdash k : A$. By assumption, the variable x is not free in k, so using Proposition 6 we can write the previous sequent as $\Gamma; \cap C_j \vdash k : A$. Now, because of the equivalence $\cap(\cap B_i \to C_j) \sim \cap B_i \to \cap C_j$, we have:

$$\dfrac{\dfrac{\Gamma, x : \cap B_i \vdash t : C_j,\ \forall j}{\Gamma \vdash \lambda x.t : \cap B_i \to C_j,\ \forall j}\,(\to_R) \quad \dfrac{\Gamma \vdash u : B_i,\ \forall i \quad \Gamma; \cap C_j \vdash k : A}{\Gamma; \cap B_i \to \cap C_j \vdash u :: k : A}\,(\to_L)}{\Gamma \vdash (\lambda x.t)(u :: k) : A.}\,(Cut)$$

□

Lemma 21 (Inverse substitution lemma).

(i) Let $\Gamma \vdash v[x := t] : A$, *and let* t *be typeable. Then there is a basis* Γ' *and a type* $B \equiv \cap B_i$, *such that* $\Gamma', x : \cap B_i \vdash v : A$ *and for all* i, $\Gamma' \vdash t : B_i$.

(ii) Let $\Gamma; C \vdash k[x := t] : A$, *and let* t *be typeable. Then there is a basis* Γ' *and a type* $B \equiv \cap B_i$, *such that* $\Gamma', x : \cap B_i; C \vdash k : A$ *and for all* i, $\Gamma' \vdash t : B_i$.

Proof. By simultaneous induction on the structure of the term v and the context k. □

Lemma 22 (Inverse append lemma). *If* $\Gamma; B \vdash k@k' : A$ *then there is a type* $C \equiv \cap C_i$ *such that* $\Gamma; B \vdash k : C_i$, $\forall i$ *and* $\Gamma; \cap C_i \vdash k' : A$.

Proof. By induction on the structure of k.

- Basic case: $k \equiv \widehat{x}.v$
  In this case $k@k' = (\widehat{x}.v)@k' = \widehat{x}.vk'$. From $\Gamma; B \vdash \widehat{x}.vk' : A$, by $GL(iii)$, we have that $\Gamma, x : B \vdash vk' : A$. Then, by $GL(iv)$, there is a $C \equiv \cap C_i$ such that $\Gamma, x : B \vdash v : C_i$, $\forall i$ and $\Gamma, x : B; \cap C_i \vdash k' : A$. From the first sequent we get $\Gamma; B \vdash \widehat{x}.v : C_i$, $\forall i$. From the second one, considering that x is not free in k', we get $\Gamma; \cap C_i \vdash k' : A$.
- $k \equiv u :: k''$
  In this case, $k@k' = (u :: k'')@k' = u :: (k''@k')$. From $\Gamma; B \vdash u :: (k''@k') : A$, by $GL(v)$, $B \equiv \cap C_i \to D$, $\Gamma; D \vdash k''@k' : A$ and $\Gamma \vdash u : C_i$, for all i. From the first sequent, by IH, we get some $E \equiv \cap E_j$ such that $\Gamma; D \vdash k'' : E_j$, $\forall j$ and $\Gamma; \cap E_j \vdash k' : A$. Finally, for each j,

  $$\dfrac{\Gamma \vdash u : C_i,\ \forall i \quad \Gamma; D \vdash k'' : E_j}{\Gamma; \cap C_i \to D(\equiv B) \vdash u :: k'' : E_j}\,(\to_L)$$

  so the proof is completed.

□

Proposition 23 (Subject expansion at root position). *If $t \to t'$, t is the contracted redex and t' is typeable in $\lambda^{\mathsf{Gtz}}\cap$, then t is typeable in $\lambda^{\mathsf{Gtz}}\cap$.*

Proof. We examine four different cases, according to the applied reduction.

- (β) : Directly follows from Lemma 20.
- (σ) : We should show that typeability of $t' \equiv v[x := u]$ leads to typeability of $t \equiv u\widehat{x}.v$.
  Assume that $\Gamma \vdash v[x := u] : A$. By Lemma 21 there are a Γ' and a $B \equiv \cap B_i$ such that $\Gamma' \vdash u : B_i,\ \forall i$ and $\Gamma', x : \cap B_i \vdash v : A$. Now
  $$\dfrac{\Gamma' \vdash u : B_i,\ \forall i \quad \dfrac{\Gamma', x : \cap B_i \vdash v : A}{\Gamma'; \cap B_i \vdash \widehat{x}.v : A}\,(Sel)}{\Gamma' \vdash u\widehat{x}.v : A.}\,(Cut)$$
- (π) : We should show that typeability of $t(k@k')$ implies typeability of $(tk)k'$. $\Gamma \vdash t(k@k') : A$, by $GL(iv)$ yields that there is $B \equiv \cap B_i$ such that $\Gamma \vdash t : B_i$, $\forall i$, and $\Gamma; \cap B_i \vdash k@k' : A$. By applying Lemma 22 on previous sequent, we get $\Gamma; \cap B_i \vdash k : C_j$, $\forall j$, and $\Gamma; \cap C_j \vdash k' : A$, for some type $C \equiv \cap C_j$. Now, for each j,
  $$\dfrac{\Gamma \vdash t : B_i,\ \forall i \quad \Gamma; \cap B_i \vdash k : C_j}{\Gamma \vdash tk : C_j}\,(Cut)$$
  So $\Gamma \vdash tk : C_j$, $\forall j$. We obtain $\Gamma \vdash (tk)k' : A$ with a further application of (Cut).
- (μ) : It should be shown that typeability of k implies typeability of $\widehat{x}.xk$. Assume $\Gamma; B \vdash k : A$. Since $x \notin k$ we can suppose that $x \notin \Gamma$, and by using Proposition 6 write $\Gamma, x : B; B \vdash k : A$. Now
  $$\dfrac{\dfrac{\Gamma, x : B \vdash x : B \quad \Gamma, x : B; B \vdash k : A}{\Gamma, x : B \vdash xk : A}\,(Cut)}{\Gamma; B \vdash \widehat{x}.xk : A.}\,(Sel)$$

□

Theorem 24 (SN ⇒ typeability). *All strongly normalising $(\beta\sigma\pi - SN)$ expressions are typeable in $\lambda^{\mathsf{Gtz}}\cap$ system.*

Proof. The proof is by induction over the length of the longest reduction path out of a strongly normalising expression E, with a subinduction on the size of E.

If E is a $\beta\sigma\pi$-normal form, then E is typeable by Proposition 19.

If E is itself a redex, let E' be the expression obtained by contracting redex E. Therefore E' is strongly normalising and by IH it is typeable. Then E is typeable, by Proposition 23.

Next suppose that E is not itself a redex nor a normal form. Then E is of one of the following forms: $\lambda x.u$, $x(u :: k)$, $u :: k$, or $\widehat{x}.u$ (in each case with u or k *not* $\beta\pi\sigma$-normal). Each of the above u and k is typeable by IH, as the subexpressions of E. It is easy then to build the typing of E, as in the proof of Proposition 19. □

Corollary 25. *A term is strongly normalising if and only if it is typeable in* $\lambda^{\mathsf{Gtz}}\cap$.

Proof. By Theorems 18 and 24. □

5 Generalised applications and explicit substitutions

We consider two extensions of the λ-calculus: the ΛJ-calculus, where application $M(N, x.P)$ is *generalised* [10]; and the $\lambda\mathtt{x}$-calculus, where substitution $M\langle x := N\rangle$ is *explicit* [17]. Intersection types have been used to characterise the strongly normalising terms of both ΛJ-calculus [14] and $\lambda\mathtt{x}$-calculus [13].

Both in [14] and [13] the "natural" typing rules for generalised application or substitution had to be supplemented with extra rules (the rule app_2 in [14]; the rules *drop* or $K-Cut$ in [13]) in order to secure that every strongly normalising term is typeable. Indeed, examples of terms are given whose reduction in ΛJ or $\lambda\mathtt{x}$ always terminates, but which would not be typeable, had the extra rules not been added to the typing system. The examples in ΛJ [14] and $\lambda\mathtt{x}$ [13] are

$$t_0 := (\lambda x.x(x, w.w))(\lambda z.z(z, w.w), y.y'), \;\; y' \neq y \;\;,$$
$$t_1 := y'\langle y := xx\rangle\langle x := \lambda z.zz\rangle \;\;,$$

respectively. Two questions are raised by these facts: first, why the "natural" rules fail to capture the strongly normalising terms; second, how to characterise in terms of reduction the terms that receive a type under the "natural" typing rules. We now prove that λ^{Gtz} and $\lambda^{\mathsf{Gtz}}\cap$ are useful for giving an answer to these questions.

Definition 26. *Let t be a λ^{Gtz}-term.*

1. *t is a* λJ-term *if every cut occurring in t is of the form $t(u :: \widehat{x}.v)$.*
2. *t is a* $\lambda\mathtt{x}$-term *if every cut occurring in t has one of the forms $t(u :: \widehat{x}.x)$ or $t(\widehat{x}.v)$.*

We adopt the terminology "λJ-term" (instead of "ΛJ-term") for the sake of uniformity. We may write $t(u, x.v)$ instead of $t(u :: \widehat{x}.v)$. Let $t(u)$ abbreviate $t(u :: \widehat{x}.x)$ and $v\langle x := t\rangle$ denote $t(\widehat{x}.v)$. An inductive characterisation is:

$$(\lambda J\text{-terms}) \;\; t, u, v ::= x \mid \lambda x.t \mid t(u, x.v)$$
$$(\lambda\mathtt{x}\text{-terms}) \;\; t, u, v ::= x \mid \lambda x.t \mid t(u) \mid v\langle x := t\rangle$$

Definition 27.

1. *$\lambda J\cap$ is the typing system consisting of the rules Ax, $\to_R$ and the following rule, where $\cap A_k = A_1 \cap \cdots \cap A_n$ and $\cap B_i = B_1 \cap \cdots \cap B_m$, for some $n, m \geq 1$:*

$$\frac{\Gamma \vdash t : \cap A_k \to B_i \ , \forall i \in \{1, \cdots, m\} \quad \Gamma \vdash u : A_k \ , \forall k \in \{1, \cdots, n\} \quad \Gamma, x : \cap B_i \vdash v : C}{\Gamma \vdash t(u, x.v) : C} \ (Gen.Elim)$$

2. *$\lambda \mathtt{x} \cap$ is the typing system consisting of the rules Ax, $\to_R$ and the following rules, where $\cap A_k = A_1 \cap \cdots \cap A_n$, for some $n \geq 1$:*

$$\frac{\Gamma \vdash t : \cap A_k \to B \quad \Gamma \vdash u : A_k \ , \forall k \in \{1, \cdots, n\}}{\Gamma \vdash t(u) : B} \ (Elim)$$

$$\frac{\Gamma \vdash t : A_k, \ \forall k \in \{1, \cdots, n\} \quad \Gamma, x : \cap A_k}{\Gamma \vdash v\langle x := t\rangle : B} \ (Subst)$$

If $n = m = 1$ in $(Gen.Elim)$, then we obtain the usual rule for assigning simple types to generalised application. If $n = 1$ in $(Elim)$ or $(Subst)$, then we obtain the usual rule for assigning simple types to application or substitution.

$\lambda J\cap$ is a "natural" system for typing λJ-terms, in two senses. First, the rules in $\lambda J\cap$ follow the natural deduction format. Notice that we retained in $\lambda J\cap$ only the rules of $\lambda^{\mathsf{Gtz}}\cap$ that act on the RHS formula of sequents, and replaced the other rules of $\lambda^{\mathsf{Gtz}}\cap$ by an elimination rule. Second, $\lambda J\cap$ has just one rule for typing generalised applications, contrary to in [14]. Similarly, $\lambda\mathtt{x}\cap$ is a "natural" system for typing $\lambda\mathtt{x}$-terms. Again, we retained in $\lambda\mathtt{x}\cap$ only the rules of $\lambda^{\mathsf{Gtz}}\cap$ that act on the RHS formula of sequents, and replaced the other rules of $\lambda^{\mathsf{Gtz}}\cap$ by an elimination rule and a substitution rule. In addition, no extra cut or substitution rules are needed, contrary to [13].

The following is an addenda to GL.

Proposition 28. *In $\lambda^{\mathsf{Gtz}}\cap$ one has:*

1. *$\Gamma \vdash t(u, x.v) : C$ iff there are $A_1, \ldots, A_n$, $B_1, \ldots B_m$ such that $\Gamma \vdash t : \cap A_k \to B_i$, for all i; and $\Gamma \vdash u : A_k$, for all k; and $\Gamma, x : \cap B_i \vdash v : C$.*
2. *$\Gamma \vdash t(u) : B$ iff there are $A_1, \ldots, A_n$ such that $\Gamma \vdash t : \cap A_k \to B$ and $\Gamma \vdash u : A_k$, for all k.*
3. *$\Gamma \vdash v\langle x := t\rangle : B$ iff there are $A_1, \ldots, A_n$ such that $\Gamma \vdash t : A_i$, for all i; and $\Gamma, x : \cap A_i \vdash v : B$.*

Proof. We just sketch the proof of statement 1. The "only if" implication follows by successive application of GL. As to the "if" implication, let $A_1, \ldots, A_n$, $B_1, \ldots B_m$ be such that $\Gamma \vdash t : \cap A_k \to B_i$, $\forall i$, $\Gamma \vdash u : A_k$, $\forall k$, and $\Gamma, x : \cap B_i \vdash v : C$. Here we use $\cap A_k \to \cap B_i \sim \cap(\cap A_k \to B_i)$. Recall $t(u :: \widehat{x}.v)$ is denoted by $t(u, \widehat{x}.v)$.

$$\dfrac{\Gamma \vdash t : \cap A_k \to B_i, \ \forall i \qquad \dfrac{\Gamma \vdash u : A_k, \ \forall k \qquad \dfrac{\Gamma, x : \cap B_i \vdash v : C}{\Gamma; \cap B_i \vdash \widehat{x}.v : C} \ (Sel)}{\Gamma; \cap A_k \to \cap B_i \vdash u :: \widehat{x}.v} \ (\to L)}{\Gamma \vdash t(u :: \widehat{x}.v) : C} \ (Cut)$$

Proposition 29.

1. *Let t be a λJ-term. $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma \vdash t : A$ iff $\lambda J\cap$ derives $\Gamma \vdash t : A$.*
2. *Let t be a $\lambda\mathtt{x}$-term. $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma \vdash t : A$ iff $\lambda\mathtt{x}\cap$ derives $\Gamma \vdash t : A$.*

Proof. The "if" implications are proved by induction on $\Gamma \vdash t : A$ in $\lambda J\cap$ or $\lambda\mathtt{x}\cap$, using the fact that *Gen.Elim*, *Elim*, and *Subst* are derived rules of $\lambda^{\mathsf{Gtz}}\cap$ (which is clear from the proof of Proposition 28). The "only if" implications are proved by induction on t, and rely on GL and its addenda (Proposition 28). □

So we get a characterisation of typeability of t in the "natural" systems $\lambda J\cap$ or $\lambda\mathtt{x}\cap$ in terms of strong normalisability of t as a sequent term:

Corollary 30.

1. *Let t be a λJ-term. t is $\beta\pi\sigma\mu - SN$ iff t is typeable in $\lambda J\cap$.*
2. *Let t be a $\lambda\mathtt{x}$-term. t is $\beta\pi\sigma\mu - SN$ iff t is typeable in $\lambda\mathtt{x}\cap$.*

In addition, the "natural" systems $\lambda J\cap$ and $\lambda\mathtt{x}\cap$ *do capture* the strongly normalising terms, the point being what we mean by "strongly normalising". Going back to the examples t_0 and t_1 of the beginning of this section, although t_0 and t_1 are strongly normalising in ΛJ and $\lambda\mathtt{x}$, respectively, they are not so in λ^{Gtz}. Indeed, after one β-reduction step, t_0 becomes $(\lambda z.z(z, w.w))\widehat{x}.((x(x, w.w))\widehat{y}.y')$, which, by abbreviation, is $y'\langle y := x(x)\rangle\langle x := \lambda z.z(z)\rangle$, that is t_1! After one σ-reduction step, t_1 becomes the clearly non-terminating $y'\langle y := (\lambda z.z(z))(\lambda z.z(z))\rangle$. So, in this sense, it is correct that the natural typing systems $\lambda J\cap$ and $\lambda\mathtt{x}\cap$ (as well as the typing systems of [14] and [13] *without* extra-rules app_2, *drop*, and $K - Cut$) fail to give a type to t_0 and t_1, because these terms are, after all, non-terminating. Why were these terms no so in their native reduction systems? In ΛJ, t_0 becomes y' after one step of β-reduction because the two substitutions of t_1 cannot be formed and hence are immediately executed. In $\lambda\mathtt{x}$, the execution of the outer substitution in t_1 is blocked because $\lambda\mathtt{x}$ has no composition of substitutions.

6 Conclusion

This paper gives a characterisation, via intersection types, of the strongly normalising intuitionistic sequent terms. This expands the range of application of the intersection types technique. One of the points of extending the Curry-Howard correspondence to sequent calculus is that such exercise will shed light on issues like reduction, strong normalisability, or typeability in the original systems in natural deduction format. In this paper this promise is fulfilled, because the characterisation of strong normalisability in the sequent calculus proves useful for analysing recent applications of intersection types in natural deduction system containing generalised applications or explicit substitutions. This analysis confirms that there is a delicate equilibrium between clean typing systems and expressive reduction systems.

References

1. R. Amadio and P-L. Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.
2. H. Barendregt and S. Ghilezan. Lambda terms for natural deduction, sequent calculus and cut elimination. *J. Funct. Program.*, 10(1):121–134, 2000.
3. M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for lambda terms. *Archiv für Mathematische Logik*, 19:139–156, 1978.
4. D. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage. *to appear in Theoretical Computer Science*, 2007.
5. J. Espírito Santo. Revisiting the correspondence between cut-elimination and normalisation. In *Proceedings of ICALP'2000*, volume 1853 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.
6. J. Espírito Santo. Completing Herbelin's programme. In S. Ronchi Della Rocca, editor, *Proceedings of TLCA'07*, volume 4583 of *Lecture Notes in Computer Science*, pages 118–132. Springer-Verlag, 2007.
7. J. Espírito Santo. Delayed substitutions. In F. Baader, editor, *Proceedings of RTA'07*, volume 4533 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag, 2007.
8. J. Espírito Santo and L. Pinto. Permutative conversions in intuitionistic multiary sequent calculi with cuts. In *in Procedings of TLCA 2003*, volume 2071 of *Lecture Notes in Computer Science*, pages 286–300, 2003.
9. H. Herbelin. A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Computer Science Logic, CSL 1994*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1995.
10. F. Joachimski and R. Matthes. Standardization and confluence for ΛJ. In *Proceedings of RTA 2000*, volume 1833 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2000.
11. K. Kikuchi. Simple proofs of characterizing strong normalization for explicit substitution calculi. In F. Baader, editor, *Proceedings of RTA 2007*, volume 4533 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2007.
12. J.L. Krivine. *Lambda-calcul, types et modèles*. Masson, Paris, 1990.
13. S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Inf. Comput.*, 189(1):17–42, 2004.
14. R. Matthes. Characterizing strongly normalizing terms of a λ-calculus with generalized applications via intersection types. In J. Rolin et al., editor, *ICALP Workshops 2000*, pages 339–354. Carleton Scientific, 2000.
15. G. Pottinger. A type assignment for the strongly normalizable λ-terms. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–577. Academic Press, London, 1980.
16. S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theor. Comput. Sci.*, 59:181–209, 1988.
17. K. Rose. Explicit substitutions: Tutorial & survey. Technical Report LS-96-3, BRICS, 1996.
18. P. Sallé. Une extension de la théorie des types en lambda-calcul. In G. Ausiello and C. Böhm, editors, *Fifth International Conference on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 398–410. Springer-Verlag, 1978.
19. H. Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. *Theoretical Computer Science*, 212(1–2):247–260, 1999.