Lambda terms for natural deduction, sequent calculus and cut elimination

Henk Barendregt* and Silvia Ghilezan†

May 26, 1998

Abstract

It is well-known that there is a good correspondence between natural deduction derivations and typed lambda terms. Moreover normalizing these terms is equivalent to eliminating cuts in the corresponding sequent calculus derivations. Several papers have been written on this topic. The correspondence between sequent calculus derivations and natural deduction derivations is, however, not a one-to-one map. This causes some syntactic technicalities. The correspondence is best explained by two extensionally equivalent type assignment systems for untyped lambda terms, one corresponding to natural deduction (λN) and the other to sequent calculus (λL). These two systems constitute different grammars for generating the same (type assignment relation for untyped) lambda terms. The second grammar is ambiguous, but the first one is not. This fact explains the many-one correspondence mentioned above. Moreover, the second type assignment system has a 'cut–free' fragment (λL^{cf}). This fragment generates exactly the typeable lambda terms in normal form. The cut elimination theorem becomes a simple consequence of the fact that typed lambda terms posses a normal form.

Acknowledgment. The paper uses the TEX macros of Paul Taylor for building prooftrees. We thank Daniel Isaacson for useful comments.

1. Introduction

It is well-known that there is a good correspondence between natural deduction derivations and typed lambda terms. The relation between lambda terms and derivations in sequent calculus, between normal lambda terms and cut–free derivations in sequent calculus and finally between normalization of terms and cut elimination of derivations has been observed by several authors (Prawitz [1965], Zucker [1974] and Pottinger [1977]). This relation is less perfect because several cut–free sequent derivations correspond to one lambda term. In Herbelin [1995] a lambda calculus with explicit substitution operators is used in order to establish a perfect match between terms of that calculus and sequent

*Department of Computer Science, Catholic University, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands; e-mail: henk@cs.kun.nl.

†Faculty of Engineering, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Yugoslavia; e-mail: gsilvia@uns.ns.ac.yu.

derivations. We will not avoid the mismatch, but get a satisfactory view of it, by seeing the sequent calculus as a more intensional way to do the same as natural deduction: assigning lambda terms to provable formulas.

Next to the well-known system $\lambda\!\rightarrow$ of Curry type assignment to type free terms, which here will be denoted by λN, there are two other systems of type assignment: λL and its cut-free fragment $\lambda L^{\rm cf}$. The three systems λN, λL and $\lambda L^{\rm cf}$ correspond exactly to the natural deduction calculus NJ, the sequent calculus LJ and the cut–free fragment of LJ, here denoted by N, L and $L^{\rm cf}$ respectively. Moreover, λN and λL generate the same type assignment relation. The system $\lambda L^{\rm cf}$ generates the same type assignment relation as λN restricted to normal terms and cut elimination corresponds exactly to normalization. The mismatch between the logical systems that was observed above, is due to the fact that λN is a syntax directed system, whereas both λL and $\lambda L^{\rm cf}$ are not. (A syntax directed version of λL is possible if rules with arbitrarily many assumptions are allowed, see Capretta and Valentini [1998].)

The type assignment system of this paper is a subsystem of one in Barbanera et al. [1995] and also implicitly present in Mints [1996]. So our contribution is mainly expository.

For simplicity the results are presented only for the essential kernel of intuitionistic logic, i.e. for the minimal implicational fragment. The method probably can be extended to the full logical system, using the terms as in Mints [1996].

2. The logical systems N, L and $L^{\rm cf}$

2.1. Definition. The set $\mathtt{form}$ of formulas (of minimal implicational propositional logic) is defined by the following abstract syntax.

$$\begin{array}{lcl} \mathtt{form} & = & \mathtt{atom} \mid \mathtt{form}\!\rightarrow\!\mathtt{form} \\ \mathtt{atom} & = & \mathtt{p} \mid \mathtt{atom}' \end{array}$$

We write $p, q, r, \ldots$ for arbitrary atoms and $A, B, C, \ldots$ for arbitrary formulas. Sets of formulas are denoted by $\Gamma, \Delta, \ldots$. The set Γ, A stands for $\Gamma \cup \{A\}$.

2.2. Definition. (i) A statement A is *derivable* in the system N from the set Γ, notation $\Gamma \vdash_N A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules.

N

$$\frac{A\in\Gamma}{\Gamma \vdash A}\ \text{axiom} \qquad \frac{\Gamma \vdash A\!\rightarrow\!B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \rightarrow\text{elim} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A\!\rightarrow\!B}\ \rightarrow\text{intr}$$

(ii) A statement A is *derivable* from assumptions Γ in the system L, notation $\Gamma \vdash_L A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules.

L	
$\dfrac{A \in \Gamma}{\Gamma \vdash A}$	axiom
$\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}$	$\to$ left
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	$\to$ right
$\dfrac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}$	cut

(iii) The system L^{cf} is obtained from the system L by omitting the rule (cut).

L^{cf}	
$\dfrac{A \in \Gamma}{\Gamma \vdash A}$	axiom
$\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}$	$\to$ left
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	$\to$ right

2.3. Lemma. *Suppose* $\Gamma \subseteq \Gamma'$. *Then*

$$\Gamma \vdash A \;\Rightarrow\; \Gamma' \vdash A$$

in all systems.

Proof. By a trivial induction on derivations. ■

2.4. Proposition. *For all* Γ *and* A *we have*

$$\Gamma \vdash_N A \;\Leftrightarrow\; \Gamma \vdash_L A.$$

Proof. ($\Rightarrow$) By induction on derivations in N. For the rule ($\to$ elim) we need the rule (cut).

$$\dfrac{\Gamma \vdash_L A \to B \qquad \dfrac{\Gamma \vdash_L A \qquad \dfrac{}{\Gamma, B \vdash_L B}\,(\text{axiom})}{\Gamma, A \to B \vdash_L B}\,(\to \text{ left})}{\Gamma \vdash_L B}\,(\text{cut})$$

($\Leftarrow$) By induction on derivations in L. The rule ($\to$ left) is treated as follows.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_N A}{\Gamma, A{\to}B \vdash_N A}\,(2.3) \quad \dfrac{}{\Gamma, A{\to}B \vdash_N A{\to}B}\,(\text{axiom})}{\Gamma, A{\to}B \vdash_N B}\,(\to \text{ elim}) \quad \dfrac{\Gamma, B \vdash_N C}{\Gamma \vdash_N B{\to}C}\,(\to \text{ intr})}{\Gamma, A{\to}B \vdash_N C}\,(\to \text{ elim})$$

The rule (cut) is treated as follows.

$$\dfrac{\Gamma \vdash_N A \qquad \dfrac{\Gamma, A \vdash_N B}{\Gamma \vdash_N A{\to}B}\,(\to \text{ intr})}{\Gamma \vdash_N B}\,(\to \text{ elim}).\ \blacksquare$$

2.5. Definition. Consider the following rule as alternative to the rule (cut).

$$\dfrac{\Gamma, A{\to}A \vdash B}{\Gamma \vdash B}\,(\text{cut}')$$

The system L' is defined by replacing the rule (cut) by (cut$'$).

2.6. Proposition. *For all* Γ *and* A

$$\Gamma \vdash_L A \;\Leftrightarrow\; \Gamma \vdash_{L'} A.$$

Proof. ($\Rightarrow$) The rule (cut) is treated as follows.

$$\dfrac{\dfrac{\Gamma \vdash_{L'} A \qquad \Gamma, A \vdash_{L'} B}{\Gamma, A{\to}A \vdash_{L'} B}\,(\to \text{ left})}{\Gamma \vdash_{L'} B}\,(\text{cut}')$$

($\Leftarrow$) The rule (cut$'$) is treated as follows.

$$\dfrac{\Gamma, A{\to}A \vdash_L B \qquad \dfrac{\dfrac{}{\Gamma, A \vdash_L A}\,(\text{axiom})}{\Gamma \vdash_L A{\to}A}\,(\to \text{ right})}{\Gamma \vdash_L B}\,(\text{cut}).\ \blacksquare$$

Note that we have not yet investigated the role of L^{cf}.

3. The type assignment systems λN, λL and λL^{cf}

3.1. Definition. The set `term` of type–free lambda terms is defined as follows.

`term`	=	`var` $\mid$ `term term` $\mid$ λ`var.term`
`var`	=	`x` $\mid$ `var`$'$

We write $x, y, z, \ldots$ for arbitrary variables in terms and $P, Q, R, \ldots$ for arbitrary terms. Equality of terms (up to renaming of bound variables) is denoted by $\equiv$. The identity is $\mathsf{I} \equiv \lambda x.x$. A term P is called a β normal form (*P is in β-nf*) if P has no *redex* as part, i.e. no subterm of the form $(\lambda x.R)S$. Such a redex is said to reduce as follows

$$(\lambda x.R)S \to_\beta R[x{:=}S],$$

where $R[x{:=}S]$ denotes the result of substituting S for the free occurrences of x. The transitive reflexive closure of $\to_\beta$ is denoted by $\twoheadrightarrow_\beta$. If $P \twoheadrightarrow_\beta Q$ and Q is in β-nf, then Q is called the β-nf of P (one can show it is unique). A collection $\mathcal{A}$ of terms is said to be *strongly normalizing* if for no $P{\in}\mathcal{A}$ there is an infinite reduction path

$$P \to_\beta P_1 \to_\beta P_2 \ldots .$$

3.2. Definition. (i) A *type assignment* is an expression of the form

$$P : A,$$

where P is a lambda term and A is a formula.

(ii) A *declaration* is a type assignment of the form

$$x : A.$$

(iii) A *context* Γ is a set of declarations such that for every variable x there is at most one declaration $x{:}A$ in Γ.

3.3. Definition. (i) A type assignment $P : A$ is *derivable* from the context Γ in the system λN (also known as $\lambda{\to}$), notation

$$\Gamma \vdash_{\lambda N} P : A,$$

if $\Gamma \vdash P : A$ can be generated by the following axiom and rules.

λN

$$\frac{(x{:}A){\in}\Gamma}{\Gamma \vdash x : A} \quad \text{axiom}$$

$$\frac{\Gamma \vdash P : (A{\to}B) \qquad \Gamma \vdash Q : A}{\Gamma \vdash (PQ) : B} \quad {\to}\ \text{elim}$$

$$\frac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)} \quad {\to}\ \text{intr}$$

(ii) A type assignment $P : A$ is *derivable* form the context Γ in the system λL, notation

$$\Gamma \vdash_{\lambda L} P : A,$$

if $\Gamma \vdash P : A$ can be generated by the following axiom and rules.

λL	
$\dfrac{(x{:}A)\in\Gamma}{\Gamma \vdash x{:}A}$	axiom
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x{:}B \vdash P : C}{\Gamma, y : A{\to}B \vdash P[x{:=}yQ] : C}$	$\to$ left
$\dfrac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)}$	$\to$ right
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x{:}A \vdash P : B}{\Gamma \vdash P[x{:=}Q] : B}$	cut

In the rule ($\to$ left) it is required that $\Gamma, y{:}A{\to}B$ is a context. This is the case if y is fresh or if $\Gamma = \Gamma, y{:}A{\to}B$, i.e. $y{:}A{\to}B$ already occurs in Γ.

(iii) The system λL^{cf} is obtained from the system λL by omitting the rule (cut).

λL^{cf}	
$\dfrac{(x{:}A)\in\Gamma}{\Gamma \vdash x{:}A}$	axiom
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x{:}B \vdash P : C}{\Gamma, y : A{\to}B \vdash P[x{:=}yQ] : C}$	$\to$ left
$\dfrac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)}$	$\to$ right

3.4. Remark. The alternative rule (cut$'$) could also have been used to define the variant $\lambda L'$. The right version for the rule (cut$'$) with term assignment is as follows.

Rule cut$'$ for $\lambda L'$	
$\dfrac{\Gamma,\ x{:}A{\to}A \vdash P : B}{\Gamma \vdash P[x{:=}\mathsf{I}] : B}$	cut$'$

Notation. Let $\Gamma = \{A_1, \ldots, A_n\}$ and $\vec{x} = \{x_1, \ldots, x_n\}$. Write

$$\Gamma_{\vec{x}} = \{x_1{:}A_1, \ldots, x_n{:}A_n\}$$

and

$$\Lambda^\circ(\vec{x}) = \{P{\in}\mathtt{term} \mid FV(P) \subseteq \vec{x}\},$$

where $FV(P)$ is the set of free variables of P.

The following result has been observed for N and λN by Curry, Howard and de Bruijn. (See Troelstra and Schwichtenberg [1996] 2.1.5. and Hindley [1997] 6B3, for some fine points about the correspondence between deductions in N and corresponding terms in λN.)

3.5. Proposition (Propositions—as—types interpretation). *Let S be one of the logical systems N, L or L^{cf} and let λS be the corresponding type assignment system. Then*

$$\Gamma \vdash_S A \Leftrightarrow \exists \vec{x}\, \exists P{\in}\Lambda^\circ(\vec{x})\ \Gamma_{\vec{x}} \vdash_{\lambda S} P : A.$$

Proof. ($\Rightarrow$) By an easy induction on derivations, just observing that the right lambda term can be constructed. ($\Leftarrow$) By omitting the terms. ■

Since λN is exactly $\lambda{\to}$, the simply typed lambda calculus, we know the following results whose proofs are not hard, but are omitted here. From corollary 4.3 it follows that the results also hold for λL.

3.6. Proposition. (i) *(Normalization theorem for λN)*

$$\Gamma{\vdash_{\lambda N}} P : A \ \Rightarrow\ P \textit{ has a } \beta\textit{-nf } P^{nf}.$$

(ii) *(Subject reduction theorem for λN)*

$$\Gamma{\vdash_{\lambda N}} P : A \ \&\ P \twoheadrightarrow_\beta P' \ \Rightarrow\ \Gamma{\vdash_{\lambda N}} P' : A.$$

(iii) *(Generation lemma for λN) Type assignment for terms of a certain syntactic form is caused in the obvious way.*

$$\begin{array}{llcll}
(1) & \Gamma \vdash_{\lambda N} & x & : A & \Rightarrow \quad (x{:}A) \in \Gamma. \\
(2) & \Gamma \vdash_{\lambda N} & PQ & : B & \Rightarrow \quad \Gamma \vdash_{\lambda N} P : (A{\to}B)\ \&\ \Gamma \vdash_{\lambda N} Q : A, \\
 & & & & \qquad \textit{for some type } A. \\
(3) & \Gamma \vdash_{\lambda N} & \lambda x.P & : C & \Rightarrow \quad \Gamma, x{:}A \vdash_{\lambda N} P : B\ \&\ C \equiv A{\to}B, \\
 & & & & \qquad \textit{for some types } A, B.
\end{array}$$

Proof. See e.g. Gandy [1980] for (i) and Barendregt [1992] for (ii) and (iii). ■

Actually, even strong normalization holds for terms typeable in λN (see e.g. de Vrijer [1987] or Barendregt [1992]).

4. Relating λN, λL and λL^{cf}

Now the proof of the equivalence between systems N and L will be 'lifted' to that of λN and λL.

4.1. Proposition. $\Gamma{\vdash_{\lambda N}} P : A \ \Rightarrow\ \Gamma{\vdash_{\lambda L}} P : A$.

PROOF. By inductions on derivations in λN. Modus ponens ($\to$ elim) is treated as follows.

$$\dfrac{\Gamma \vdash_{\lambda L} P : A \to B \qquad \dfrac{\Gamma \vdash_{\lambda L} Q : A \quad \Gamma, x{:}B \vdash_{\lambda L} x{:}B}{\Gamma, y{:}A \to B \vdash_{\lambda L} yQ : B}\ (\to \text{ left})}{\Gamma \vdash_{\lambda L} PQ : B}\ (\text{cut}).\ \blacksquare$$

4.2. PROPOSITION. (i) $\Gamma \vdash_{\lambda L} P : A \ \Rightarrow\ \Gamma \vdash_{\lambda N} P' : A$, *for some* $P' \twoheadrightarrow_\beta P$.

(ii) $\Gamma \vdash_{\lambda L} P : A \ \Rightarrow\ \Gamma \vdash_{\lambda N} P : A$.

PROOF. (i) By induction on derivations in λL. The rule ($\to$ left) is treated as follows (the justifications are left out, but they are as in the proof of 2.4).

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_{\lambda N} Q : A}{\Gamma, y{:}A \to B \vdash_{\lambda N} Q : A} \qquad \dfrac{}{\Gamma, y{:}A \to B \vdash_{\lambda N} y{:}A \to B}}{\Gamma, y{:}A \to B \vdash_{\lambda N} yQ : B} \qquad \dfrac{\Gamma, x{:}B \vdash_{\lambda N} P : C}{\Gamma \vdash_{\lambda N} (\lambda x.P) : B \to C}}{\Gamma, y{:}A \to B \vdash_{\lambda N} (\lambda x.P)(yQ) : C}$$

Now $(\lambda x.P)(yQ) \to_\beta P[x{:=}yQ]$ as required. The rule (cut) is treated as follows.

$$\dfrac{\Gamma \vdash_{\lambda N} Q : A \quad \dfrac{\Gamma, x{:}A \vdash_{\lambda N} P : B}{\Gamma \vdash_{\lambda N} (\lambda x.P) : A \to B}\ (\to \text{ intr})}{\Gamma \vdash_{\lambda N} (\lambda x.P)Q : B}\ (\to \text{ elim})$$

Now $(\lambda x.P)Q \to_\beta P[x{:=}Q]$ as required.

(ii) By (i) and the subject reduction theorem for λN (3.6(ii)). $\blacksquare$

4.3. COROLLARY. $\Gamma \vdash_{\lambda L} P : A \ \Leftrightarrow\ \Gamma \vdash_{\lambda N} P : A$.

PROOF. By propositions 4.1 and 4.2(ii). $\blacksquare$

Now we will investigate the role of the cut–free system.

4.4. PROPOSITION.

$$\Gamma \vdash_{\lambda L^{\text{cf}}} P : A \ \Rightarrow\ P \textit{ is in } \beta\textit{-nf.}$$

PROOF. By an easy induction on derivations. $\blacksquare$

4.5. LEMMA. *Suppose*

$$\Gamma \vdash_{\lambda L^{\text{cf}}} P_1 : A_1,\ \ldots\ ,\ \Gamma \vdash_{\lambda L^{\text{cf}}} P_n : A_n.$$

Then

$$\Gamma, x{:}A_1 \to \ldots \to A_n \to B \vdash_{\lambda L^{\text{cf}}} xP_1 \ldots P_n : B$$

for those variables x *such that* $\Gamma, x{:}A_1 \to \ldots \to A_n \to B$ *is a context.*

PROOF. We treat the case $n = 2$, which is perfectly general. We abbreviate $\vdash_{\lambda L^{\mathrm{cf}}}$ as $\vdash$.

$$\dfrac{\Gamma \vdash P_1 : A_1 \qquad \dfrac{\Gamma \vdash P_2 : A_2 \qquad \dfrac{}{\Gamma, z{:}B \vdash z : B}\,(\text{axiom})}{\Gamma, y{:}A_2{\to}B \vdash yP_2 \equiv z[z{:=}yP_2] : B}\,(\to \text{ left})}{\Gamma, x{:}A_1{\to}A_2{\to}B \vdash xP_1P_2 \equiv (yP_2)[y{:=}xP_1] : B}\,(\to \text{ left})$$

Note that x may occur in some of the P_i. ■

4.6. PROPOSITION. *Suppose that P is a β-nf. Then*

$$\Gamma \vdash_{\lambda N} P : A \;\Rightarrow\; \Gamma \vdash_{\lambda L^{\mathrm{cf}}} P : A.$$

PROOF. By induction on the following generation of normal forms.

$$\mathrm{nf} = \mathtt{var} \mid \mathtt{var}\ \mathrm{nf}^{+} \mid \lambda\mathtt{var}.\mathrm{nf}$$

The cases $P \equiv x$ and $P \equiv \lambda x.P_1$ are easy. The case $P \equiv xP_1 \dots P_n$ follows from the previous lemma, using the generation lemma for λN (3.6(iii)). ■

Now we get as bonus the Hauptsatz of Gentzen [1936] for minimal implicational sequent calculus.

4.7. THEOREM (Cut elimination).

$$\Gamma \vdash_L A \;\Rightarrow\; \Gamma \vdash_{L^{\mathrm{cf}}} A.$$

PROOF.
$$\begin{array}{lll}
\Gamma \vdash_L A & \Rightarrow & \Gamma_{\vec{x}} \vdash_{\lambda L} P : A, \quad \text{for some } P{\in}\Lambda^{\circ}(\vec{x}), \text{ by 3.5,} \\
& \Rightarrow & \Gamma_{\vec{x}} \vdash_{\lambda N} P : A, \quad \text{by 4.2(ii),} \\
& \Rightarrow & \Gamma_{\vec{x}} \vdash_{\lambda N} P^{\mathrm{nf}} : A, \quad \text{by 3.6(i),(ii),} \\
& \Rightarrow & \Gamma_{\vec{x}} \vdash_{\lambda L^{\mathrm{cf}}} P^{\mathrm{nf}} : A, \quad \text{by 4.6,} \\
& \Rightarrow & \Gamma \vdash_{L^{\mathrm{cf}}} A, \quad \text{by 3.5.}
\end{array}$$
■

As it is clear that the proof implies that cut-elimination can be used to normalize terms typable in $\lambda N = \lambda{\to}$, Statman [1979] implies that the expense of cut-elimination is beyond elementary time (Grzegorczyk class 4). Moreover, as the cut-free deduction is of the same order of complexity as the corresponding normal lambda term, the size of the cut-free version of a derivation is non elementary in the size of the original derivation.

5. Discussion

The main technical tool is the type assignment system λL corresponding exactly to sequent calculus (for minimal propositional logic). The type assignment system λL is a subsystem of a system studied in Barbanera et al. [1995]. The terms involved in λL are also in Mints [1996]. The difference between the present approach and the one by Mints is that in that paper derivations in L

are first order citizens, whereas in λL the provable formulas and the lambda terms are.

In λN typeable terms are built up as usual (following the grammar of lambda terms). In λL^{cf} only normal terms are typeable. They are built up from variables by transitions like

$$P \;\longmapsto\; \lambda x.P$$

and

$$P \;\longmapsto\; P[x{:=}yQ]$$

This is an ambiguous way of building terms, in the sense that one term can be built up in several ways. For example, one can assign to the term $\lambda x.yz$ the type $C{\to}B$ (in the context $z{:}A, y{:}A{\to}B$) via two different cut–free derivations:

$$\dfrac{\dfrac{x{:}C, z{:}A \vdash z : A \qquad x{:}C, z{:}A, u{:}B \vdash u : B}{x{:}C, z{:}A, y{:}A{\to}B \vdash yz : B}\,(\to \text{ left})}{z{:}A, y{:}A{\to}B \vdash \lambda x.yz : C{\to}B}\,(\to \text{ right})$$

and

$$\dfrac{z{:}A \vdash z{:}A \qquad \dfrac{x{:}C, z{:}A, u{:}B \vdash u : B}{z{:}A, u{:}B \vdash \lambda x.u : C{\to}B}\,(\to \text{ right})}{z{:}A, y{:}A{\to}B \vdash \lambda x.yz : C{\to}B}\,(\to \text{ left})$$

These correspond, respectively, to the following two formations of terms

$$\begin{array}{llll} u & \longmapsto & yz & \longmapsto & \lambda x.yz, \\ u & \longmapsto & \lambda x.u & \longmapsto & \lambda x.yz. \end{array}$$

Therefore there are more sequent calculus derivations giving rise to the same lambda term. This is the cause of the mismatch between sequent calculus and natural deduction as described in Zucker [1974], Pottinger [1977] and Mints [1996]. See also Dyckhoff and Pinto [1997], Schwichtenberg [1997] and Troelstra [1998].

In Herbelin [1995] the mismatch between L-derivations and lambda terms is repaired by translating these into terms with explicit substitution:

$$\begin{array}{c} \lambda x.(u < u{:=}yz >), \\ (\lambda x.u) < u{:=}yz > . \end{array}$$

In our paper lambda terms are considered as first class citizens also for sequent calculus. This gives an insight into the mentioned mismatch by understanding it as an intensional aspect how the sequent calculus generates these terms.

It is interesting to note, how in the full system λL the rule (cut) generates terms not in β–normal form. The extra transition now is

$$P \;\longmapsto\; P[x{:=}F].$$

This will introduce a redex, if x occurs actively (in a context xQ) and F is an abstraction ($F \equiv \lambda x.R$), the other applications of the rule (cut) being superfluous. Also, the alternative rule (cut$'$) can be understood better. Using this rule the extra transition becomes

$$P \longmapsto P[x{:=}\mathsf{I}].$$

This will have the same effect (modulo one β–reduction) as the previous transition, if x occurs in a context xFQ. So with the original rule (cut) the argument Q (in the context xQ) is waiting for a function F to act on it. With the alternative rule (cut$'$) the function F comes close (in context xFQ), but the 'couple' FQ has to wait for the 'green light' provided by I.

Also, it can be observed that if one wants to manipulate derivations in order to obtain a cut–free proof, then the term involved gets reduced. By the strong normalization theorem for λN $(= \lambda{\to})$ it follows that eventually a cut–free proof will be reached.

We have not studied in detail whether cut elimination can be done along the lines of this paper for the full system of intuitionistic predicate logic, but there seems to be no problem. More interesting is the question, whether there are similar results for classical and linear logic.

References

Barbanera, F., M. Dezani-Ciancaglini and U. de' Liguoro [1995]
Intersection and union types: syntax and semantics, *Information and Computation* **119**, pp. 202–230.

Barendregt, H.P. [1992]
Lambda calculi with types, *in:* S. Abramsky, D. M. Gabbai and T. S. E. Maibaum (eds.), *Handbook of Logic in Computer Science*, Vol. 2, Oxford University Press, Oxford, pp. 117–309.

Capretta, V. and S. Valentini [1998]
A general method for proving the normalization theorem for first and second order typed λ-calculi, *Mathematical Structures in Computer Science*. To appear.

Dyckhoff, R. and L. Pinto [1997]
Permutability of proofs in intuitionistic sequent calculi, *Theoretical Computer Science*. To appear.

Gandy, R. [1980]
An early proof of normalization by A. M. Turing, *in:* J. P. Seldin and J. R. Hindley (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, pp. 453–457.

Gentzen, G. [1936]
Untersuchungen über das logischen Schliessen, *Mathematische Zeitschrift* **39**, pp. 405–431. Translation in: *Collected papers of Gerhard Gentzen*, ed. M. E. Szabo, North-Holland, Amsterdam [1969], pp. 68–131.

Herbelin, H. [1995]
A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure, *Computer Science Logic (CSL'94)*, Lecture Notes in Computer Science **933**, Springer Verlag, Berlin, pp. 61–75.

Hindley, J.R. [1997]
Basic Simple Type Theory, Cambridge University Press, Cambridge, UK.

Mints, G. [1996]
Normal forms for sequent derivations, *in:* P. Odifreddi (ed.), *Kreiseliana. About and Around Georg Kreisel*, A.K. Peters, Wellesley, Massachusetts, pp. 469–492.

Pottinger, G. [1977]
Normalization as a homomorphic image of cut–elimination, *Annals of Mathematical Logic* **12**, pp. 323–357.

Prawitz, D. [1965]
Natural deduction. A proof-theoretical study, Almquist and Wiksell, Stockholm.

Schwichtenberg, H. [1997]
Termination of permutative conversion inintuitionistic Gentzen calculi, *Theoretical Computer Science*. To appear.

Statman, R. [1979]
The typed λ-calculus is not elementary recursive, *Theoretical Computer Science* **9**, pp. 73–81.

Troelstra, A. S. [1998]
Marginalia on sequent calculi, *Studia Logica*. To appear.

Troelstra, A. S. and H. Schwichtenberg [1996]
Basic Proof Theory, Cambridge University Press, Cambridge, UK.

de Vrijer, R. [1987]
Exactly estimating functionals and strong normalization, *Indagationes Mathematicae* **49**, pp. 479–493.

Zucker, J. [1974]
Cut–elimination and normalization, *Annals of Mathematical Logic* **7**, pp. 1–112.