THEORETICAL PEARL
Lambda terms for natural deduction, sequent calculus and cut elimination

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

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

For Roger Hindley on his 60th birthday

Abstract

It is well known that there is an isomorphism between natural deduction derivations and typed lambda terms. Moreover, normalising these terms corresponds to eliminating cuts in the equivalent 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-one map, which 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 (\(\lambda N\)) and the other to sequent calculus (\(\lambda L\)). 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 (\(\lambda L^\text{cf}\)). 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.

1 Introduction

Systems of natural deduction for propositional and predicate logic have been introduced in Gentzen (1935) to represent in a natural way (intuitionistic) arguments. In that same paper Gentzen also introduced the sequent calculus systems, equivalent to the system of natural deduction, i.e. having the same set of derivable statements. By leaving out one of the derivation rules this system has a natural sub-system, the cut-free sequent calculus, having again the same set of derivable statements (this fact is stated by Gentzen’s Hauptsatz or cut-elimination theorem). This subsystem enjoys certain properties, by which several meta-properties about the full logical systems
can be proved. For example, in this way it can be shown that intuitionistic systems satisfy the disjunction property

\[ \vdash A \lor B \Rightarrow \vdash A \text{ or } \vdash B. \]

Another proof-theoretic property, valid for both classical and intuitionistic logic, that can be derived from Gentzen’s *Hauptsatz* is the Craig interpolation theorem

\[ \vdash A \rightarrow B \Rightarrow \vdash A \rightarrow C \text{ and } \vdash C \rightarrow B, \]

for some \( C \) with as non-logical symbols (propositional atoms, relation symbols or function symbols) a subset of those occurring both in \( A \) and in \( B \) (see Troelstra and Schwichtenberg (1996)).

Later, another method of obtaining such meta-statements was given by Prawitz (1965), by showing that in the systems of natural deduction each derivation can be reduced to a *normal form* derivation, which proves the same conclusion as the original one.

So Gentzen obtained the meta-theoretic results by relying on the subset of the derivations in sequent calculus that can be characterised easily by leaving out the cut-rule, whereas Prawitz on the other hand used the system of natural deduction and the structural requirement on derivations of being in normal form. The *Hauptsatz* of Gentzen is more easy to formulate than Prawitz’ normalisation theorem: no structural properties of derivations need to be stated. On the other hand, for the proof of the *Hauptsatz* as well as for the proof of its consequences, one does need to analyse structural properties of sequent calculus derivations.

It became clear that there was a relation between the two methods and more in particular between the cut-free derivations in the sequent calculus and the normal derivations in the system of natural deduction (Prawitz, 1965; Zucker, 1974; Pottinger, 1977). The correspondence is not one-one: several cut-free derivations correspond to one normal derivation. This may have caused some of the mentioned work to be quite lengthy.

There is a perfect correspondence between natural deduction derivations and typed lambda terms. This was hinted at (for the implicational fragment) by Curry (1942), first formulated well by Howard (1980) (written in 1969), and used intensively, but not explicitly formulated, by de Bruijn (in his Automath project for the automated verification of mathematics – see Nederpelt *et al.*, 1994). The correspondence is often called the Curry Howard(-de Bruijn) isomorphism. As the map also involves a correspondence between statements (formulas) and types, it is also referred to as the formulas-as-types interpretation.

Herbelin (1995) relates sequent calculus derivations and terms of the typed lambda calculus extended with explicit substitution operators (e.g. see Bloo and Rose (1996) for an introduction to this subject). This clarifies the situation, as typed \( \lambda \)-terms with explicit substitution operators are halfway between sequent calculus derivations and typed lambda calculus, see the discussion in Section 5.

We prefer to describe the situation in a different way from the point of view of ordinary lambda terms (without explicit substitution operators). A satisfactory
view can be obtained, by considering the sequent calculus as a more intensional way to do the same as natural deduction: assigning lambda terms to provable formulas. There is a well-known system that assigns types to untyped lambda terms, the Curry assignment system $\lambda \to$, which here will be denoted by $\lambda N$. Next to this system there are two other systems of type assignment: $\lambda L$ and its cut-free fragment $\lambda L_{cf}$. These systems have been described by Gallier (1993), in Barbanera et al. (1995) (appearing as subsystems of one that also includes intersection types), and by Mints (1996). The three systems $\lambda N$, $\lambda L$ and $\lambda L_{cf}$ exactly correspond respectively to the natural deduction calculus $NJ$, the sequent calculus $LJ$ and the cut-free fragment of $LJ$, here denoted by $N$, $L$ and $L_{cf}$. Moreover, $\lambda N$ and $\lambda L$ generate the same type assignment relation. The system $\lambda L_{cf}$ generates the same type assignment relation as $\lambda N$ restricted to normal terms and cut elimination exactly corresponds to normalisation.

In $\lambda N$ given a $\Gamma$ and an $M$, one can find relatively easily a proposition $A$ such that $\Gamma \vdash M : A$. Conversely, in $\lambda L_{cf}$ given the $\Gamma$ and the $A$ one can find relatively easily an $M$ such that $\Gamma \vdash M : A$. (Still, the complexity of this is PSPACE complete as shown by Statman, 1979.)

Using this approach, the Hauptsatz can be seen as a (canonical) consequence of the normalisation theorem for typeable lambda terms. Since all of this must have been known to several people, our contribution is mainly expository: but we have not seen the story told in this way. The emphasis is on lambda terms rather than on derivations. As a matter of fact, lambda terms are more easy to reason about than the two dimensional derivations. Since derivations in the natural deduction system are isomorphic to typed lambda terms, there is some preference for this logical system, notably for the intuitionistic case.

For simplicity the results are presented only for the essential kernel of intuitionistic logic, i.e. for the minimal implicational fragment. The proof of the Hauptsatz probably can be extended along the same lines to the full logical system, using the terms as in Gallier (1993).

2 The logical systems $N$, $L$ and $L_{cf}$

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

$$
\begin{align*}
\text{form} & = \text{atom} \mid \text{form} \to \text{form} \\
\text{atom} & = p \mid \text{atom}'
\end{align*}
$$

We write $p$, $q$, $r$,… for arbitrary atoms and $A$, $B$, $C$,… for arbitrary formulas. Sets of formulas are denoted by $\Gamma$, $\Delta$,… The set $\Gamma$, $A$ stands for $\Gamma \cup \{A\}$. We consider derivability from finite sets of formulas (for the system $L$ this is a bit unorthodox, but inessential).

Definition 2.2
(i) A statement $A$ is derivable in the system $N$ from the set $\Gamma$, notation $\Gamma \vdash_N A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules:
(ii) A statement $A$ is derivable from assumptions $\Gamma$ in the system $L$, notation $\Gamma \vdash_{L} A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules:

$$
\begin{array}{c}
A \in \Gamma \\
\Gamma \vdash A
\end{array}
\begin{array}{c}
\text{axiom}
\end{array}
$$

$$
\begin{array}{c}
\Gamma \vdash A \rightarrow B \\
\Gamma \vdash A
\end{array}
\begin{array}{c}
\Gamma \vdash B
\end{array}
\begin{array}{c}
\rightarrow \text{elim}
\end{array}
$$

$$
\begin{array}{c}
\Gamma, A \vdash B
\end{array}
\begin{array}{c}
\Gamma \vdash A \rightarrow B
\end{array}
\begin{array}{c}
\rightarrow \text{intr}
\end{array}
$$

(iii) The system $L_{\text{cf}}$ is obtained from the system $L$ by omitting the rule (cut).

$$
\begin{array}{c}
A \in \Gamma \\
\Gamma \vdash A
\end{array}
\begin{array}{c}
\text{axiom}
\end{array}
$$

$$
\begin{array}{c}
\Gamma \vdash A \\
\Gamma, B \vdash C
\end{array}
\begin{array}{c}
\Gamma, A \rightarrow B \vdash C
\end{array}
\begin{array}{c}
\rightarrow \text{left}
\end{array}
$$

$$
\begin{array}{c}
\Gamma, A \vdash B
\end{array}
\begin{array}{c}
\Gamma \vdash A \rightarrow B
\end{array}
\begin{array}{c}
\rightarrow \text{right}
\end{array}
$$

$$
\begin{array}{c}
\Gamma \vdash A \\
\Gamma, A \vdash B
\end{array}
\begin{array}{c}
\Gamma \vdash B
\end{array}
\begin{array}{c}
\text{cut}
\end{array}
$$

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

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

in all systems.

Proof
By a trivial induction on derivations.

Proposition 2.4
For all $\Gamma$ and $A$ we have

$$
\Gamma \vdash_{N} A \iff \Gamma \vdash_{L} A.
$$
Proof

(⇒) By induction on derivations in \( N \). For the rule (\( \rightarrow \) elim) we need the rule (cut).

\[
\frac{\Gamma \vdash_L A \quad \Gamma, B \vdash_L B}{\Gamma, A \rightarrow B \vdash_L B} \quad \text{(\( \rightarrow \) left)}
\]

\[
\frac{\Gamma \vdash_L B}{\Gamma \vdash_L L} \quad \text{\textit{Proof}}
\]

(⇐) By induction on derivations in \( L \). The rule (\( \rightarrow \) left) is treated as follows:

\[
\frac{\Gamma, B \vdash_N C \quad \Gamma \vdash_N B \rightarrow C}{\Gamma, A \rightarrow B \vdash_N C} \quad \text{\textit{Lemma 2.3}}
\]

\[
\frac{\Gamma, A \vdash_B A \rightarrow B \quad \Gamma \vdash_B B \rightarrow B}{\Gamma, A \vdash_B B} \quad \text{\textit{Assumption}}
\]

The rule (cut) is treated as follows:

\[
\frac{\Gamma, A \vdash_B B \quad \Gamma \vdash_B B \rightarrow B}{\Gamma \vdash_B B} \quad \text{\textit{Proof}}.
\]

\[
\Gamma \vdash_L L \quad \text{\textit{Proof}}.
\]

Definition 2.5

Consider the following rule as alternative to the rule (cut):

\[
\frac{\Gamma, A \vdash_B B \quad \Gamma \vdash_B B \rightarrow B}{\Gamma \vdash_B B} \quad \text{\textit{Proposition 2.6}}
\]

For all \( \Gamma \) and \( A \)

\[
\Gamma \vdash_L A \iff \Gamma \vdash_{L'} A.
\]

Proof

(⇒) The rule (cut) is treated as follows:

\[
\frac{\Gamma \vdash_{L'} A \quad \Gamma, A \vdash_{L'} B}{\Gamma, A \rightarrow B \vdash_{L'} B} \quad \text{\textit{Proof}}.
\]

(⇐) The rule (cut') is treated as follows:

\[
\frac{\Gamma, A \vdash_{L} A \quad \Gamma, A \vdash_{L} A \rightarrow B}{\Gamma \vdash_{L} A \rightarrow B} \quad \text{\textit{Proof}}.
\]

Note that we have not yet investigated the role of \( L^\xi \).
3 The type assignment systems $\lambda N$, $\lambda L$ and $\lambda L^{st}$

Definition 3.1

The set $\text{term}$ of type-free lambda terms is defined as follows:

$$\text{term} = \text{var} | \text{term} \text{ term} | \lambda \text{var}. \text{term}$$

$$\text{var} = x | \text{var}'$$

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 $1 \equiv \lambda x.x$. A term $P$ is called a $\beta$ normal form ($P$ is in $\beta$-nf) if $P$ has no redex as part, i.e. no subterm of the form $(\lambda x.R)S$. A term with redex is said to reduce as follows:

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

Here $C[\ ]$ is the rest of the term (the context) and $R[x := S]$ denotes the result of substituting $S$ for the free occurrences of $x$. The transitive reflexive closure of $\rightarrow_{\beta}$ is denoted by $\rightarrow_{\beta}^{\beta}$. If $P \rightarrow_{\beta}^{\beta} Q$ and $Q$ is in $\beta$-nf, then $Q$ is called the $\beta$-nf of $P$ (one can show it is unique). A collection $A$ of terms is said to be strongly normalising if for no $P \in A$ there is an infinite reduction path

$$P \rightarrow_{\beta} P_1 \rightarrow_{\beta} P_2 \ldots .$$

Definition 3.2

(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 $\Gamma$ is a set of declarations such that for every variable $x$ there is at most one declaration $x : A$ in $\Gamma$.

Definition 3.3

(i) A type assignment $P : A$ is derivable from the context $\Gamma$ in the system $\lambda N$ (also known as $\lambda \rightarrow$), notation $\Gamma \vdash_{\lambda N} P : A$,

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

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash x : A$</td>
<td>$\Gamma \vdash (x : A) \in \Gamma$</td>
</tr>
<tr>
<td>$\Gamma \vdash (PQ) : B$</td>
<td>$\Gamma, x : A \vdash P : B$</td>
</tr>
<tr>
<td>$\Gamma \vdash \lambda x.P : (A \rightarrow B)$</td>
<td>$\Gamma \vdash P : (A \rightarrow B)$</td>
</tr>
</tbody>
</table>
(ii) A type assignment $P : A$ is derivable from the context $\Gamma$ in the system $\lambda L$, notation

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

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

<table>
<thead>
<tr>
<th>$\lambda L$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x : A) \in \Gamma$</td>
<td>$\frac{\text{axiom}}{\Gamma \vdash x : A}$</td>
</tr>
<tr>
<td>$\Gamma \vdash Q : A$</td>
<td>$\frac{\text{→ left}}{\Gamma, x : B \vdash P : C}$</td>
</tr>
<tr>
<td>$\Gamma, y : A \rightarrow B \vdash P[,x := yQ,] : C$</td>
<td>$\frac{\text{→ right}}{\Gamma \vdash (\lambda x.P) : (A \rightarrow B)}$</td>
</tr>
<tr>
<td>$\Gamma \vdash Q : A$</td>
<td>$\frac{\text{cut}}{\Gamma, x : A \vdash P : B}$</td>
</tr>
<tr>
<td>$\Gamma \vdash P[,x := Q,] : B$</td>
<td></td>
</tr>
</tbody>
</table>

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

(iii) The system $\lambda L^\ast$ is obtained from the system $\lambda L$ by omitting the rule (cut).

<table>
<thead>
<tr>
<th>$\lambda L^\ast$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x : A) \in \Gamma$</td>
<td>$\frac{\text{axiom}}{\Gamma \vdash x : A}$</td>
</tr>
<tr>
<td>$\Gamma \vdash Q : A$</td>
<td>$\frac{\text{→ left}}{\Gamma, x : B \vdash P : C}$</td>
</tr>
<tr>
<td>$\Gamma, y : A \rightarrow B \vdash P[,x := yQ,] : C$</td>
<td>$\frac{\text{→ right}}{\Gamma \vdash (\lambda x.P) : (A \rightarrow B)}$</td>
</tr>
<tr>
<td>$\Gamma \vdash Q : A$</td>
<td>$\frac{\text{cut}}{\Gamma, x : A \vdash P : B}$</td>
</tr>
<tr>
<td>$\Gamma \vdash P[,x := Q,] : B$</td>
<td></td>
</tr>
</tbody>
</table>

**Remark 3.4**

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:

<table>
<thead>
<tr>
<th>Rule cut' for $\lambda L'$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\text{cut'}}}{\Gamma, x : A \rightarrow A \vdash P : B}$</td>
<td></td>
</tr>
</tbody>
</table>

**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

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

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

The following result has been observed for $N$ and $\lambda N$ 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 $\lambda N$.)

**Proposition 3.5** [Propositions-as-type interpretation].

Let $S$ be one of the logical systems $N$, $L$ or $L^\text{cf}$ and let $\lambda S$ be the corresponding type assignment system. Then

$$\Gamma \vdash_S A \iff \exists \exists P \in \Lambda^\text{c}(\exists x) \; \Gamma \vdash_{\lambda S} P : A.$$

**Proof**

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

Since $\lambda N$ 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 $\lambda L$.

**Proposition 3.6**

(i) (Normalisation theorem for $\lambda N$)

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

(ii) (Subject reduction theorem for $\lambda N$)

$$\Gamma \vdash_{\lambda N} P : A \text{ and } P \to_{\beta} P' \Rightarrow \Gamma \vdash_{\lambda N} P' : A.$$

(iii) (Generation lemma for $\lambda N$) Type assignment for terms of a certain syntactic form is caused in the obvious way.

1. $\Gamma \vdash_{\lambda N} x : A \Rightarrow (x : A) \in \Gamma.$
2. $\Gamma \vdash_{\lambda N} PQ : B \Rightarrow \Gamma \vdash_{\lambda N} P : (A \to B) \text{ and } \Gamma \vdash_{\lambda N} Q : A,$
   for some type $A$.
3. $\Gamma \vdash_{\lambda N} \lambda x.P : C \Rightarrow \Gamma, x : A \vdash_{\lambda N} P : B \text{ and } C \equiv A \to B,$
   for some types $A, B$.

**Proof**

(i) See, for example, Turing’s proof in Gandy (1980). The idea is that reduction of the rightmost redex of highest rank decreases the number of such redexes, where rank is defined by

$$\begin{align*}
\text{rank}(p) &= 0; \\
\text{rank}(A \to B) &= \max\{\text{rank}(A) + 1, \text{rank}(B)\}.
\end{align*}$$

(ii) and (iii) See Barendregt (1992).

Actually, even strong normalisation holds for terms typeable in $\lambda N$ (e.g. see de Vrijer (1987) or Barendregt (1992)).
4 Relating $\lambda N$, $\lambda L$ and $\lambda L^f$

Now the proof of the equivalence between systems $N$ and $L$ will be 'lifted' to that of $\lambda N$ and $\lambda L$.

**Proposition 4.1**

$\Gamma \vDash_{\lambda N} P : A \Rightarrow \Gamma \vDash_{\lambda L} P : A$.

**Proof**

By induction on derivations in $\lambda N$. Modus ponens ($\to$ elim) is treated as follows (use a fresh $y$).

\[
\begin{align*}
\Gamma \vdash_{\lambda L} Q : A & \quad \Gamma, x : B \vdash_{\lambda L} x : B \\
\Gamma \vdash_{\lambda L} P : A \to B & \quad \Gamma, y : A \to B \vdash_{\lambda L} yQ : B \quad \text{(cut)}
\end{align*}
\]

**Proposition 4.2**

(i) $\Gamma \vDash_{\lambda L} P : A \Rightarrow \Gamma \vDash_{\lambda N} P' : A$, for some $P' \to_{\beta} P$.

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

**Proof**

(i) By induction on derivations in $\lambda L$. The rule ($\to$ left) is treated as follows (the justifications are left out, but they are as in the proof of 2).

\[
\begin{align*}
\Gamma, x : B \vdash_{\lambda N} P & : C \\
\Gamma \vdash_{\lambda N} Q & : A \\
\Gamma, y : A \to B \vdash_{\lambda N} Q : A & \\
\Gamma, y : A \to B \vdash_{\lambda N} y : A & \to B
\end{align*}
\]

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

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

\[
\begin{align*}
\Gamma, x : A \vdash_{\lambda N} P & : B \\
\Gamma \vdash_{\lambda N} (\lambda x.P) : A \to B & \text{($\to$ intr)} \\
\Gamma \vdash_{\lambda N} Q & : A \text{($\to$ elim).}
\end{align*}
\]

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

(ii) By (i) and the subject reduction theorem for $\lambda N$ (3(ii)).

**Corollary 4.3**

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

**Proof**

By propositions 4 and 4(ii).

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

**Proposition 4.4**

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

**Proof**

By an easy induction on derivations.
Lemma 4.5
Suppose
\[ \Gamma \vdash_{IL} P_1 : A_1, \ldots, \Gamma \vdash_{IL} P_n : A_n. \]
Then
\[ \Gamma, x : A_1 \to \cdots \to A_n \to B \vdash_{IL} x P_1 \ldots P_n : B \]
for those variables \( x \) such that \( \Gamma, x : A_1 \to \cdots \to A_n \to B \) is a context.

Proof
We treat the case \( n = 2 \), which is perfectly general. For \( \vdash_{IL} \) we write \( \vdash \) and \( y \) will be a fresh variable.

\[ \begin{array}{c}
\Gamma \vdash P_2 : A_2 \\
\Gamma \vdash P_1 : A_1 \\
\Gamma, y : A_2 \to B \vdash y P_2 \equiv z[z := y P_2] : B \\
\Gamma, x : A_1 \to A_2 \to B \vdash x P_1 P_2 \equiv (y P_2)[y := x P_1] : B
\end{array} \]

Note that \( x \) may occur in some of the \( P_i \).

Proposition 4.6
Suppose that \( P \) is a \( \beta \)-nf. Then
\[ \Gamma \vdash_{IL} P : A \Rightarrow \Gamma \vdash_{IL} P : A. \]

Proof
By induction on the following generation of normal forms:
\[ \text{nf} = \text{var} | \text{var} \text{nf} | \lambda \text{var} \text{nf}. \]
The cases \( P \equiv x \) and \( P \equiv \lambda x. P_1 \) are easy. The case \( P \equiv x P_1 \ldots P_n \) follows from the previous lemma, using the generation lemma for \( \lambda N \) (3(iii)).

Now we get as a bonus the Hauptsatz of Gentzen (1935) for minimal implicational sequent calculus.

Theorem 4.7 [Cut elimination]
\[ \Gamma \vdash_L A \Rightarrow \Gamma \vdash_{Ld} A. \]

Proof
\[ \begin{array}{c}
\Gamma \vdash_L A \Rightarrow \Gamma \vdash_L P : A, \quad \text{for some } P \in \Lambda^x(\bar{x}), \text{ by 3}, \\
\Rightarrow \Gamma \vdash_{IL} P : A, \quad \text{by 4(ii)}, \\
\Rightarrow \Gamma \vdash_{IL} P_{\text{nf}} : A, \quad \text{by 3(i), (ii)}, \\
\Rightarrow \Gamma \vdash_{IL} P_{\text{nf}} : A, \quad \text{by 4}, \\
\Rightarrow \Gamma \vdash_{Ld} A, \quad \text{by 3}. \quad \square
\end{array} \]
The proof can be depicted as follows:

\[
\lambda N \iff \lambda L \\
N \iff L \iff \lambda N^\text{cf} = \Rightarrow \lambda L^\text{cf} \\
N^\text{cf} \iff L^\text{cf}
\]

As it is clear that the proof implies that successive elimination of cuts leads to a method normalising terms typeable in $\lambda N = \lambda \rightarrow$, the main result in Statman (1979) implies that the expense of such a procedure is beyond elementary time (Grzegorczyk class 4). Moreover, as a cut-free derivation is of the same order of complexity as the corresponding normal lambda term, the size of a derivation after such a procedure may not be elementary in the size of the original derivation. On the other hand, abandoning the desideratum of convertibility of the corresponding lambda terms, one may eliminate cuts by a procedure (see Hudelmaier, 1992) with an elementary upper bound on derivation growth.

5 Discussion

The main technical tool is the type assignment system $\lambda L$ corresponding exactly to sequent calculus (for minimal propositional logic). The type assignment system $\lambda L$ has been introduced by Gallier (1993) in Barbanera et al. (1995) (for a different purpose), and by Mints (1996). The difference between the present approach and those by Gallier and Mints is that in these papers, derivations in $L$ are first class citizens, whereas in $\lambda L$ the provable formulas and the lambda terms are.

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

\[
P \mapsto \lambda x. P
\]

and

\[
P \mapsto P[x := y Q].
\]

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 $C \rightarrow B$ (in the context $z : A$, $y : A \rightarrow B$) via two different cut-free derivations:

\[
\begin{align*}
x : C, z : A &\vdash z : A \\
x : C, z : A, u : B &\vdash u : B \\
x : C, z : A, y : A &\vdash B \vdash yz : B \\
\end{align*}
\]

\[
(\rightarrow \text{left})
\]

\[
\begin{align*}
x : C, z : A, u : B &\vdash u : B \\
z : A &\vdash z : A \\
\end{align*}
\]

\[
(\rightarrow \text{right})
\]

and

\[
\begin{align*}
x : C, z : A, u : B &\vdash u : B \\
z : A &\vdash z : A \\
\end{align*}
\]

\[
(\rightarrow \text{right})
\]

\[
\begin{align*}
z : A &\vdash A \vdash \lambda x. u : C \rightarrow B \\
\end{align*}
\]

\[
(\rightarrow \text{left})
\]

\[
\begin{align*}
z : A, y : A &\vdash B \vdash \lambda x. yz : C \rightarrow B \\
\end{align*}
\]
These correspond, respectively, to the following two formations of terms

\[ u \mapsto \lambda x. yz, \]
\[ u \mapsto \lambda x. u \mapsto \lambda x. yz. \]

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 by Zucker (1974), Pottinger (1977) and Mints (1996). See also Dyckhoff and Pinto (1999), Schwichtenberg (1999) and Troelstra (1999).

Herbelin (1995) pointed out that the L-derivations can be made into a one-to-one correspondence with typed lambda terms with explicit substitution. Define the system \( \lambda L^+ \), which is \( \lambda L \) with explicit substitutions as follows. \( \lambda L^{+\text{cf}} \) is the system without the cut-rule.

\[
\lambda L^+
\]

\[
\begin{array}{c}
(x : A) \in \Gamma \\
\Gamma \vdash x : A \\
\Gamma \vdash Q : A \\
\Gamma, y : A \rightarrow B \vdash P(x := yQ) : C \\
\Gamma, x : A \vdash P : B \\
\Gamma \vdash (\lambda x.P) : (A \rightarrow B) \\
\Gamma \vdash P(x := Q) : B
\end{array}
\]

Here \( P(x := Q) \) is an explicit substitution operator. Deductions in \( L \) and \( L^{\text{cf}} \) are in a one-to-one correspondence with lambda terms with explicit substitutions. The transition from \( L \) to \( \lambda L \) factorises through \( \lambda L^+ \) and the many-to-one aspect is caused by performing the substitutions.

So the above derivations correspond to

\[ \lambda x. (u(u := yz)), \]
\[ (\lambda x.u)(u := yz). \]

In the present paper, lambda terms (without explicit substitutions) are considered as first class citizens also for sequent calculus. This gives an insight into the mismatch mentioned: one of the causes is the intensional aspect how the sequent calculus generates lambda terms.
It is interesting to note, how in the full system $\lambda L$ the rule (cut) generates terms not in $\beta$-normal form. The extra transition now is

$$P \rightarrow P[x := F].$$

This will introduce a redex, if $x$ occurs actively (in a context $xQ$) and $F$ is an abstraction ($F = \lambda x.R$), 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

$$P \rightarrow P[x := I].$$

This will have the same effect (modulo one $\beta$-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 $I$. To obtain a cut-free proof one can manipulate derivations in such a way that the terms involved get reduced. By the strong normalisation theorem for $\lambda N$ ($=\lambda \rightarrow$) 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. See Urban and Bierman (1999) for work in the direction of classical logic.

Acknowledgements

The paper uses the \TeX{} macros of Paul Taylor for building proof-trees and the package XY-pic of Kris Rose and Ross Moore to construct diagrams. We thank Venanzio Capretta, Daniel Isaacson, Simon Peyton Jones, Christian Urban and notably Roy Dyckhoff for useful discussions.

References


