THEORETICAL PEARLS

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 sixtieth 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 (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 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.

1. Introduction

Systems of *natural deduction* for propositional and predicate logic have been introduced in Gentzen [1935] in order 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

$$\vdash A \vee B \;\Rightarrow \vdash A \text{ or } \vdash B.$$

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

$$\begin{array}{lll} \vdash A{\rightarrow}B & \Rightarrow & \vdash A{\rightarrow}C \text{ and } \vdash C{\rightarrow}B, \text{ for some } C \text{ with as non-logical symbols} \\ & & \text{(propositional atoms, relation symbols or function symbols)} \\ & & \text{a subset of those occurring both in } A \text{ and in } B, \end{array}$$

see Troelstra and Schwichtenberg [1996].

Later another method of obtaining such meta-statements has been given in 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, see Prawitz [1965], Zucker [1974] and 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 in 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 involves also 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 (see e.g. Bloo and Rose [1996] for an introduction to this subject). This clarifies the situation, as typed λ-terms with explicit substitution operators are halfway between sequent calculus derivations and typed lambda calculus, see the discussion in §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 $\lambda\!\rightarrow$, which here will be denoted by λN. Next to this system there are two other systems of type assignment: λL and its cut-free fragment λL^{cf}. These systems have been described in Gallier [1993], in Barbanera et al. [1995] (appearing as subsystems of one that also includes intersection types) and in Mints [1996]. The three systems λN, λL and λL^{cf} correspond exactly to respectively 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, λN and λL generate the same type assignment relation. The system λL^{cf} generates the same type assignment relation as λN restricted to normal terms and cut elimination corresponds exactly to normalisation.

In λN given a Γ and an M one can find relatively easily a proposition A such that $\Gamma \vdash M : A$. Conversely, in λL^{cf} given the Γ 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}

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

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

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 $\Gamma \cup \{A\}$. We consider derivability from finite sets of formulas (for the system L this is a bit unorthodox, but inessential).

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	
$\dfrac{A \in \Gamma}{\Gamma \vdash A}$	axiom
$\dfrac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$	$\to$ elim
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	$\to$ 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.

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

The rule (cut) is treated as follows.

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

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

$$\boxed{\dfrac{\Gamma, A{\to}A \vdash B}{\Gamma \vdash B} \qquad \text{cut}'}$$

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.

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

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` | `term term` | λ`var.term`
`var`	=	`x` | `var`$'$

We write $x, y, z, \dots$ for arbitrary variables in terms and $P, Q, R, \dots$ 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 $(\lambda x.R)S$. A term with redex is said to reduce as follows

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

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 $\to_\beta$ 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 normalising* if for no $P{\in}\mathcal{A}$ there is an infinite reduction path

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

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	
$\dfrac{(x{:}A)\in\Gamma}{\Gamma \vdash x : A}$	axiom
$\dfrac{\Gamma \vdash P : (A{\to}B) \qquad \Gamma \vdash Q : A}{\Gamma \vdash (PQ) : B}$	$\to$ elim
$\dfrac{\Gamma, x{:}A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)}$	$\to$ 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 $\Gamma = \{A_1,\dots,A_n\}$ and $\vec{x} = \{x_1,\dots,x_n\}$. Write

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

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) *(Normalisation 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)*

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

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

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

Proof. (i) See e.g. 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

$$\begin{array}{rcl} \mathtt{rank}(p) & = & 0; \\ \mathtt{rank}(A \to B) & = & \mathtt{max}\{\mathtt{rank}(A) + 1, \mathtt{rank}(B)\}. \end{array}$$

(ii) and (iii) See Barendregt [1992]. ■

Actually, even strong normalisation 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 induction on derivations in λN. Modus ponens ($\to$ elim) is treated as follows (use a fresh y).

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

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).

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

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

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

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)). ■

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). ■

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

4.4. Proposition.

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

Proof. By an easy induction on derivations. ■

4.5. Lemma. *Suppose*

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

Then

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

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. For $\vdash_{\lambda L^{\mathrm{cf}}}$ we write $\vdash$ and y will be a fresh variable.

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

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 $P \equiv xP_1 \ldots P_n$ follows from the previous lemma, using the generation lemma for λN (3.6(iii)). ■

Now we get as bonus the *Hauptsatz* of Gentzen [1935] for minimal implicational sequent calculus.

4.7. Theorem (Cut elimination).

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

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

The proof can be depicted as follows.

$$\begin{tikzcd}
& \lambda N \arrow[dd, Rightarrow] & & \lambda L \arrow[ll, Rightarrow] \arrow[dd, dash] \\
N \arrow[ur, dash] \arrow[rr, dash] \arrow[dd, dash] & & L \arrow[ur, Rightarrow] \arrow[dd, dash] & \\
& \lambda N^{\mathrm{nf}} \arrow[rr, Rightarrow] & & \lambda L^{\mathrm{cf}} \arrow[dl, Rightarrow] \\
N^{\mathrm{cf}} \arrow[ur, dash] \arrow[rr, dash] & & L^{\mathrm{cf}} &
\end{tikzcd}$$

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 λL corresponding exactly to sequent calculus (for minimal propositional logic). The type assignment system λL has been introduced in Gallier [1993], in Barbanera et al. [1995] (for a different purpose) and in Mints [1996]. The difference between the present approach and the ones by Gallier and Mints is that in these papers derivations in L are first class 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 $C{\rightarrow}B$ (in the context $z{:}A, y{:}A{\rightarrow}B$) 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

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

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 [1999], Schwichtenberg [1999] and Troelstra [1999].

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

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

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

$$\begin{array}{ccccc} L & = & \lambda L^{+} & — & \lambda L \\ | & & | & & | \\ L^{\mathtt{nf}} & = & \lambda L^{+\mathrm{cf}} & — & \lambda L^{\mathrm{cf}} \end{array}$$

So the above derivations correspond to

$$\lambda x.(u\langle u{:=}yz\rangle),$$
$$(\lambda x.u)\langle u{:=}yz\rangle.$$

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 mentioned mismatch: 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 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. In order 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 λN ($= \lambda\!\rightarrow$) 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.

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.

Bloo, R. and K. H. Rose [1996]. Combinatory reduction systems with explicit substitution that preserve strong normalization, *in:* H. Ganzinger (ed.), *Rewrite Techniques and Applications*, Lecture Notes in Computer Science 1103, Springer, Berlin and New York, pp. 169–183.

Curry, H.B. [1942]. The combinatory foundations of mathematical logic, *Journal of Symbolic Logic* **7**, pp. 47–64.

Dyckhoff, R. and L. Pinto [1999]. Permutability of proofs in intuitionistic sequent calculi, *Theoretical Computer Science* **212**, pp. 141–155.

Gallier, J. [1993]. Constructive logics (Part I: A tutorial on proof systems and typed lambda-calculi), *Theoretical Computer Science* **110**(2), pp. 249–339.

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. [1935]. Untersuchungen über das logischen Schliessen, *Mathematische Zeitschrift* **39**, pp. 176–210, 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.

Howard, W.A. [1980]. The formulas-as-types notion of construction, *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. 479–490.

Hudelmaier, J. [1992]. Bounds for cut elimination in intuitionistic propositional logic, *Archive for Mathematical Logic* **31**, pp. 331–353.

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.

Nederpelt, R.P., J.H. Geuvers and R.C. de Vrijer (eds.) [1994]. *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics **133**, North-Holland, Amsterdam.

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. [1999]. Termination of permutative conversion inintuitionistic Gentzen calculi, *Theoretical Computer Science* **212**, pp. 247–260.

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

Troelstra, A. S. [1999]. Marginalia on sequent calculi, *Studia Logica* **62**, pp. 291–303.

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

Urban, C. and G. Bierman [1999]. Strong normalisation of cut-elimination in classical logic, *in:* J.-Y. Girard (ed.), *Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science **1581**, Springer Verlag, Berlin, pp. 365–380.

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.