Revisiting the Correspondence between Cut Elimination and Normalisation

José Espírito Santo*

Laboratory for Foundations of Computer Science
Division of Informatics, The University of Edinburgh, James Clerk Maxwell Building,
The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland UK
jes@dcs.ed.ac.uk

Abstract. Cut-free proofs in Herbelin's sequent calculus are in 1-1 correspondence with normal natural deduction proofs. For this reason Herbelin's sequent calculus has been considered a privileged middle-point between L-systems and natural deduction. However, this bijection does not extend to proofs containing cuts and Herbelin observed that his cut-elimination procedure is not isomorphic to β-reduction.
In this paper we equip Herbelin's system with rewrite rules which, at the same time: (1) complete in a sense the cut elimination procedure firstly proposed by Herbelin; and (2) perform the intuitionistic "fragment" of the tq-protocol - a cut-elimination procedure for classical logic defined by Danos, Joinet and Schellinx. Moreover we identify the subcalculus of our system which is isomorphic to natural deduction, the isomorphism being with respect not only to proofs but also to normalisation.
Our results show, for the implicational fragment of intuitionistic logic, how to embed natural deduction in the much wider world of sequent calculus and what a particular cut-elimination procedure normalisation is.

1 Introduction

In his paper about a "λ-calculus structure" isomorphic to a "Gentzen-style sequent calculus structure"[6], Herbelin proposed to define a λ-like calculus corresponding to a LJ-like sequent calculus in the same way as λ-calculus corresponds to natural deduction.

Herbelin starts by refining the simplest, many-one assignment of terms to sequent calculus proofs, usually denoted by φ and that comes from the theory of the relationship between sequent calculus and natural deduction [10, 15, 9, 11, 3]. The refinement is to consider a restriction of LJ called LJT (respec. a term calculus called $\overline{\lambda}$-calculus) whose cut-free proofs (respec. cut-free terms) are in 1-1 correspondence with normal natural deduction proofs (respec. normal λ-terms).

* The author is supported by Fundação para a Ciência e Tecnologia, Portugal.

Dyckhoff and Pinto [2, 3] showed the merits of the *cut-free* fragment of LJT as a proof-theoretical tool and emphasized its privileged intermediate position between sequent calculus and natural deduction. The purpose of this paper is to define an intermediate system of this kind for proofs possibly containing cuts and a cut-elimination procedure which together give an isomorphic copy of natural deduction in sequent calculus format, with respect not only to proofs *but also to proof normalisation.*

Full LJT is not the solution to this problem. The bijection with natural deduction does not extend to proofs with cuts and Herbelin observed that his cut-elimination procedure fails to implement full β-reduction (it just implements a strategy).

The $\overline{\lambda}$-calculus includes an operator of explicit substitution so that local steps of cut permutation can be written as elementary steps of substitution propagation (calculi of explicit substitution for similar purposes can be found in [14, 5, 13]). Instead of making substitution explicit, we perform the complete upwards permutation of a cut in a single step of reduction by a global operation. This is inspired in the so-called tq-protocol, a cut-elimination procedure for classical "coloured" proofs defined by Danos, Joinet and Schellinx [1].

We equip LJT with a reduction procedure of this kind which completes, in a sense, LJT's original procedure, obtaining a sequent calculus and corresponding λ-calculus which we call HJ^+ and λ_H^+, respectively. We prove that HJ^+ is just performing the intuitionistic "fragment" of the tq-protocol and that the typable subcalculus of λ_H^+ is strongly normalising and confluent. Furthermore, we identify natural subsystems HJ and λ_H such that HJ (respec. λ_H) is isomorphic, in the strong sense required above, to NJ (respec. λ). In particular, both λ_H^+ and λ_H implement full β-reduction.

The reader finds in Table 1 (where *inc* stands for inclusion) a map of systems and translations which will appear in the following.

Notations and Terminology

We just treat intuitionistic implicational logic (implication written as $\supset$). Barendregt's convention applies to all calculi in this paper. A *context* is a *consistent* set of *declarations* $x : A$. By consistent we mean that if $x : A$ and $x : B$ are in a context, then $A = B$. Contexts are ranged over by Γ. We write $x \in \Gamma$ meaning $x : A \in \Gamma$ for some A. $\Gamma, x : A$ denotes the *consistent union* $\Gamma \cup \{x : A\}$, which means that, if x is already declared in Γ, then it is declared with type A.

We call *left* (respec. *right*) subderivation of a cut instance the subderivation in which the cutformula occurs in the RHS (respec. LHS) of its endsequent. Such cutformula is called the *left* (respec. *right*) cutformula of that instance.

2 Background

2.1 Herbelin's LJT and $\overline{\lambda}$-calculus

We refer to [6] for details about LJT and $\overline{\lambda}$. We adopt some simplification of syntax introduced in [2] (but the numbering of cuts is different!). Table 2 presents

Table 1. Map of systems and translations

$$
\begin{tikzcd}
 & LJT/\overline{\lambda} \arrow[d, "inc"] & & \\
 & LJT^{+}/\overline{\lambda}^{+} \arrow[d, dotted, no head] & & \\
LJ^{t} \arrow[r, shift right, "\overline{\overline{\varphi}}"'] & HJ^{+}/\lambda_{H}^{+} \arrow[l, shift right, "\overline{\overline{\rho}}"'] \arrow[r, shift right, "(\_)^{-}"'] & HJ/\lambda_{H} \arrow[l, shift right, "inc"'] \arrow[r, shift right, "\Theta"'] & NJ/\lambda \arrow[l, shift right, "\Psi"']
\end{tikzcd}
$$

the syntax and typing rules of $\overline{\lambda}$ (and thus the inference rules of LJT) and the reduction rules of $\overline{\lambda}$ which define the cut-elimination procedure of LJT (*Der* stands for Dereliction).

In $\overline{\lambda}$ there are two kinds of expressions: terms and lists of terms. The term $x[t_1, ..., t_n]$ can be seen as the λ-term $(...(xt_1)...t_n)$ but with the advantage of having the head-variable at the surface. $t\{x := v\}$ and $l\{x := v\}$ are explicit substitution operators and ll' is an explicit append of list (cf. the reduction rules). Notice that in $t\{x := v\}$ and $l\{x := v\}$, x is bound and $x \notin FV(v)$.

There are two kinds of derivable sequents: $\Gamma; - \vdash t : B$ and $\Gamma; A \vdash l : B$. In both there is a distinguished position in the LHS called *stoup*. The crucial restriction of LJT is that the rule $L\supset$ introduces $A \supset B$ in the stoup and B has to be in the stoup of the right subderivation's endsequent. Forget for a second rules cut_2 and cut_4. In this case (in particular in cut-free LJT), besides Ax, no rule can introduce a formula in the stoup and thus the last rule of the right subderivation of an instance of $L\supset$ is again $L\supset$ and so on until Ax is reached.

There are two kinds of cuts (head-cut and mid-cut) according to whether the right cutformula is in the stoup or not. Notice that in the reduction rules there are no permutation of cuts.

2.2 LJ^t and the Intuitionistic "tq-protocol"

Table 3 presents the sequent calculus LJ^t and a corresponding, nameless term calculus in which a cut-elimination procedure is expressed.

We leave to the reader to provide the definitions of free and bound variable in a term L. The idea is that, in $\mathsf{L}(x, L_1, (y)L_2)$, x occurs free and y bound. By Barendregt's convention, neither y occurs free in L_1 nor x occurs bound in L_1 or

Table 2. Herbelin's LJT and $\overline{\lambda}$-calculus

$$u, v, t ::= xl \,|\, \lambda x.t \,|\, tl \,|\, t\{x := v\}$$
$$l, l' ::= [] \,|\, t :: l \,|\, ll' \,|\, l\{x := v\}$$

$$Ax \frac{}{\Gamma; A \vdash [] : A} \quad Der \frac{\Gamma; A \vdash l : B}{\Gamma, x : A; - \vdash xl : B}$$

$$L \supset \ \frac{\Gamma; - \vdash t : A \quad \Gamma; B \vdash l : C}{\Gamma; A \supset B \vdash t :: l : C} \quad R \supset \ \frac{\Gamma, x : A; - \vdash t : B}{\Gamma; - \vdash \lambda x.t : A \supset B} x \notin \Gamma$$

mid-cuts

$$cut_1 \frac{\Gamma; - \vdash v : A \quad \Gamma, x : A; - \vdash t : B}{\Gamma; - \vdash t\{x := v\} : B} x \notin \Gamma \quad cut_2 \frac{\Gamma; - \vdash v : A \quad \Gamma, x : A; C \vdash l : B}{\Gamma; C \vdash l\{x := v\} : B} x \notin \Gamma$$

head-cuts

$$cut_3 \frac{\Gamma; - \vdash t : A \quad \Gamma; A \vdash l : B}{\Gamma; - \vdash tl : B} \quad cut_4 \frac{\Gamma; C \vdash l : A \quad \Gamma; A \vdash l' : B}{\Gamma; C \vdash ll' : B}$$

$(\overline{\lambda}11)$	$(\lambda x.t)(u :: l) \to t\{x := u\}l$
$(\overline{\lambda}12)$	$t[] \to t$
$(\overline{\lambda}21)$	$(xl)l' \to x(ll'),\ l' \neq []$
$(\overline{\lambda}31)$	$(u :: l)l' \to u :: (ll')$
$(\overline{\lambda}32)$	$[]l \to l$
$(\overline{\lambda}41)$	$(xl)\{x := v\} \to vl\{x := v\}$
$(\overline{\lambda}42)$	$(yl)\{x := v\} \to yl\{x := v\}, y \neq x$
$(\overline{\lambda}43)$	$(\lambda y.u)\{x := v\} \to \lambda y.u\{x := v\}$
$(\overline{\lambda}51)$	$(u :: l)\{x := v\} \to u\{x := v\} :: l\{x := v\}$
$(\overline{\lambda}52)$	$[]\{x := v\} \to []$

Table 3. LJ^t

$$L ::= \mathsf{Ax}(x) \mid \mathsf{Cut}(L, (y)L) \mid \mathsf{L}(x, L, (y)L) \mid \mathsf{R}((x)L)$$

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

$$L\supset \; \frac{\Gamma \vdash L_1 : A \quad \Gamma, y : B \vdash L_2 : C}{\Gamma, x : A \supset B \vdash \mathsf{L}(x, L_1, (y)L_2) : C}, y \notin \Gamma$$

$$R\supset \; \frac{\Gamma, x : A \vdash L : B}{\Gamma \vdash \mathsf{R}((x)L) : A \supset B} x \notin \Gamma$$

$$Cut \frac{\Gamma \vdash L_1 : A \quad \Gamma, y : A \vdash L_2 : C}{\Gamma \vdash \mathsf{Cut}(L_1, (y)L_2) : C} y \notin \Gamma$$

Structural step S_1:

$\mathsf{Cut}(L_1, (x)L_2) \rightarrow L_2[L_1/x]$, if x is not freshly and logically introduced in L_2

Structural step S_2:

$\mathsf{Cut}(L_1, (x)L_2) \rightarrow L_1[(x)L_2/-]$,
if x is freshly and logically introduced in L_2 and $L_1 \neq \mathsf{R}((z)L_0)$ all z, L_0

Logical step:

$\mathsf{Cut}(\mathsf{R}((z)L_0), (x)\mathsf{L}(x, L_1, (y)L_2))$
$\rightarrow \mathsf{Cut}(\mathsf{Cut}(L_1, (z)L_0), (y)L_2)$,
if x is freshly introduced in $\mathsf{L}(x, L_1, (y)L_2))$

where

$$\begin{aligned}
\mathsf{Ax}(x)[L/x] &= L \\
\mathsf{Ax}(y)[L/x] &= \mathsf{Ax}(y), y \neq x \\
\mathsf{L}(x, L', (z)L'')[L/x] &= \mathsf{Cut}(L, (x)\mathsf{L}(x, L'[L/x], (z)L''[L/x]) \\
\mathsf{L}(y, L', (z)L'')[L/x] &= \mathsf{L}(y, L'[L/x], (z)L''[L/x]), y \neq x \\
\mathsf{R}((y)L')[L/x] &= \mathsf{R}((y)L'[L/x]) \\
\mathsf{Cut}(L', (y)L'')[L/x] &= \mathsf{Cut}(L'[L/x], (y)L''[L/x])
\end{aligned}$$

$$\begin{aligned}
\mathsf{Ax}(y)[(x)L/-] &= L[y/x] \\
\mathsf{L}(y, L', (z)L'')[(x)L/-] &= \mathsf{L}(y, L', (z)L''[(x)L/-]) \\
\mathsf{R}((y)L')[(x)L/-] &= \mathsf{Cut}(\mathsf{R}((y)L'), (x)L) \\
\mathsf{Cut}(L', (y)L'')[(x)L/-] &= \mathsf{Cut}(L', (y)L''[(x)L/-])
\end{aligned}$$

L_2, although x may occur free in L_1 or L_2 (meaning that an implicit contraction is happening).

The cut-elimination procedure is a "fragment" of the so-called tq-protocol, a strongly normalising and confluent procedure for classical, "coloured" proofs defined in [1]. To be precise, it is the restriction of the tq-protocol to intuitionistic, t-coloured proofs in which an "orientation" of the multiplicative connective $\supset$ has been fixed.

Roughly, the protocol works as follows: a cut is firstly permuted upwards through its right subderivation (structural step S_1) and then through its left subderivation (structural step S_2) until it becomes a *logical* cut, to which the logical step applies, giving rise to new cuts of lower degree. A logical cut is a cut whose both cutformulas are *freshly and logically introduced*, *i.e.* introduced by a logical rule ($L\supset$ or $R\supset$) without implicit contraction. An equivalent description of step S_1 (respec. step S_2) is: to push the left (respec. right) subderivation upwards through the "tree of ancestors" [1] of the right (respec. left) cutformula.

The operations $L_2[L_1/x]$ and $L_1[(x)L_2/-]$ implement the structural steps S_1 and S_2, respectively, and are inspired in the operations of substitution and co-substitution defined by Urban and Bierman in [12].

3 HJ^+ and the λ_H^+-calculus

We refer to Table 4 for the definition of HJ^+ and λ_H^+. The motivation for these systems rests in the following observations.

The "life cycle" of a cut in LJT has three stages. It starts as a mid-cut and the first stage is a upwards permutation through its right subderivation, performed by rules $\overline{\lambda}4i$ and $\overline{\lambda}5j$. The goal is to generate head-cuts (see rule $\overline{\lambda}41$). The operation *subst* performs this permutation in a single step. In doing so, cuts of the form $l\{x:=v\}$ become "internal" to this process and hence are not needed in the syntax. Now observe that in LJT such permutation of a mid-cut can complete only if, in its course, we do not need to permute this mid-cut with another cut. This is why, in the definition of *subst*, extra clauses occur corresponding to the permutations

$$
\begin{array}{ll}
(\overline{\lambda}44) & (tl)\{x:=v\} \to t\{x:=v\}l\{x:=v\} \ , \\
(\overline{\lambda}45) & t\{y:=u\}\{x:=v\} \to t\{x:=v\}\{y:=u\{x:=v\}\} \ .
\end{array}
$$

Let us return to the head-cuts generated by the first stage. Notice that in a head-cut vl, if $l \neq []$ then its right cutformula is freshly and logically introduced. Such a cut is permuted upwards through its left subderivation by the rules $\overline{\lambda}21$ and $\overline{\lambda}3i$, generating along the way $\overline{\lambda}11$-redexes, *i.e.* logical cuts in the LJ^t sense. The last stage of the "life cycle" of these logical cuts is $\overline{\lambda}11$-reduction, by which cuts of lower degree are generated.

Again the operation *insert* performs in a single step the permutations of the second stage and cuts ll' become "internal" and thus superfluous in the syntax. Extra clauses in *insert*'s definition correspond to the permutations

Table 4. HJ^+ and λ_H^+-calculus

$$
\begin{aligned}
u, v, t &::= xl \,|\, \lambda x.t \,|\, tl \,|\, t\{x := v\} \\
l, l' &::= [] \,|\, t :: l
\end{aligned}
$$

$$
Ax \frac{}{\Gamma; A \vdash [] : A} \qquad Der \frac{\Gamma; A \vdash l : B}{\Gamma, x : A; - \vdash xl : B}
$$

$$
L\supset \ \frac{\Gamma; - \vdash t : A \quad \Gamma; B \vdash l : C}{\Gamma; A \supset B \vdash t :: l : C} \qquad R\supset \ \frac{\Gamma, x : A; - \vdash t : B}{\Gamma; - \vdash \lambda x.t : A \supset B} x \notin \Gamma
$$

$$
mid - cut \, \frac{\Gamma; - \vdash v : A \quad \Gamma, x : A; - \vdash t : B}{\Gamma; - \vdash t\{x := v\} : B} x \notin \Gamma
$$

$$
head - cut \, \frac{\Gamma; - \vdash t : A \quad \Gamma; A \vdash l : B}{\Gamma; - \vdash tl : B}
$$

$$
\begin{aligned}
(mid) &\quad t\{x := v\} \to subst(v, x, t) \\
(head) &\quad tl \to insert(l, t), \text{ if } t \text{ is not a } \lambda\text{-abstraction or } l = [] \\
(log) &\quad (\lambda x.t)(u :: l) \to t\{x := u\}l
\end{aligned}
$$

where

$$
\begin{aligned}
subst(v, x, x[]) &= v \\
subst(v, x, xl) &= v\, subst(v, x, l), l \neq [] \\
subst(v, x, yl) &= y\, subst(v, x, l), y \neq x \\
subst(v, x, \lambda y.t) &= \lambda y.subst(v, x, t) \\
subst(v, x, tl) &= subst(v, x, t) subst(v, x, l) \\
subst(v, x, t\{y := u\}) &= subst(v, x, t)\{y := subst(v, x, u)\}
\end{aligned}
$$

$$
\begin{aligned}
subst(v, x, u :: l) &= subst(v, x, u) :: subst(v, x, l) \\
subst(v, x, []) &= []
\end{aligned}
$$

$$
\begin{aligned}
insert([], t) &= t \\
insert(l, xl') &= x\, append(l', l),\ l \neq [] \\
insert(l, \lambda x.t) &= (\lambda x.t)l,\ l \neq [] \\
insert(l, tl') &= t\, append(l', l),\ l \neq [] \\
insert(l, t\{x := v\}) &= insert(l, t)\{x := v\},\ l \neq []
\end{aligned}
$$

$$
\begin{aligned}
append(t :: l, l') &= t :: append(l, l') \\
append([], l') &= l'
\end{aligned}
$$

$$
\begin{array}{ll}
(\overline{\lambda}22) & \qquad (tl')l \to t(l'l) \;, \\
(\overline{\lambda}23) & (t\{x := v\})l \to (tl)\{x := v\} \;.
\end{array}
$$

Define LJT^{+} as LJT plus the four new reduction rules just presented. We leave to the reader the formalisation of the obvious relations between LJT, LJT^{+} and HJ^{+}.

On the other hand, it should be clear that reductions $\to_{m(id)}$ and $\to_{h(ead)}$ in HJ^{+} have a strong connection with the structural steps S_1 and S_2, respectively, of LJ^t and that, roughly (but not exactly), a mid-cut is a S_1-redex and a head-cut is a S_2-redex. This is formalised by defining a map $\overline{\overline{\rho}} : HJ^{+} \to LJ^t$ as in Table 5 (where $z \notin FV(l)$ in the second and last clauses of the definition of $\overline{\overline{\rho}}$).

Table 5. Translations $\overline{\overline{\rho}}$ and $\overline{\overline{\varphi}}$

$\overline{\overline{\rho}}(x[]) = \mathsf{Ax}(x)$	$\overline{\overline{\varphi}}(\mathsf{Ax}(x)) = x[]$
$\overline{\overline{\rho}}(x(t :: l)) = \mathsf{L}(x, \overline{\overline{\rho}}(t), (z)\overline{\overline{\rho}}(zl))$	$\overline{\overline{\varphi}}(\mathsf{R}((x)L)) = \lambda x.\overline{\overline{\varphi}}(L)$
$\overline{\overline{\rho}}(\lambda x.t) = \mathsf{R}((x)\overline{\overline{\rho}}(t))$	$\overline{\overline{\varphi}}(\mathsf{L}(x, L_1, (y)L_2)) = \overline{\overline{\varphi}}(L_2)\{y := x[\overline{\overline{\varphi}}(L_1)]\}$
$\overline{\overline{\rho}}(t\{x := v\}) = \mathsf{Cut}(\overline{\overline{\rho}}(v), (x)\overline{\overline{\rho}}(t))$	$\overline{\overline{\varphi}}(\mathsf{Cut}(L_1, (y)L_2)) = \overline{\overline{\varphi}}(L_2)\{y := \overline{\overline{\varphi}}(L_1)\}$
$\overline{\overline{\rho}}(tl) = \mathsf{Cut}(\overline{\overline{\rho}}(t), (z)\overline{\overline{\rho}}(zl))$	

Lemma 1. *$\overline{\overline{\rho}}(subst(v, x, t)) = \overline{\overline{\rho}}(t)[\overline{\overline{\rho}}(v)/x]$, if $x \notin FV(v)$.*

Proposition 1. *If $t \to_m t'$ in HJ^{+} then either $\overline{\overline{\rho}}(t) \to_{S_1} \overline{\overline{\rho}}(t')$ or $\overline{\overline{\rho}}(t) = \overline{\overline{\rho}}(t')$ in LJ^t.*

The single anomaly is a *mid*-step of the form

$$(xl)\{x := v\} \to_m vl \;, \tag{1}$$

where $x \notin FV(l)$, which collapses in LJ^t because $\overline{\overline{\rho}}((xl)\{x := v\}) = \overline{\overline{\rho}}(vl)$. There is another (and last) anomaly, this time regarding head-cuts. The term $t[]$ is a S_1-redex and not a S_2-redex. We can split $\to_h$ as

$$
\begin{array}{l}
(h_1)\; t[] \to t \\
(h_2)\;\; tl \to insert(l, t), \text{ if } t \text{ is not a } \lambda\text{-abstraction and } l \neq []
\end{array}
$$

and then:

Lemma 2. *$\overline{\overline{\rho}}(insert(l, t)) = \overline{\overline{\rho}}(t)[(z)\overline{\overline{\rho}}(zl)/-]$, if $z \notin FV(l)$ and $l \neq []$.*

Proposition 2. *If $t \to_{h_i} t'$ in HJ^{+} then $\overline{\overline{\rho}}(t) \to_{S_i} \overline{\overline{\rho}}(t')$ in LJ^t, $i = 1, 2$.*

Finally:

Proposition 3. *If $t \rightarrow_{log} t'$ in HJ^+ then $\overline{\overline{\rho}}(t) \rightarrow_{Log} \overline{\overline{\rho}}(t')$ in LJ^t.*

Corollary 1. *The typable terms of λ_H^+ are strongly normalising.*

Proof. By strong normalisation of the tq-protocol, Propositions 1,2,3 and the fact that there can be no infinite reduction sequence starting from a λ_H^+-term and only made of steps (1). $\square$

In the following it will be useful, namely for proving confluence of λ_H^+, to consider a translation $\overline{\overline{\varphi}} : LJ^t \rightarrow HJ^+$, as defined in Table 5.

4 HJ and the λ_H-calculus

HJ (see Table 6) was obtained by simplifying HJ^+ in such a way that the new calculus could be proved isomorphic to NJ by means of functions Ψ, Θ extending those defined in [2] between cut-free LJT and normal NJ.

The first thing to do is to get rid of mid-cuts and $\rightarrow_m$. This requires that log becomes

$$(\lambda x.t)(u :: l) \rightarrow subst(u, x, t)l \ . \tag{2}$$

However this is not enough. One problem is that we would have $\Theta(t[]) = \Theta(t)$ and thus Θ would not be injective. Hence we must require $l \neq []$ in every head-cut tl. The second problem is that $\Psi(M)$ will be h-normal, for all λ-term M. This requires two measures: (a) We restrict ourselves to h-normal terms. When mid-cuts are dropped, $t(u :: l)$ is h-normal iff t is a λ-abstraction. Thus head-cuts are required to be of the restricted form $(\lambda x.t)(u :: l)$. (b) We drop $\rightarrow_h$ and have to reduce immediately, by performing $insert$, the h-redexes generated in (2). Now $subst(u, x, t)l$ can itself be a h-redex and the h-redex ul' may be created at subformulas of t of the form xl'. This explains the first clause in the new definition of $subst$ in HJ and the new version of (2) which we call β_H.

Every λ_H-term is a λ_H^+-term and next proposition says that β_H is a packet of several steps of reduction in HJ^+ and, indirectly, in the tq-protocol.

Proposition 4. *If $t \rightarrow_{\beta_H} t'$ in HJ then $t \rightarrow^+_{l,m,h} t'$ in HJ^+.*

Conversely, there is a translation $(\_)^- : HJ^+ \rightarrow HJ$ defined by:

$$\begin{aligned}
(xl)^- &= xl^- \ , \\
(\lambda x.t)^- &= \lambda x.t^- \ , \\
(tl)^- &= insert(l^-, t^-) \ , \\
(t\{x := v\})^- &= subst(v^-, x, t^-) \ , \\
\\
(u :: l)^- &= u^- :: l^- \ , \\
([])^- &= [] \ .
\end{aligned}$$

Define $\overline{\rho}$ as the restriction of $\overline{\overline{\rho}}$ to HJ and $\overline{\varphi} = (\_)^- \circ \overline{\overline{\varphi}}$. These $\overline{\rho}, \overline{\varphi}$ extend those of [3].

Table 6. HJ and λ_H-calculus

$$u, v, t ::= xl \mid \lambda x.t \mid (\lambda x.t)(v :: l)$$
$$l, l' ::= [] \mid t :: l$$

$$Ax\,\frac{}{\Gamma; A \vdash [] : A} \qquad Der\,\frac{\Gamma; A \vdash l : B}{\Gamma, x : A; - \vdash xl : B}$$

$$L\supset\ \frac{\Gamma; - \vdash t : A \quad \Gamma; B \vdash l : C}{\Gamma; A \supset B \vdash t :: l : C} \qquad R\supset\ \frac{\Gamma, x : A; - \vdash t : B}{\Gamma; - \vdash \lambda x.t : A \supset B}\,x \notin \Gamma$$

$$beta - cut\,\frac{\Gamma, x : A; - \vdash t : B \quad \Gamma; - \vdash v : A \quad \Gamma; B \vdash l : C}{\Gamma; - \vdash (\lambda x.t)(v :: l) : C}\,x \notin \Gamma$$

$$(\beta_H) \quad (\lambda x.t)(v :: l) \rightarrow insert(l, (subst(v, x, t))$$

where

$$\begin{aligned}
subst(v, x, xl) &= insert(subst(v, x, l), v)\\
subst(v, x, yl) &= y\,subst(v, x, l),\ y \neq x\\
subst(v, x, \lambda y.t) &= \lambda y.subst(v, x, t)\\
subst(v, x, (\lambda y.t)(u :: l)) &= (\lambda y.subst(v, x, t))(subst(v, x, u) :: subst(v, x, l))
\end{aligned}$$

$$\begin{aligned}
subst(v, x, u :: l) &= subst(v, x, u) :: subst(v, x, l)\\
subst(v, x, []) &= []
\end{aligned}$$

$$\begin{aligned}
insert([], t) &= t\\
insert(l, xl') &= x\,append(l', l),\ l \neq []\\
insert(l, \lambda x.t) &= (\lambda x.t)(u :: l'),\ l = u :: l'\\
insert(l, (\lambda x.t)(u :: l')) &= (\lambda x.t)\,(u :: append(l', l)),\ l \neq []
\end{aligned}$$

$$\begin{aligned}
append(t :: l, l') &= t :: append(l, l')\\
append([], l') &= l'
\end{aligned}$$

Proposition 5. $\overline{\varphi} \circ \overline{\rho} = id$.

Corollary 2. *The typable subcalculs of λ_H^+ is confluent.*

Proof. Since the typable subcalculus of λ_H^+ is strongly normalising, it suffices, by Newman's Lemma, to prove uniqueness of normal forms. Suppose $t \to^* t_1, t_2$ and that both t_i are cut-free. By the simulation results above, $\overline{\overline{\rho}}(t) \to^* \overline{\overline{\rho}}(t_1), \overline{\overline{\rho}}(t_2)$ and, since $\overline{\overline{\rho}}$ preserves cut-freeness, both $\overline{\overline{\rho}}(t_i)$ are cut-free. As $\overline{\overline{\rho}}$ preserves typability, $\overline{\overline{\rho}}(t_1)$ and $\overline{\overline{\rho}}(t_2)$ are obtained within the tq-protocol and, by confluence of the tq-protocol, $\overline{\overline{\rho}}(t_1) = \overline{\overline{\rho}}(t_2)$. Now crucially $t_1, t_2 \in HJ$ because t_1, t_2 are cut-free. Then, $t_1 = \overline{\varphi}(\overline{\rho}(t_1)) = \overline{\varphi}(\overline{\overline{\rho}}(t_1)) = \overline{\varphi}(\overline{\overline{\rho}}(t_2)) = \overline{\varphi}(\overline{\rho}(t_2)) = t_2$. ◻

Now let us turn to the relation between HJ and NJ. It is convenient to give the syntax of λ-calculus as

$$\begin{aligned} M, N &::= x \mid \lambda x.M \mid app(A) \\ A &::= xM \mid (\lambda x.M)N \mid AM \ . \end{aligned}$$

Translations Ψ and Θ between HJ and NJ are given in Table 7.

Table 7. Translations Ψ and Θ

$\Psi(x) = x[]$ $\Psi(\lambda x.M) = \lambda x.\Psi M$ $\Psi(app(A)) = \Psi'(A, [])$ $\Psi'(xM, l) = x(\Psi M :: l)$ $\Psi'((\lambda x.M)N, l) = (\lambda x.\Psi M)(\Psi N :: l)$ $\Psi'(AM, l) = \Psi'(A, \Psi M :: l)$	$\Theta(x[]) = x$ $\Theta(x(u :: l)) = \Theta'(x\Theta u, l)$ $\Theta(\lambda x.t) = \lambda x.\Theta t$ $\Theta((\lambda x.t)(u :: l)) = \Theta'((\lambda x.\Theta t)\Theta u, l)$ $=$ $\Theta'(A, []) = app(A)$ $\Theta'(A, u :: l) = \Theta'(A\Theta u, l)$

Proposition 6. $\Theta \circ \Psi = id$ *and* $\Psi \circ \Theta = id$.

Proof. Extend the proof in [2]. ◻

Lemma 3. $\Psi(M[N/x]) = subst(\Psi N, x, \Psi M)$.

Corollary 3. $\Theta(subst(v, x, t)) = \Theta t[\Theta v/x]$.

The promised isomorphism of normalisation procedures is the following

Theorem 1.

1. *If $M \to_\beta M'$ in NJ then $\Psi M \to_{\beta_H} \Psi M'$ in HJ.*
2. *If $t \to_{\beta_H} t'$ in HJ then $\Theta t \to_\beta \Theta t'$ in NJ.*

Hence β and β_H are isomorphic, but β performs normalisation in NJ whereas β_H performs cut elimination in HJ.

5 Further Work

There are two main directions of further work.

First, to extend this work to the other connectives of intuitionistic predicate logic. Challenging seems to be the positive fragment and the treatment in this framework of the anomalies caused by disjunction and reported in [15].

Second, to generalise the whole enterprise to classical logic. The key players should be Herbelin's LKT and $\overline{\lambda}_{\mu}$-calculus [7], Parigot's natural deduction and λ_{μ}-calculus [8] and an appropriate LK^t. We plan to report on this in [4].

Acknowledgements

We thank Samson Abramsky for encouragement and guidance, Luís Pinto for his enthusiasm about this work and René Vestergaard for many chats on structural proof theory.

References

1. V. Danos, J-B. Joinet, and H. Schellinx. A new deconstructive logic: linear logic. *The Journal of Symbolic Logic*, 62(2):755–807, 1997.
2. R. Dyckhoff and L. Pinto. Cut-elimination and a permutation-free sequent calculus for intuitionistic logic. *Studia Logica*, 60:107–118, 1998.
3. R. Dyckhoff and L. Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212:141–155, 1999.
4. J. Espírito Santo, 2000. PhD Thesis (in preparation).
5. J. Gallier. Constructive logics. Part I. *Theoretical Computer Science*, 110:248–339, 1993.
6. H. Herbelin. A λ-calculus structure isomorphic to a Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn, editors, *Proceedings of CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1995.
7. H. Herbelin. Sequents qu'on calcule, 1995. PhD Thesis, Université Paris VII.
8. M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classic natural deduction. In *Int. Conf. Logic Prog. Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.
9. G. Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.
10. D. Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almquist and Wiksell, Stockholm, 1965.
11. A.M. Ungar. *Normalization, Cut-eliminations and the Theory of Proofs.* Number 28 in CSLI Lecture Notes. 1992.
12. C. Urban and G. Bierman. Strong normalisation of cut-elimination in classical logic. In *Proceedings of TLCA'99*, Lecture Notes in Computer Science. Springer-Verlag, 1999.
13. R. Vestergaard and J.Wells. Cut rules and explicit substitutions. In *Second International Workshop on Explicit Substitutions*, 1999.
14. P. Wadler. A Curry-Howard isomorphism for sequent calculus, 1993. Manuscript.
15. J. Zucker. The correspondence between cut-elimination and normalization. *Annals of Mathematical Logic*, 7:1–112, 1974.